CP-based Social Scheduling

Jacob Feldman, Eugene Freuder, Leonid Ioffe, James Little, Oscar Manzano

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{j.feldman | e.freuder | l.ioffe | j.little | o.manzano}@4c.ucc.ie

Abstract

This demonstration will showcase how CP-based schedulers can be plugged into popular social utilities such as Google Apps or Facebook. In particular, we will demonstrate a CP-based meeting scheduling plug-in for Google Calendar that enriches social event management tools with automatic scheduling facilities.

Introduction

Social applications revolve around people and their relationships. They not only enhance social experiences but "can solve real world tasks where the social graph assists us in making decisions. They can be useful to facilitate meetings, purchases, recommendations, information management" (see Social Design Best Practices, *http://code.google.com/apis/opensocial*).
Nowadays millions of people are using social utilities such as Facebook, MySpace, Google Apps, and others to setup meetings and to coordinate different events with friends and colleagues who work, study and live around them. Even large enterprise-oriented systems such as IBM Lotus Notes transfer themselves into social utilities to take into consideration real-life interests and personal schedules of corporate employees (*http://www-306.ibm.com/software /lotus/products/connections*).

The majority of social utilities provide powerful while friendly web interfaces that help users to put events, meetings, and other common activities into shared calendars. However, the actual event/meeting scheduling and rescheduling remains a prerogative of human administrators with manual selection of time and location that may or may not satisfy all participants. As was pointed at ICAPS-2007, "current status still remains behind automatic scheduling abilities being incorporated into these systems" (*http://abotea.rsise.anu.edu.au/satellite-events-icaps07/demos/5/ra.pdf*). Automatic scheduling is especially important for organizations that have multiple, frequently overlapping meetings. Manual meeting rescheduling with repetitive renegotiations between different participants becomes the real issue.
At the same time automatic meeting scheduling with various temporal and resource constraints is one of the most studied area of Constraint Programming (CP) with multiple success stories. So, why do we not see CP-based schedulers being incorporated in social utilities? Below we concentrate on this question and offer an architectural approach for CP-based "Social Scheduling" that is intended to work with different social networks utilizing different CP solvers. To demonstrate the proposed architecture, we have implemented a CP-based plug-in for Google Calendar that enriches social event management tools with automatic scheduling capabilities.

Scheduling in Social Environment

There are many off-the-shelf event management products with powerful build-in automatic meeting schedulers. However, they operate outside of social networks and do not have access to the information about actual people availability. A big problem for adaptation of these products in the social context is their "heavy weight" – offering too many scheduling capabilities they request too much from their potential users. Instead of looking at the social world from a scheduling perspective, it is necessary to recognize that contrary to manufacturing or airline businesses the social world does not consider a scheduler as a mission-critical component.

To be accepted by the social environment, an automatic scheduler should become a simple pluggable component that will be invoked only when necessary with minimum effort and maximal utilization of information already provided by humans to different social networks. Simplicity and intuitiveness of the scheduling user interface is a key for success. Today trivial "meeting schedulers" that just offer to all participants several timeslots to choose from are among the most popular scheduling tools. They are less concerned about scheduling power but more about natural integration with calendaring, broadcasting, notification, and other services already provided by social utilities.

CP may play extremely well in this context. Here CP's ability to dynamically formulate a scheduling problem by adding/removing custom constraints "on-the-fly" and not worrying about the scheduling algorithm become the real value. A CP-based scheduler would be able to address real scheduling and resource allocation issues that are beyond the reach of simplified meeting schedulers. This is

especially important for social networks that integrate corporate and personal events. For example, for a recurring meeting that goes over 12 weeks the current Google scheduler will produce a "No match" answer if a particular location is not available during last two weeks. The advanced CP scheduler will schedule the first 10 weeks and find a replacement location for the last two weeks or will produced an appropriate warning.

The automatic scheduler should be able to validate manual scheduling decisions, provide warnings and recommendations, and propose automated solutions. It should avoid "no match" responses, but rather produce "explanations" and possible alternatives. The scheduler should be highly interactive and customizable. Specialized constraints should enable the creation of bespoke schedulers to reflect the needs of individuals, groups, and/or organizations such as working hours, time zones or preferred blocked days or day parts. Some constraints might be acquired with machine learning methods to personalize painlessly. The scheduler should be reactive and proactive; changes in personal calendars maintained by potentially different social utilities should "propagate" and trigger appropriate scheduler actions of the centralized scheduler. The automatic scheduler should try to minimize changes in previously scheduled events. The scheduler could be "always on", producing warnings and recommendations in response to changes.

The above requirements are not unusual for CP-based online schedulers. Practical social event scheduling scenarios do not create really challenging constraint satisfaction and optimization problems – most of them could be handled by standard search algorithms provided by almost any CP solver. The real challenge is in setting up an architecture that incorporates CP-based schedulers into existing social networks while satisfying the described requirements.

Architecture

At the high level the proposed architecture for a social scheduling framework is presented in Figure 1. We try to keep the architecture vendor-neutral so it can work with different social utilities and different CP solvers. However, today Google Calendar is dominating the online personal calendar space being actively used inside different social networks. Google even initiated a common OpenSocial API for social applications across multiple websites. While other social utilities also provide their own event management applications with powerful collaborative interfaces (see Facebook and MySpace "Events"), Google Calendar Server is the one that allows external applications not only to read events but also to modify them programmatically. An ability to change event attributes and to add new ones is essential for automatic schedulers. Google Calendar Server and our own Event Scheduling Server play a key role in the described

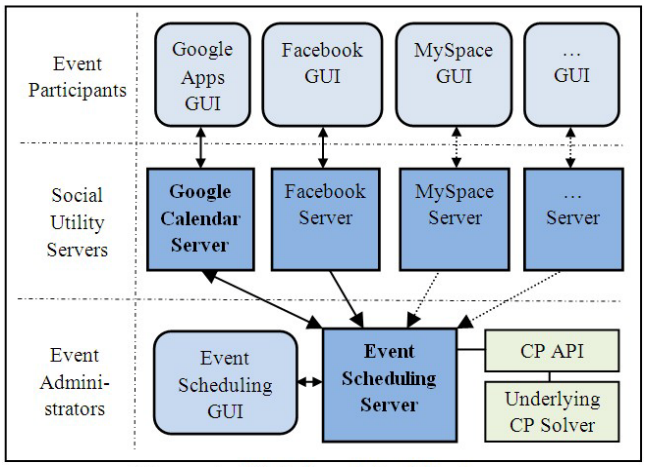

Figure 1: High-Level Architecture

architecture. While we will receive events from different social utilities we will use Google Calendar Server as a placeholder for all events controlled by our advanced scheduler. From an event scheduling perspective all users are divided in two categories:

- Event Participants: also known as meeting guests who collaborate with others only through native web interfaces such as Google Calendar or Facebook/MySpace Events
- Event Administrators: who also have access to a more powerful Event Scheduling GUI and might define different event constraints and preferences and execute an automatic event scheduler to find the optimal start times and locations.

The Event Scheduling Server has three major functions:

- Interaction with a special Event Scheduling GUI
- Communication with involved social utility servers
- Implementation of an automatic event scheduler.

Through the Event Scheduling GUI event administrators are able to see and modify all events that are currently under their control defining earliest start and latest finish times, durations, required and optional participants, etc. They are kept informed about all automatically discovered conflicts in people and location availabilities, are able to setup priorities for conflicting events, and to launch an automatic scheduler to resolve the conflicts. The scheduler is implemented using a generic CP API and may switch between different underlying open source or commercial CP solvers without changes in the scheduler code (see *http://cpinside.com*).

Use Case Scenario

Let's consider a scenario when a company ABC decides to improve its multi-departmental meeting scheduling with commonly used software taking into consideration real-world availability of its employees. It may choose any calendaring software from Lotus Notes to Google

Calendar. We will assume that for this demo Google was selected. The ABC meeting administrator ("admin") creates two Google calendars:

- **ABC Meetings Calendar** – for company events only. All ABC employees could see all events they are involved in and accept/reject participation in these events via the native Google Calendar interface. Only ABC meeting administrator(s) could make changes in this calendar using either the native interface or an advanced Event Scheduling GUI
- **ABC Personal Events** – another Google calendar where ABC's employees are supposed to put their personal events such as vacations, business trips or doctor visits to inform the company about their unavailability. Only employees themselves could change these events.

Additionally ABC's employees may use other social utilities to keep the company informed about their unavailability. For example, in Facebook people may create private events and invite an ABC admin to attend them. Alternatively, they may let the company know that all their public Facebook events of a certain category should be considered during ABC meeting scheduling. People share with their employer only email addresses under which they are known on the selected social networks, but they remain in complete control which events to share and which to hide.

Now let's look at how the admin communicates with the system. The admin always may use the standard Google Calendar GUI to access the ABC own Scheduler:

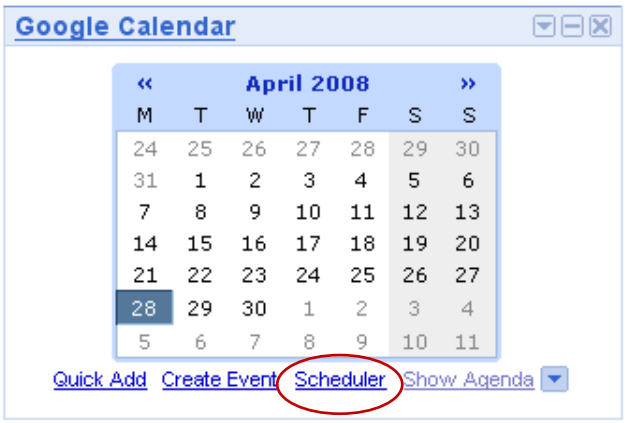

Figure 2: Launching Event Scheduler from Google Calendar

The admin will be asked to enter authorization requisites and if the authorization phase passed successfully, the Event Scheduling Server will be launched (if it was not launched earlier) on the company's own web application server. Next, the Scheduling Server will download all active company meetings of this admin from the Google Server. All non-ABC events in which participants of the downloaded ABC meetings are possibly involved will also be downloaded from the Google Server. Similar personal events could be downloaded from other social utilities. The scheduler will be automatically executed to validate all downloaded meetings and events against currently active constraints and preferences. Finally, the admin will be presented with a scheduling GUI that may look like the one in Figure 4.

On this screen the admin can see all downloaded ABC events and their status, along with possible warnings and errors generated by the automatic scheduler. All participants who may have scheduling conflicts will be marked in yellow or red based on the seriousness of the conflict, e.g. unavailability of an optional meeting attendee could be ignored (based on a system setting). Unavailability reasons including events downloaded from participants personal calendars might also be presented to the admin. From this view the admin may:

- Modify meeting priorities
- Change scheduling makespan, constraints and preferences, working hours and other company-level settings
- Re-Schedule all or only selected meetings by calling the CP-based Scheduler
- Broadcast all changes back to the Google Calendar with the proper notifications
- Download the latest changes from all social utilities servers
- Open and update individual meetings.

The admin may also deal with one meeting at a time like as in Figure 3:

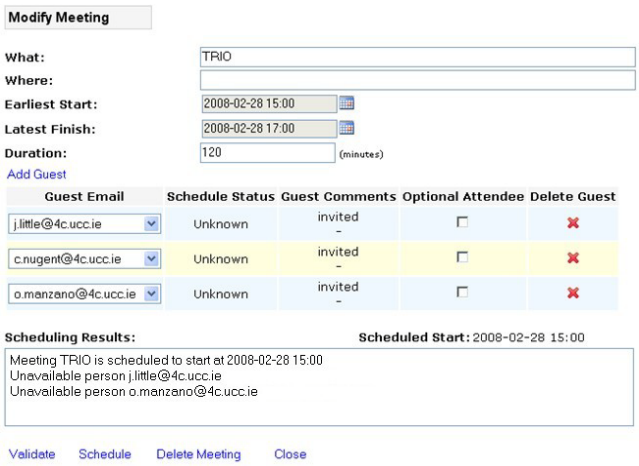

Figure 3: Admin GUI – Single Meeting View

The Single Meeting View allows the admin to enter additional meeting scheduling attributes such as "Earliest Start", "Latest Finish", "Duration". Manual event schedulers usually do not support these attributes and force a user to enter exact start and end event times. It is important this information will be attached to the standard meeting events and saved on the Google Server during

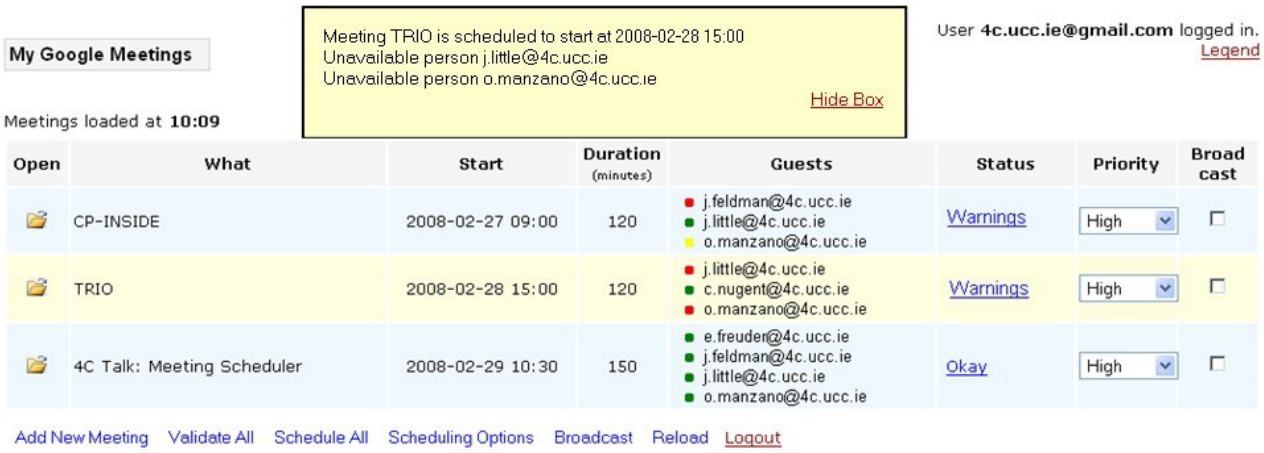

Figure 4: Event Scheduling GUI

broadcasting. If the admin, after making changes, decides to launch the scheduler directly from this view, only this particular meeting will be rescheduled considering all other meetings as frozen.

Implementation

Our current meeting scheduler demonstrates a possible implementation of the 3rd layer described in the Figure 1. We use mainly Java-based development tools to implement 3 major components:

- Event Scheduling GUI
- Event Scheduling Server
- Automatic Event Scheduler.

To support the vendor-neutral architecture described in Figure 1 we have developed a Java package that constitute the BOM (business object model) for our event scheduling problem. This is a simple API that allows a developer to express all business concepts and functions in generic terms without any knowledge of a concrete implementation of social networks and an actually used scheduler. It contains interfaces and partial implementations for such concepts as Meeting, MeetingGuest, Meeting Admin that could be naturally extended to GoogleMeeting or LotusMeeting.

The current implementation uses:

- Apache Tomcat (*http://tomcat.apache.org*) as a foundation for our own Event Scheduling Server. While we could apply any web development technique to implement the GUI, we have chosen OpenRules (*http://openrules.com*) because it allowed us to quickly represent the above graphics using simple Excel forms, to concentrate on the interaction logic naturally represented in business rules, and to automatically deploy the resulting web application on the Tomcat server.
- Java API (*http://code.google.com/apis/calendar*) for Google Calendar that provides an authorized access to the Google Server, allows our server to download all events visible to the admin, attach our own data to the standard Google events and save them back on the Google Server, and finally broadcast all changes introduced on our server. To receive events from other social utilities we applied Java APIs provided again by Google.
- CP-Inside framework recently developed by Cork Constraint Computation Centre (*http://cpinside.com*) that allowed us to represent the event scheduling problem using a generic Java CP API. As a result, the same implementation of our event scheduling problem successfully works with different underlying CP solvers such as Choco (*http://choco.sourceforge.net*) and ILOG JSolver (*www.ilog.com*). While the underlying CP solvers do not provide direct scheduling facilities they work well with a basic scheduling component built on top of the CP-Inside API. The efficiency of the resulting Meeting Scheduler and the overall system has been satisfactory.

The implemented prototype has shown that the proposed non-intrusive approach for plugging CP-based components into modern social networks does not introduce an significant overhead or affects their scalability and efficiency, which otherwise would become a barrier for real-world acceptance. While this demonstration concentrated on meeting scheduling, a similar approach may be applied for vacation scheduling, for scheduling of business trips at various locations, for generation of personalized TV schedules, and other tasks that deal with multiple combinations of business and personal events inside social networks.

Acknowledgements: This material is based upon works supported by Enterprise Ireland under Grant CFTD/06/209 and by the Science Foundation Ireland under Grant No. 05/IN/I886.