Distributed Agenda Management Through Decentralised Vowels Co-ordination Approach

João L.T. da Silva[1] and Yves Demazeau[1]

Laboratoire LEIBNIZ-IMAG,
46, Avenue Felix Viallet
38031 Grenoble, France
Phone: +330476574807
Fax : +330476575081
{Joao-Luis.Tavares, Yves.Demazeau}@imag.fr

Abstract. In this paper we present an application to a Distributed Agenda Management problem through a decentralised multi-agent co-ordination model. Each agent represents a user's personal agenda, which must to co-ordinate with other agendas in order to reach a joint commitment, such as a meeting scheduling, respecting user privacy and local constraints. Constraints represent personal preferences and committed engagements that must take into account during a search for a globally consistent schedule. The aim of this paper is illustrate the application of the decentralised co-ordination model addressing distributed satisfaction constraints through negotiation, plan relations and social dependence.

Paper Type:

Paper Track

keywords:

- Inteligent Agents
- Planning and Scheduling

Conference topics:

Multi-Agent Systems and Distributed AI

Distributed Agenda Management Through Decentralised Vowels Co-ordination Approach

João L.T. da Silva[1] and Yves Demazeau[1]

Laboratoire LEIBNIZ-IMAG,
46, Avenue Felix Viallet
38031 Grenoble, France
{Joao-Luis.Tavares, Yves.Demazeau}@imag.fr

Abstract. In this paper we present an application to a Distributed Agenda Management problem through a decentralised multi-agent co-ordination model. Each agent represents a user's personal agenda, which must to co-ordinate with other agendas in order to reach a joint commitment, such as a meeting scheduling, respecting user privacy and local constraints. Constraints represent personal preferences and committed engagements that must take into account during a search for a globally consistent schedule. The aim of this paper is illustrate the application of the decentralised co-ordination model addressing distributed satisfaction constraints through negotiation, plan relations and social dependence.

1 Introduction

Advances in the Distributed Artificial Intelligence domain has provided the user with more flexible, dynamic and autonomous systems that are able to complex co-operation with each other. As a matter of fact, since distributed systems are spreading rapidly on the Internet the user needs more and more functionalities to manage heterogeneous and distributed knowledge. The Multi-Agent Systems (MAS) field has proposed techniques for co-ordination among artificial automated agents which provide those functionalities to real problems as it has been reported in [1].

One of the applications that we are concerned with is of the Distributed Agenda Management domain. This application consists of multiple independent agents that represent a user's Agenda and that must co-ordinate themselves in order to solve a scheduling problem. This domain is open and highly dynamic since information such as meeting hours, user preferences or user priorities can change over time and even during a negotiation phase. Many works have addressed the problem of the distributed meeting scheduling [6, 5, 7, 14, 9] through Distributed Constraint Satisfaction Problem approach [18, 17] or Distributed Resource Allocation [12].

The purpose of our work is to specify a Distributed Agenda Manager where agents can manage calendar appointments and schedule meetings on behalf of user according its individual preferences and social dependencies ruled by an organisational structure. The interest in take agenda management application through a decentralised multi-agent co-ordination approach lays on its natural distribution and the need for privacy of personal user information. As a matter of fact, we provide a co-ordination model which is decentralised on the Vowels Paradigm and we illustrated a multi-agent specification for the agenda application under this approach.

2 The Vowels Co-ordination Model

Our decentralised co-ordination model is based on the Vowels approach focusing particularly on plan relations and social dependence. The Vowels paradigm [4] describe a methodology to build multi-agent systems based on a componential principle to reason in terms of Agents, Environments, Interactions, Organisations and their dynamics as a core of a Multi-agent Oriented Programming approach.

Our model is oriented to every component of a MAS, addressing some co-ordination approaches into an integrated way. At the organisational level, we take Sichman's social reasoning mechanism [16] and we extend it to cope with planning approach also at the Interaction level. The plan relations are borrowed from Martial's work [11] about co-ordination actions among planning agents. In this case, we utilise potential plan relations at agent and organisation levels through the extension of the DEPendence NETwork (DEPNET) from [16]. At Environment level, we associate a description of the activities, resources and world representation through a TÆMS-like approach [3]. In the next section, we present a short overview about our research on decentralised co-ordination model under Vowels approach[1].

2.1 Co-ordination requirement dependencies

Our central claim is about relationships between plans, actions, goals and specification of co-ordination dependencies. Dependence is defined through classes that describe co-ordination requirements concerning resource, goal, plan and action relationships. For instance, an action requirement could be related to *simultaneity constraints* (Mutex) or *producer/consumer conflicts*; with regards to co-ordination mechanisms, the former can be managed through scheduling techniques, while the latter is about synchronisation.

In MAS, these dependencies account for multi-level constraints among multi-agent components: *personal constraints* at Agent level; *situational constraints* at Environment level; *relational constraints* at Interaction level and *organisational constraints* at Organisation level. We have proposed a co-ordination approach [2] that aims to decentralise the dynamic of MAS by splitting the control among MA components according their specifications and skills. In our work, goal and plan requirements are treated at Agent level while action and resource requirements are take into account in the Environment component.

[1] A broader coverage of the model specification and description can be found in [2]

2.2 Co-ordination Model Description

(A)gent Level Co-ordination: at this level, a co-ordination problem is mainly concerned with plan synchronisation, action/resource dependencies, goal evaluation and scheduling optimisation. We take into account *personal constraints* generated by the agent's internal reasoning which is concerned with plan relationships and task dependencies. For our Agent level co-ordination, we have defined a hierarchical planning representation with additional description relations, which extends an internal representation approach to deal with social dependence [16].

(E)nvironment Level Co-ordination: resource management and *situational constraints* at the level of tasks concerning multiple agents acting together are dealt by the (E) component. To solve certain task and resource dependencies, we assume a relationship description at task structure based on co-ordination mechanisms from Decker's GPGP approach [3]. Thus, methods to execute on the planning level take into account some relations like Enables, Facilitates, Cancel and so on, that are represented into the task structure (local or global dependence network at organisation level). Resource relationships are taken as dependencies between plans as part of the action description that require these resources. Thus, co-ordination actions such as synchronisation can be triggered.

(I)nteraction Level Co-ordination: this level is concerned with *relational constraints*, meaning multi-agent communication through message passing specification, protocol management and negotiation requirements. In order to reach negotiation requirements, we describe a set of plan relations based on co-ordination action taxonomy [11]. This leads us to the description of potential communication needs through an intersection procedure between the task structure (E) and the dependence network (O). Additionally, some constraints detected from these intersection operations may guide in the choice of adequate protocols.

(O)rganisation Level Co-ordination: the Organisation notion takes into account roles, relations among them and social dependence among agents [16]. A role is an abstraction of an agent behaviour which interacts with other roles. This behaviour is described through global goals and plans that play those roles in a MAS by using the same external description structure used into the agent model. At social level, the set of roles interacts one with the other through relations and dependencies among them. These relations determine the existence of a link between two roles and the need of interaction between agents. Throughout this external description of other's skills and goals, we are able to cope with plan/action/resource relationships that embark an implicit co-ordination by coalition formation through complementary needs among agents. Additionally, these coalitions will influence the local control at each multi-agent component to evaluate plan priorities and reschedule tasks.

3 Distributed Agenda Management

The Distributed Agenda Management we present in this paper is concerned about temporal co-ordination and social dependence. We assume each Agenda as an autonomous agent acting for the service of a particular user. Hence, the main problem is to find out a local schedule as a result of a globally consensus about distributed constraints among the agents. Constraints are defined through user preferences, internal task dependencies and external co-ordination requirements, such as time/resource availability and plan relationships.

Agenda agents can manage calendar appointments and schedule meetings on behalf of user according its individual preferences and social dependencies ruled by an organisational structure. The Agenda agent must preserve some privacy requirements of users and thus, reasoning with possibly incomplete information which can be required only to meet some joint constraint. In the following, we will specify the Distributed Agenda Management according to our multi-agent co-ordination model.

3.1 Vowels Specification

The problem lays on the classical domains of distributed constraint satisfaction and distributed resource allocation. Typically, the problem is the satisfaction of a common goal in a physical and/or temporal context, taking into account local constraints to meet a globally consistent solution. In this context, an agenda represents a user and takes into account its personal preferences to manage dependencies between local tasks and to satisfy distributed constraints during a meeting scheduling. To guarantee a certain degree of privacy of the users, we want to avoid exchanging all his/her agenda information during the negotiation of an acceptable meeting for all the agents. For this, a decentralised approach is suitable because it allows a flexible local control, although with more interactions, which lead us to take into account dependencies at interaction level (negotiation and plan modification).

An agent (*host*) depends on the others (*attendees*) in order to accomplish its plan (MeetingSchedule), more specifically with regard to a consistent schedule. There is a dependence between **A-O**, since roles define some action/resource priorities. Between **A-I**, message deadlines and type of protocol define schedule optimisation. Dependencies between **O-E** are related to actions of certain roles, since some roles can have distinct relationship with the environment (different action and perception patterns). For instance, a *host* can cancel meetings while *attendees* do not. **A-E** dependence stands for what actions to do and about resource status (calendar). Some organisational levels may want to preserve some privacy information concerning some roles, this characterise a dependence at (**O-I**). A dependence between **I-E** assumes that in case of fault, **I** can inform **E** to cancel some calendar block action because a deadline threshold to confirm some message has expired.

3.2 Agenda-Agent component design

For (A) component we embark meeting schedule heuristics and constraint management (personal user preferences and engagements already committed). Co-ordination among agents for meeting scheduling is defined in terms of dependencies computed through plan and resources modifications (time shifting and unavailable resources changing).

When the Agenda receives a goal from its user, a plan is chosen to satisfy this goal that relates plan dependencies and agent constraints to be satisfied. For instance, in a task/sub-task dependency, the task *ConfirmMeeting* depends on the environment relationship to the task *VerifyCalendar* in order to carry out the plan *ScheduleMeeting*. The agent infers that a situational constraint implements a co-ordination requirement and activates a distributed component that carries out such task at environment level.

As we claim in [2], agent goals are reached by associated plans which are composed by an ordered set of completed instantiated actions. Goals, such as ScheduleMeeting are represented by an AND hierarchy. Thus, figure 1 illustrates that to achieve *ScheduleMeeting*, a suitable time slot must be obtained through *VerifyCalendar*, an agreement must be meet with *ConfirmMeeting* and *AppendMeeting* can be completed.

Desired appointments on user's Agenda are described through a hierarchical planning approach in such way that agents start by exchange more abstract plans to a first negotiation level. As far as constraints are not solved, agents can proceed with negotiation by refining only those abstract plans that have to be detailed. For instance, when an agent starts a negotiation cycle, it first blocks a time slot under the calendar. In the following, a plan *AppendMeeting* is engaged by the agent containing a *VerifyCalendar* task as showed in figure 1. The Calendar checks for available time slots and should there be a resource temporal conflict, the plan can be refined and the action *blockInterval* is expanded creating a dependency towards the agent, which represents an interaction with the environment (Calendar).

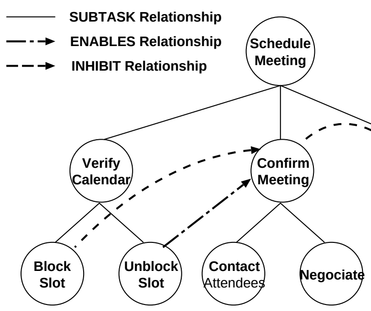

Fig. 1. Hierarchical plan representation from a partial task structure for achieve a ScheduleMeeting goal.

Constraints through user preferences. We define user preferences which are take into account during a negotiation to a consensual meeting scheduling. By the fact, constraints are defined in terms of user preferences and committed engagements. Thus, a meeting is defined as a future joint appointment which must be agreed by all participants concerning these constraints.

Each user creates its own appointments according to its agenda status for the chosen time slots. These appointments can be individual tasks or meetings that describe a goal that be achieved by its agenda. User preferences are categorised by temporal preferences which define priorities over time slots and contextual preferences which refer to personal and professional activities. Priorities are defined in terms of urgency and importance over activities. For instance, an "exam preparation" can be both highly urgent and important while "exercising" is important (for user healthy) but not so much urgent.

The user defines his/her own level of priority over appointments by assigning a combination of urgency and importance level to each activity to be scheduled. Temporal preferences can be setup by choosing work times for each day in the week, optional extended hours, weekends and holidays. Those last ones define how temporal constraints will be considered to negotiate over additional working times. Contextual preferences are defined through task responsability and organisational structure.

3.3 Agenda-Environment component design

The environment is defined according to the group of agents and the calendar definition. At this level, the calendar represents action/resource constraints between agents. These constraints are described through temporal availability from calendar time slots, which are defined as *unavailable*, *occupied* or *free*. An *unavailable* time slot is an interval that possesses a private appointment allocated by the user and it can not be considered to negotiation or relaxation. An *occupied* time slot is allocated to a public appointment, which can be negotiable or can be relaxed according to user preferences. Finally, a *free* slot time can accept a new appointment.

In our approach we assume a description over resources and its relationship with the actions permitted by the environment through a task structure hierarchy. The figure 2 illustrates a partial task structure to check for resource availability (here time slots, but also rooms and equipment). In this work, we use a finite state automata model to describe the resource status (pre- and post-conditions of time slots) and a *mutex* relationship between tasks and resources.

3.4 Agenda-Interaction component design

The (I) component is in charge of managing protocols and priority interactions in terms of message passing. Agents need to exchange relevant information to achieve a globally consistent schedule. The (I) component deals with communication and protocol management. At co-ordination level, we are concerned with

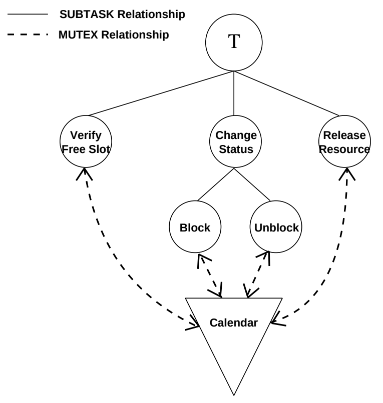

Fig. 2. Example of a partial task structure to manage with resources through suitable relationships to deal with dependencies.

how to choose adequate protocols according to dynamic situation and role allocation. Additionally, this level of co-ordination must guarantee some consistency during message passing. For instance, in a meeting scheduling task, if an attendee can not receive the message, all the related messages have to be cancelled. In this case, a protocol manager may properly provide the atomicity of transactions.

According to dependence constraints, the interaction control may properly manage with distinct protocols for different kinds of information exchange. In the agent connection phase, for instance, the Interaction controller triggers an initial co-ordination mechanism of information gathering which updates the external description (at agent level) of each one.

For our experiment, we have chosen to adapt Sian's co-operative learning protocol [15], because we only need to provide the agents with a simple consensus protocol. The protocol that we need is a simple one, since that a part of constraint management is taken in each component control. In this protocol, an agent elaborates a hypothesis about a believed information, which is shared with a group of associated agents. Each agent evaluates this hypothesis against its own beliefs and then confirms, modifies or rejects it. When the group reaches a consensus, they update their knowledge base with this new information. In our case, the hypothesis stands for a meeting request, which has to be agreed by all attendees or dropped in case of non-consensual result. Several meeting requests can be involved with different agent groups, so the Interaction model has to deal with concurrently scheduling.

At interaction level, we assume that relevant information must be exchanged to build local schedules. Here, we take a traditional meeting scheduling specification to provide the (I) component with the content of the message. We use the protocol management approach [8] which provides the (I) component with a methodology to assemble protocols based on micro-protocol components. The Sian's negotiation protocol is described, in terms of this approach as following:

$$negociate\,(h_i, A_i, m_i) \rightarrow propose\,(h_i, A_i, m_i)$$

Formally, a specification of a meeting to be scheduled is represented by a tuple:

$$m_i = (S_i, l_i, w_i, T_i)$$

A set of agents (A) has as goal to schedule a globally consistent meeting i (m_i). The meeting is proposed by a host $(h_i \in A)$ who will interact with a set of attendees $(A_i \subseteq A)$ in order to find out a consensual schedule. The meeting is required to be schedule at the time interval (S_i) and with a required length in time unites (l_i). Time interval is defined by a couple $\langle date, time\rangle$ for *start* and *end* intervals, so $(S_i = \{\langle d_{start}, t_{start}\rangle, \langle d_{end}, t_{end}\rangle\})$. A priority is assigned to the meeting (w_i) that is directly related to local constraints to be evaluated and solved during co-ordination. Optionally, a free time set (T_i) by which the meeting can be scheduled is proposed by the host.

3.5 Agenda-Organization component design

Co-ordination requirements related to organisational constraints (cf. section 2.1) are concerned with role external description and the organisational structure.

The (O) component takes into account social relations among the participants in order to represent the organisational hierarchy and responsibilities between activities according to the roles that agents play. The agent belongs to a group, a set of roles, which defines the organisational structure of an institution or agent society. We assume that the organisational hierarchy and the responsibility influence preferences during a negotiation to find a consensus. Hence, the role that an agent plays in the society (organisational hierarchy) can bias the search for a consistent schedule. For example, a meeting with the director has a higher priority than that with a colleague, whereas responsibility on a task such as *PrepareLecture* can be more important than a task *OrganizeDesktop*.

At the co-ordination level, as every agent is accessible to the other participants by an external description, social dependencies [16] can be detected with regard to their goals and plans. In this way, the (O) component could manage a thematic group formation according to the meeting joint action (*ScheduleMeeting*) given a range of priorities related to the global importance for this scheduled meeting, for instance.

3.6 An appointment taking example

A user decides to schedule a meeting for an important project and adds an appointment on his/her agenda, providing a time interval, a deadline to the meeting and a set of attendees. His/her Agenda starts by selecting a new goal *ScheduleMeeting* and calculating the constraints for this goal. *VerifyCalendar* is trigger to search for free time slots in the given interval. After that, *AppendMeeting* can be started by selecting *ContactAttendees* and *Negotiate* tasks. The *Negotiate* task will detect a dependency with the (I) component through a distributed component to trigger the negotiation protocol. The list of the each participant's *id* is selected by *ContactAttendees* task while the protocol manager starts to send a $propose(h_i, A_i, m_i)$ to each $a \in A_i$. At this moment, a constraint is added to the agent constraint set since the host (h_i) depends on the attendees (A_i) to achieve the goal $AppendMeeting(m_i)$. The (I) component manages the protocol termination taking into consideration the dependency (A-I) which provides the host with the answers from attendees.

When all attendees send $agree(a, h_i, m_i)$ to the host without any other constraint, the goal *AppendMeeting* can be reached and *ScheduleMeeting* is confirmed. If the host receives a $modify(a, h_i, m'_i)$ message, a new cycle of *ScheduleMeeting* is started in order to take into account the constraints under the proposed meeting modification (m'_i). A confirmation is reached if everyone agree with the new meeting. If a $disagree(a, h_i, m_i)$ or $noopinion(a, h_i, m_i)$ is received, the meeting is dropped and a $cancel(h_i, A_i, m_i)$ is sent, concluding with the negotiation phase. The user is notify that his/her meeting was not possible.

To deal with constraints, the agent systematically checks for free time slots into the meeting interval. If no free slots are found, the agent checks for temporal constraints that could be relaxed, such as time preferences and weak commitments. This last one stands for occupied time slots that are checked for temporal and contextual constraints. That means, whether exists an individual event which can be moved or a meeting with low priority. For instance, a meeting with a person who owns a lower importance than the actual hosting is a good candidate to renegotiation.

4 Summary and Related Work

Distributed Agenda Management falls on classical Distributed Constraints Satisfaction Problem and it is mainly related to scheduling problems and distributed resource allocation. Some of these algorithms have claimed for an exchange of partial knowledge from each agent in order to take all constrained variables into account. In algorithms based on backtracking (synchronous/asynchronous backtracking, weak-commitment) agents have to communicate some tentative values assignment to their neighbours in order to detect whether they have reached a local-minimum. The Flexible Contracting Model approach [14] propose several search biases which allow to properly evaluate alternative choices. Persistent conflicts are dealt through a hierarchical representation of calendars that enable a structured cancellation mechanism. However, these approaches neither assumes the privacy of the user's calendar information nor resource interdependencies and social dependence through organisation structure.

In this paper, we have illustrated a decentralised multi-agent model applied to a dynamic and over-constrained problem that addresses some of those points. The general model is based on dependencies between multi-agent components according to Vowels paradigm. This approach takes resource interdependencies and social dependence into account in order to detect and apply co-ordination mechanisms to achieve a globally consistent solution.

We have implemented this solution in JAVA, through Volcano platform [13], a Vowels-oriented multi-agent platform designed to assist the multi-agent system creation process. Up to now, we have implemented the (A) component and a graphical user interface to provide all agenda functionalities to the user. Interaction features are provide through JATLite [10] and Interaction Protocol Manager [8]. From JATLite we take only the Router layer to basic inter-agent communication. Protocol management is placed over this layer in order to deal with negotiation and information gathering among agents. The first version will be soon operational and we intend to validate it intensively into the MAGMA research group to manage the complex meeting interactions among the members. By the future, we intend to offer it as a free open source to the research community in order to validate in a large-scale real-life environment.

References

1. J.L. Tavares da Silva and Y. Demazeau. Multiagent centered real-time planning. In *III Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agents Systems (SBIA/IBERAMIA'00)*, pages 131–142, Sao Paulo, Brazil, November 2000.
2. J.L. Tavares da Silva and Y. Demazeau. Vowels co-ordination model. *Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, 2002. (to appear).
3. Keith Decker and Victor Lesser. Generalized partial global planning. *International Journal on Intelligent Cooperative Information Systems*, 1(2):319–346, June 1992.
4. Y. Demazeau. Steps towards multi-agent oriented programming. In *1st International Workshop on Multi-Agent Systems (IWMAS'97)*, Boston, 1997.
5. Paulo Ferreira and Jacques Wainer. Scheduling meetings through multi-agent negotiating. In Maria Carolina Monard and Jaime Sichman, editors, *Proceedings of 15th Brazilian Symposium on Artificial Intelligence*, pages 126–135. Springer, 2000.
6. Leonardo Garrido and Katia Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press, 1995.
7. Thomas Haynes and Sandip Sen. Satisfying user preferences while negotiating meetings. In *Second International Conference on Multi-Agent Systems (ICMAS-96)*, page 440, 1996.
8. Marc-Philippe Huget. *Une ingnierie des protocoles d'interaction pour les systmes multi-agents*. PhD thesis, Universit Paris IX - Dauphine, U.F.R. Science des Organisations, Paris, France, 15 juin 2001.
9. J. R. Jennings and A. J. Jackson. Agent-based meeting scheduling: A design and implementation. *Electronic Letters*, 31(5):350–352, March 1995.
10. Heecheol Jeon, Charles Petrie, and Mark R. Cutkosky. JATLite: A java agent infrastructure with message routing. *IEEE Internet Computing*, 4(2):87–96, 2000.
11. Frank von Martial. *Coordinating plans of autonomous agents*, volume 610 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1992.

12. Pragnesh Jay Modi, Hyuckchul Jung, Milind Tambe, Wei-Min Shen, and Shriniwas Kulkarni. Dynamic distributed resource allocation: A distributed constraint satisfaction approach. In John-Jules Meyer and Milind Tambe, editors, *Pre-proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, pages 181–193, August 2001.
13. Pierre-Michel Ricordel and Yves Demazeau. Volcano, a vowels-oriented multi-agent platform. In *Proceedings of the International Conference of Central Eastern Europe on Multi-Agent Systems (CEEMAS 2001)*, Krakow, Poland, 2001.
14. Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling: Preliminary results. In *Conference on Organizational Computing Systems*, Groups within Organizations, pages 55–68, 1991.
15. S. S. Sian. Adaptation based on cooperative learning in multi-agent systems. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-90)*, pages 257–272, Amsterdam, The Netherlands, 1991. Elsevier Science Publishers B.V.
16. Jaime Simão Sichman, Rosaria Conte, Cristiano Castelfranchi, and Yves Demazeau. A social reasoning mechanism based on dependence networks. In A. G. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-92)*, pages 188–192, Chichester, August 8–12 1994. John Wiley and Sons.
17. Takuo Tsuruta and Toramatsu Shintani. Scheduling meetings using distributed valued constraint satisfaction algorithm. In Werner Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pages 383–387, Berlin, Germany, August 20–25 2000. IOS Press.
18. Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.