From: Proceedings of the Second International Conference on Multiagent Systems. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

A Multiagent Meeting Organizer that satisfies Soft Constraints

Joo-Hwee Lim†, *Jiankang Wu*† & *Siet-Leng Lai*‡

†RWCP Novel ISS Laboratory
Institute of Systems Science
Heng Mui Keng Terrace, Singapore 119597
Email: {joohwee, jiankang}@iss.nus.sg

‡Center for Computer Studies
Ngee Ann Polytechnic
535, Clementi Road, Singapore 599489
Email: lsl@np.ac.sg

As a distributed resource allocation problem, meeting scheduling is a tedious and time-consuming process. This paper proposes a multiAgent MEETing organiZER (AMEETZER) that respresents and reasons with soft constraints related to the meeting attendees and resources. An AMEETZER accepts call-for-meeting request from its user (host) and communicates with other AMEETZERs of the proposed attendees and with the agent managing the meeting rooms to arrive at some commonly free time slot, taking into account the pre-specified and dynamically created hard and soft constraints of all attendees and meeting resources. When necessary, the AMEETZER could perform negotiation of meeting time or relaxation of constraints with or without the intervention of its user. Last but not least, the AMEETZER reminds its user of forthcoming meetings at appropriate times.

With distributed AMEETZERs, meeting requests that involve disjoint sets of attendees and resources can be scheduled concurrently, as contrast to a centralized approach which faces communication and processing bottlenecks as well as fault tolerance and complex schedule maintenance issues. To further increase the concurrency of the system, each AMEETZER is decomposed into four subagents, *Receptionist* (interacts with user to manage his calendar and preferences and carries out meeting proposal and negotiation etc), *Scheduler* (infers the optimal time (and room) for a meeting under hard and soft constraints etc), *Messenger* (communicates with other AMEETZERs), and *Learner* (learns scheduling preferences). Together the calendars and preferences are used to generate instances of soft constraint set that specify the preference values ($\in [0,1]$) on the meeting times. These preference values are communicated to and inferenced by the host AMEETZER to generate utility-optimal time for the meeting.

Another challenging aspect in real life scheduling problem is the representation and reasoning of hard and soft constraints. In real life, *goals* can also be conveniently expressed as preferrences (soft constraints) since satisfaction-seeking is more realistic than optimality-seeking. Indeed, in fuzzy decision making, the decision set is the intersection (denoted as $\wedge$) (or more general, *confluence*) of the goals and the constraints,

$$\mu_D(X^*) = \max_X\{\mu_G(X) \wedge \mu_C(X)\} \tag{1}$$

where D, G, C are decision set, goal fuzzy set, constraint fuzzy set respectively and μ_D, μ_G, μ_C are their membership functions respectively, X^* is the optimal solution in the domain over which X ranges.

In AMEETZER, conventional crisp constraints (n-ary relations) are extended to fuzzy constraints founded upon fuzzy relations. Constraints imposed on meeting time by the attendees or meeting rooms can be expressed as fuzzy relations defined on the Cartesian product space of *Day* (in a week) $\times$ *Time* (in a day), denoted as T. In a simple case, a time preference is represented as

$$c_i = \{\mu(t) | t \in T\} \tag{2}$$

where $\mu(t) \in [0,1]$ represents the preference for a meeting to be scheduled at time t as seen by the system, an attendee, or a meeting room. While $\mu(t) = 0$ denotes an impossible time for meeting possibly due to other commitment, $\mu(t) = 1$ indicates maximal preference. Priorities among attendees (or meeting resources) and meetings can easily be incorporated by introducing weighting coefficients for c_i or sieving c_i through modifiers (or linguistic hedges).

Using the framework of (1), AMEETZER infers the most preferred time t^* by all N attendees as

$$\mu(t^*) = \max_t\{\min_{i \le N} c_i(t)\} \tag{3}$$

which provides a well-defined utility measure (optimal degree of joint constraint satisfaction).

Ameetzer is implemented in Java to take advantage of its object-orientation and platform independence.