

# Constraint Satisfaction Techniques and Software Agents

Monique Calisti and Nicoleta Neagu  
Whitestein Technologies AG  
Pestalozzistrasse 24  
8032 Zürich, Switzerland  
Phone: +41 44 256 5000  
Fax: + 41 44 256 5001  
{mca,nne}@whitestein.com

## Abstract

Although constraint satisfaction techniques and agent technology have been combined in several ways, especially over the last few years, it is still debated how, beyond ad hoc approaches, these two computational paradigms can be effectively integrated. The aim of this paper is to analyze and discuss how constraint satisfaction techniques can be used to engineer agents' reasoning and resolve coordination of distributed agents, either cooperative or self-interested, by reporting on our concrete experience.<sup>1</sup>

## 1 Introduction

Even though traditionally distinct, both the constraint satisfaction community and the multi-agent system (MAS) world have been increasingly exploring how to effectively combine concepts and techniques of the two domains. Many distributed problems that require finding a consistent combination of agents' actions can be formalized as distributed constraint satisfaction problems. On the other hand, when information about choices, i.e., variables and constraints, are naturally distributed or can be distributed (for instance, in order to balance out computational efforts), agent technology can be deployed to engineer an effective solution. Before discussing the integration of

constraints based techniques and agents in the following sections, a short overview of essential notions is given in the following.

**Constraint satisfaction** is a powerful computational paradigm which proposes techniques to find assignments for problem variables subject to constraints on which only certain combinations of values are acceptable. The success and the increasing application of this paradigm in various domains mainly derive by the fact that many combinatorial problems can be expressed in a natural way as a Constraint Satisfaction Problem (CSP), and can subsequently be solved by applying powerful CSP techniques [19]. A distributed CSP (DCSP) is a CSP in which the variables and constraints are distributed among distinct autonomous *agents* [23]. Each agent has one or multiple variables and tries to determine its/their value/s. In general, there exist intra- and inter-agent constraints and the value assignment must satisfy all these constraints. In order to verify inter-agent constraints, i.e., constraints between variables controlled by distinct agents, some form of *agent communication* needs to be supported. In both its centralized and distributed formulation, the CSP framework has been extended to allow variants, which increase real-world applicability. These are in particular: (1) *soft* CSP, which allows to the classical notion of constraints the possibility of dealing with important features such as fuzzyness, uncertainty optimization, probability and partial satisfaction [6], and (2) *dynamic* CSP, where constraints may change for external reasons or depending on choices made during the search [13].

---

<sup>1</sup>This paper has been presented to the *Agents and Constraints* workshop, AIIA'04, September 2004, Perugia, Italy.

**Agent Technology.** In general, there is no consensus on a commonly accepted definition of what an agent is and what are, or should be, its main properties. However, the common colloquial understanding is that the term *agent* indicates an entity (person, organization, system, etc.) that (1) acts on behalf of some other entity (an owner), (2) in an autonomous fashion. Thus, an agent is given the mandate to achieve defined goals. To do this, it autonomously selects appropriate actions, depending on the prevailing conditions in the environment, based on its own capabilities and means until it succeeds, fails, needs decisions or new instructions or is stopped by its owner. In the multi-agent systems (MAS) community, *software agents* are conceived as autonomous computational entities situated in some environments which they can sense and act upon in a dynamic (reactive and/or proactive) way according to the environment's changes and their design objectives [21]. A variety of coordination mechanisms, some of them including explicit exchange of information often given in the form of structured messages, have been proposed in the MAS context, such as organizational structuring, contracting, planning and negotiation techniques<sup>2</sup>. In this paper, we focus on how DCSP, sometimes in combination with other techniques such as automated negotiation, can also be used for agents coordination.

However, before entering into the central part of the paper it is important to eliminate ambiguity over terminology. The terms "agent" and "agent communication" have been used in the DCSP and MAS worlds with slightly different meanings. While in the constraint community an agent is a computational entity acting as a decision maker following pre-defined coordination mechanisms (i.e., DCSP algorithms) and sharing an implicit common representation of the world with other agents (i.e., no explicit use of structured communication stack and ontologies), from a MAS perspective an agent is autonomously deciding whether or not to follow specific coordination mechanisms and can communicate with other agents by means of structured semantic-grounded message exchange [2]. In a way, the DCSP agent definition represents a looser version of MAS formalization, also indicated in the MAS community as "weak agency". In our framework, we make use of DCSP modelling princi-

ples and algorithms for MAS agents, which implies the explicit representation of states of the world involving notions of choices as discussed herein.

## 2 Constraint Satisfaction and Agent Reasoning

Solving a problem implies defining and making choices. In this perspective, modelling agent reasoning by making use of CSP methods and techniques requires:

- To represent problem solving or agent reasoning options in terms of *choice making* - i.e., identify variables  $v_1, v_2, \dots, v_i$ .
- Gather and process information about possible *choices* - i.e., values variables can possibly take, grouped in domains  $D_1, D_2, \dots, D_i$  - and related *constraints*, where a constraint is defined by a predicate  $p_k(v_k1, \dots, v_kn)$  that is true if and only if the value assignment of all  $v_ki$  satisfies this constraint.
- Access and apply appropriate *problem solving techniques* (many CSP algorithms are available) to determine the set of possible combinations of choices by taking into account existing constraints.

The techniques for solving CSPs can be subdivided in two main groups: search (e.g., *backtracking* and *iterative*) algorithms and inference (e.g., *consistency*) methods [12]. Consistency algorithms are pre-processing procedures that are invoked before search algorithms. Backtracking methods construct a partial solution (i.e., they assign values to a subset of variables) that satisfies all of the constraints within the subset. This partial solution is expanded by adding new variables one by one. When for one variable, no value satisfies the constraints between the partial solution, the value of the most recently added variable is changed, i.e., backtracked. Iterative methods do not construct partial solutions. In this case, a whole flawed solution is revised by a hill-climbing search. States that can violate some constraints, but in which the number of constraint violations cannot be decreased by changing any single variable value (i.e., local-minima) can be escaped by changing the weight of constraints and/or restarting from another initial state. Iterative improvement is efficient but not complete.

<sup>2</sup>For a good survey on coordination in MAS we recommend chapter 3 of [5].

## 2.1 An Effective Combination: How?

In CSP terms, the assignment of one of the values from a domain  $D_i$  to a variable  $v_i$  corresponds to making a choice. Therefore, if agent reasoning is modelled following a CSP-based approach, an agent decision is taken when a choice for an agent's variable is made. Whenever some constraints involve variables controlled by distinct agents, agent-to-agent interactions may need to be triggered. Furthermore, if there are no possible choices to be made within the existing set of constraints, i.e., over-constrained problem, an agent can try to relax some constraints by interacting (e.g., negotiating) with other agents, humans, environments, etc.

*Example:* Nicoleta and Monique want to meet in Italy in September. The Personal Travel Assistant Agents (PTA) of Nicoleta and Monique take charge of planning the trips in a coordinated manner. To this purpose, several choices need to be made: exact date and location for the meeting, flights for both participants, etc. Every PTA will own the following variables: meeting time =  $v_1$ , location =  $v_2$ , flight =  $v_3$ . The variable domains for  $PTA_N$  are:  $D_1 = \{d1, d2\}$ ,  $D_2 = \{l1, l2\}$ ,  $D_3 = \{A, S\}$ , and for  $PTA_M$  are:  $D_1 = \{d2, d3\}$ ,  $D_2 = \{l2, l3\}$ ,  $D_3 = \{L, S\}$ . To select the meeting date and location, the two PTAs need to communicate and possibly negotiate (i.e., inter-agent constraints negotiation). Then to select the flight, once a date has been selected for the meeting, every PTA can make a choice depending only on intra-agent constraints or user preferences.

As detailed in [1], we used constraints for modelling a description of a desired goal state an agent aims to achieve, and for expressing states of the world involving interdependent choices. In this perspective, an agent's decision making process has been broken down into three main steps: problem modelling, information gathering and combination, and problem solving.

*Problem modelling* corresponds to identify the choices to be made, according to the agent's state and its perception of the world's state, which become the variables in the problem formulation. Then it is necessary to identify which options are available for each of the choices - this generates the domains of values for each of the variables, and finally specify how choices are related - generating and collecting the constraints (relations and exclusions) which apply to problem solutions. *Information gather-*

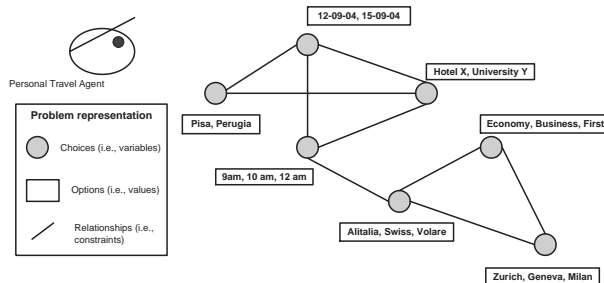


Figure 1: A CSP-based problem representation of the personal travel assistant example.

*ing* and thereby *combination* can involve interaction with other agents. To this purpose we developed and adopted the Constraint Choice Language [20], an agent content language designed to support agent problem solving by providing explicit representations of choices and choice problems. The final choice problem, with all values and constraints gathered, represents a well defined search and solution space. Every valid solution in such space is an acceptable combination of choices (or actions/plans) the agent can make (or execute) according to its goals.

## 2.2 Challenges and Benefits

CSPs have established themselves as a powerful formalism for expressing problems involving multiple interdependent choices. Although some experience is required with this domain, modelling agent reasoning in CSP terms is in general intuitive and most importantly generates problem descriptions with well defined properties and well studied solution techniques - i.e., many CSP algorithms, techniques, libraries and engines are available and ready to be deployed. In particular, since abductive mechanisms have been successfully integrated with CSP [16] and since many real-world problems are abductive by nature, the combination of such computational approaches open up interesting opportunities for abductive agent reasoning and communication, see for instance [8].

An interesting approach combining constraint logic programming and a data model approach, to provide agents with a flexible way to plan and direct their actions

and to manipulate and represent their knowledge is described in [4]. The author explores the declarative use of constraints within a BDI Agent framework to represent knowledge as complex quantified constraints and apply these techniques to a courier scenario where cooperating agents communicate, delegate and exchange desires and information using Generalized Partial Global Planning mechanisms to solve a given set of tasks.

In [11], a constraint-based agent framework is proposed for designing, simulating, building, verifying, optimizing and debugging *controllers*, which are used as agents' reasoning engines. In this model, each agent contains a constraint based controller which communicates with the agent environment through an interface called agent body. According to the reports regarding the environment received from the agent body and agent's internal specified constraints, the controller based on a constraint solver determines the next action/s to be taken by the agent. Most of their work has been applied to robot soccer players.

However, not always and/or not entirely agent logic or even system requirements can be properly captured and expressed in terms of choice making. For instance, when values to be assigned to variables are not necessarily known a priori, when there exist too many variables or domains are too large, it may be more opportune to adopt other reasoning approaches.

### 3 Constraint Satisfaction and Agent Coordination

As anticipated earlier, many distributed problems and applications such as planning, scheduling, resource allocation, configuration, etc., can be represented as distributed constraint satisfaction problems, in which the variables and constraints are distributed among distinct communicating agents. In this case, distribution of computation (in the form of distinct software agents) is adopted as a mean to engineer a solution in a more effective way, and agents are usually assumed to be *cooperative* - see Section 3.1.

On the other hand, in many multi-agent based applications, coordination of distinct agents correspond to distributed constraint satisfaction problem. In this case, depending on whether agents are cooperative or self-interested it is possible to either adopt existing DCSP al-

gorithms or combine and modify DCSP and CSP techniques in a way that self-interests can be respected - see Section 4.

#### 3.1 Cooperative Agents for Effective DCSP Solving

The most trivial algorithms for solving DCSPs are the *centralized method* and the *synchronous backtracking*. In the first case a leader agent gathers all information about the problem, variables' domains and the constraints between them and then solves the CSP alone using classic centralized constraint satisfaction algorithms. This approach is quite inefficient since there are costs associated with gathering information, there are unused resources (i.e., agents sitting idle) and there is no data (both variable values and constraints) privacy. The synchronous algorithm assumes that agents agree on an instantiation order. The first agent generates a partial solution that is submitted to the second agent, which generates an extension to the partial received solution and sends it to the third agent and so on. If the solution cannot be extended then a backtracking message is sent to the previous agent. With this approach there are still costs to decide the transmission of the instantiation order; furthermore, only one agent is active at a time.

Several *asynchronous search* algorithms that allow agents to work in parallel and asynchronously have also been developed [26]. For instance, in the asynchronous backtracking (ABT) algorithm agents work in parallel and asynchronously. The ABT algorithm applies to binary and directed constraints [26] (i.e., for every binary constraint between any two variables only one of the two agents is responsible for evaluating the constraint). Every agent instantiates its variable  $v_i$  concurrently and sends  $v_i$  assigned value to the agents which own variables constrained by  $v_i$  value. When a constraint evaluating agent detects a constraint violation, a *nogood* message is sent back to the related agents causing backtracking. In asynchronous backtracking algorithm access to variables is restricted to owner agents, but constraints may have to be revealed to others. A different approach to privacy is proposed in [18], where constraints are private, but variables can be manipulated by any agent. This algorithms is called *asynchronous aggregation search* and it differs

from previous approach in that it treats sets of partial solutions, i.e., every agent owns multiple variables that it assigns values to, and exchanged information concerns aggregated valuations for combinations of variables, instead of constraints. The asynchronous weak commitment (AWT) search proposes a variation of the ABT algorithm: a partial solution is not modified, but completely abandoned after a failure [24]. The AWT algorithm is complete if all nogood messages are maintained and it is about 10 times faster than the ABT approach. However, the explosion of nogood messages is the most difficult part to control. To coordinate the different forms of asynchronous interactions, the algorithms establish a static or a dynamic order among agents that determines the cooperation patterns between agents. Nishibe, Yokoo and Ishida [15] discuss and evaluate asynchronous backtracking with different ordering schemes: value ordering, variable ordering and value/variable ordering. In particular, they apply these techniques to the communication path assignment in communication networks.

Other algorithms from centralized CSPs have been further adapted for distributed environments such as the distributed consistency algorithm [27], the distributed breakout algorithm [25] and the distributed partial constraint satisfaction [9].

### 3.2 DCSP and Dynamic Agent-Based Transportation Scheduling

Transport logistics deals with building feasible schedules for vehicle routing and trucks dispatching applications. Such a problem takes into account various business constraints such as customer logistic requirements, visit time windows, fleet capacity, trucks' timetables, and work-hour regulations. Moreover all the information about the trucks availability, clients orders and time windows is distributed among various dispatching centers. DCSPs appear to be an appropriate technology to be applied to transport logistic scheduling (TLS) problems because of the way they can handle the dynamicity of constraints in real-world problems and because of their flexibility in modelling and solving scheduling problems. Furthermore, a DCSP-based representation of the TLS naturally respects the distributiveness of the data among the delivering trucks, the demanding clients and the receiving

clients.

The focus of our work in this domain is on scheduling trucks pick up and delivery times for distribution of orders between different locations. We consider the following three types of agents: an agent for each client which requests orders to be sent; an agent for each order/s receiver and an agent for each truck transporting orders. Given a set of orders, trucks, various possible locations of the clients and receivers, constraints on possible pick up and delivery time, the scheduling problem is solved when it has been defined for all the trucks from where (client location) they should pick up which order and at what time, as well as when they should then deliver each order to specific receiver sites. As displayed in Figure 2, clients agents' orders, which contain the receiver location, the order capacity and the client time windows, are sent to all the truck agents. Truck agents communicate with specific receivers agents in order to gather information about their time windows. Each client agent and receiver agent has geographical information about its location and its neighborhood, i.e. distances to neighborhood locations, roads existence and availability. This information can be gathered by truck agents if needed. Relatively to its current orders, each truck agent may accept or decline new client orders. Each truck agent contains information about the orders it has to deliver and the pick up and delivery time at the respective locations for each order. We modelled this scenario (which represents a just a part of the entire transport scheduling problem) as a DCSP, as the whole problem data is distributed among truck agents, that need to cooperate.

A solution to the scheduling problem can be reached by applying a distributed and asynchronous backtracking algorithm. This algorithm provides a complete resolution of the problem, however, it tends to be slow when the problem size increases. For gaining on efficiency, local search algorithms [22], [25], are more appropriate. Finally, when a change in the system occurs, e.g., change of an order or client time constraints, we look for alternative solutions to the current schedule by studying symmetries of the current change (i.e., equivalent variable assignments) in order to avoid a new computation of the whole schedule from the scratch. The symmetries can be determined by the use of interchangeability algorithms first proposed by Freuder [7] for centralized systems that we extended further for distributed systems in [14].

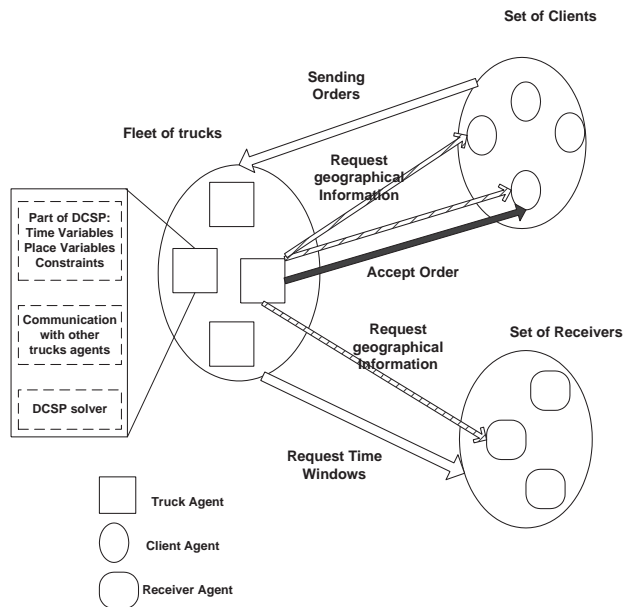


Figure 2: The DCSP-modelled TLS problem is solved by cooperative agents. Truck agents gather all the necessary information from client and receiver agents, model it as a DCSP and solve it by adopting specific DCSP algorithms.

## 4 Enforcing MAS Coordination with DCSP

Several works have deployed DCSP oriented formalism and techniques in order to enforce coordination in multi-agent systems and efficiently model and solve constraint-directed negotiation approaches. One of the earliest approaches that modelled the process of negotiation as a distributed constraint satisfaction problem is the *constraint-directed negotiation* paradigm [17]. Here, negotiation for resolving conflicts, i.e., coordination, in resource allocation is modelled as constraint relaxation and constraints are used for evaluation of existing alternatives as well as for creating new ones. Three main constraint-directed negotiation algorithms are proposed and experimentally validated. A centralized mediator clusters several announcements and bids from multiple agents into atomic contracts (i.e., bids are grouped into *cascades*). The main problem

with this kind of coordination though is that it cannot be applied in scenarios where the centralization of information may not be feasible. Among various DCSP-based approaches for coordination in multi-agent systems avoiding data centralization, in [10], Liu and Sycara proposed to partition the job shop scheduling problem constraints into *constraint types*. Responsibility for enforcing constraints of a particular type is given to *specialist* agents that coordinate to iteratively change the instantiation of variables under their control according to their specialized perspective. Cooperative agents converge to the final solution through incremental local revisions of an initial, possibly inconsistent, instantiation of all variables.

DCSP-based approach can also be used to enforce coordination of software entities that, despite self-interests, need to exhibit cooperative behavior. In the following, a concrete example shortly reporting on our work in this direction [1] is given.

### 4.1 Coordination of Self-Interested Agents for Network Resource Allocation

Allocation of service demands spanning multiple Telecom providers' domains can be considered as a DCSP since the variables (local network resources such as bandwidth, routers, switches, etc.) are distributed among self-interested agents and since constraints exist among them. More precisely, end-to-end routes are decomposed into fragments (i.e., distinct variables) corresponding to independent decision makers (i.e., different agents representing distinct Telecom operators called NPAs, network provider agents).

In DCSP terms, there is one variable per NPA whose values are route fragments through that provider. As a first step, each NPA verifies which local routes can be used to possibly allocate an incoming service demand (depending, for instance, on how much free resources are still available within a specific provider's network). This check is done by performing CSP node consistency mechanisms based on local constraints, which are determined by network availability and providers' control and/or management policies reflected by the corresponding NPAs' goals. Then, since the various route fragments must connect to form a global end-to-end path, NPAs coordinate to select which, among the possible local routes

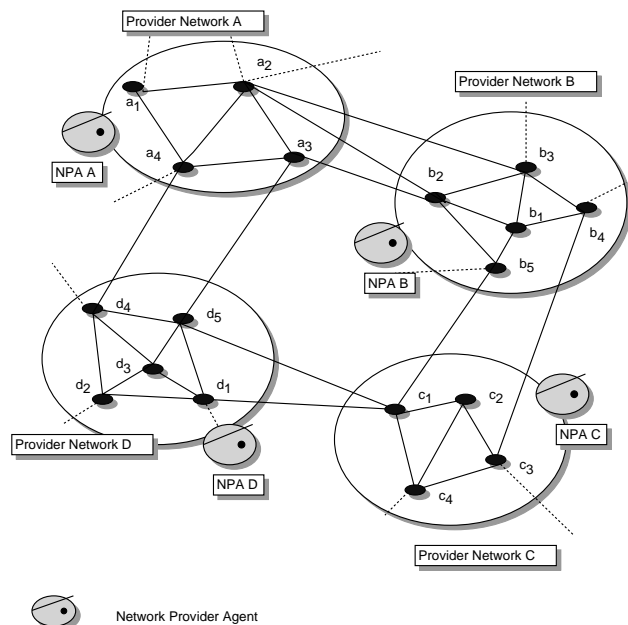


Figure 3: The multi-provider resource allocation process is formalized as a DCSP. The variables (local network resources such as bandwidth, routers, switches, etc.) are distributed among agents and constraints exist among them.

identified in step one, can be retained given existing constraints with neighbor domains. By adopting the distributed arc consistency mechanism [3], the only information that NPAs have to share in order to prune out inconsistent local routes is about inter-agent constraints (in this specific case which network links can be deployed between distinct providers' networks). As proven formally in [1], the combination of such CSP mechanisms guarantees the completeness of this approach. Finally, the space of possible solutions within every NPAs, if not empty, provides the basis for subsequent agent negotiations to decide which specific route to select.

To conclude, like in many real-world applications, in conflict with openness and inter-operability needs, there exists an intrinsic requirement for self-interested and competing entities to keep strategic information such as internal strategies, policies, utility functions, etc. confidential. In our framework, the trade-off between individ-

ual objectives and coordination needs has been reached by the combined use of DCSP techniques and automated negotiation.

## 5 Discussion and Conclusion

This paper discusses possible ways of integrating CSP / DCSP based techniques and MAS by reviewing a number of related works and reporting on our practical experience of combining them for building solutions to real-world problems. This aims, to stimulate discussion in order to identify the main benefits and challenges of merging these two computational approaches around two main questions.

*What can agents do for D/CSP?* Many real-world problems and applications such as planning, scheduling, resource allocation, configuration, etc., can be represented as distributed constraint satisfaction problems, in which the variables and constraints are distributed among distinct problem solvers. Rather than centralizing information into a single point, which can become quite inefficient (both in terms of memory and computational requirements) and insecure (single point of failure), multi-agent systems provide a natural way of distributing problem solving among distinct agents.

*What D/CSP can do for agents?* Constraints can be used for modelling a description of a desired goal state an agent aims to achieve, and for expressing states of the world involving interdependent choices. This makes it possible to deploy CSP solving techniques for facilitating agent decision making. Furthermore, in many multi-agent based applications, coordination of distinct agents correspond to distributed constraint problem satisfaction. In this case, depending on whether agents are cooperative or self-interested it is possible to either adopt existing DCSP algorithms or combine and modify DCSP and CSP techniques in a way that self-interests can be respected. In particular, with respect to agent coordination, constraints can be used to model dependencies on the way agents' interaction can take place, without the need of pre-defining all possible interaction patterns (or protocols), and they can guarantee to find consistent joint decisions even without needing to reveal confidential information.

## References

- [1] M. Calisti. *Coordination and Negotiation for Solving Multi-Provider Service Provisioning*. PhD thesis, Artificial Intelligence Laboratory - Swiss Federal Institute of technology of Lausanne, 2001.
- [2] M. Calisti. Abstracting Communication in Distributed Agent-Based Systems. In *Proceedings of the Concrete Communication Abstractions of the Next Distributed Object Systems Workshop, ECOOP02*, 2002.
- [3] M. Calisti, C. Frei, and B. Faltings. A Distributed Approach for QoS-based Multi-Domain Routing. *AiDIN'99, AAAI Workshop on Artificial Intelligence for Distributed Information Networking*, 1999.
- [4] Stuart W. Chalmers. *Agents and Constraint Logic*. PhD thesis, Department of Computing Science, University of Aberdeen, UK, 2004.
- [5] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, University of London, Queen Mary College, Department of Electronic Engineering, 2000.
- [6] Eugene C. Freuder. Partial Constraint Satisfaction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, Detroit, Michigan, USA*, pages 278–283, 1989.
- [7] Eugene C. Freuder. Eliminating Interchangeable Values in Constraint Satisfaction Problems. In *In Proc. of AAAI-91*, pages 227–233, Anaheim, CA, 1991.
- [8] M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. An abductive framework for information sharing in multi-agent systems. In J. Dix and J. Leite, editors, *CLIMA IV - Computational Logic in Multi-Agent Systems Fort Lauderdale, FL, USA - January 6-7 2004*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2004. To Appear.
- [9] K. Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *Principles and Practice of Constraint Programming*, pages 222–236, 1997.
- [10] L. JyiShane and K. Sycara. Emergent constraint satisfaction through multi-agent coordinated interaction. 1993.
- [11] A. K. Mackworth. Constraint-based agents: The abc's of cba's. In *Proc. of Principles and Practice of Constraint Programming (CP)*, LNCS. Springer-Verlag, 2000.
- [12] Alan Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. Wiley, 1992. Volume 1, second edition.
- [13] S. Mittal and B. Falkenhainer. Dynamic Constraint Satisfaction Problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 90)*, pages 25–32, 1990.
- [14] Nicoleta Y. Neagu. *Interchangeability in Constraint Satisfaction Problems*. PhD thesis, Thesis no 2910, Artificial Intelligence Laboratory - Swiss Federal Institute of technology of Lausanne (EPFL), 2003.
- [15] Y. Nishibe, K. Kuwabara, T. Ishida, and M. Yokoo. Speed-up of distributed constraint satisfaction and its application to communication network path assignments, 1994.
- [16] Nikolay Pelov, Emmanuel De Mot, and Marc De-necker. Logic programming approaches for representing and solving constraint satisfaction problems: A comparison. In *Logic Programming and Automated Reasoning*, pages 225–239, 2000.
- [17] Arvind Sathi and Mark S. Fox. Constraint-directed negotiation of resource reallocations. In Michael N. Huhns and Les Gasser, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Pitman, 1989.
- [18] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *In Proceedings of AAAI*, pages 917–922, 2000.
- [19] Mark Wallace. Practical applications of constraint programming. *Constraints*, 1(1/2):139–168, 1996.

- [20] S. Willmott, M. Calisti, B. Faltings, Macho-Gonzalez S., Belahdar O., and Torrens M. CCL: Expressions of Choice in Agent Communication. In *Proceedings of the Fourth International Conference on Multi Agent Systems (ICMAS-2000)*. IEEE Press (in print), 2000.
- [21] M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
- [22] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *First International Conference on Principles and Practice of Constraint Programming (CP-95)*, 1995.
- [23] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, September/October 1998.
- [24] Makoto Yokoo. Asynchronous weak-commitment search for solving large-scale distributed constraint satisfaction problems. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, page 467, San Francisco, CA, 1995. MIT Press. (poster).
- [25] Makoto Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Second International Conference on Multiagent Systems (ICMAS-96)*, pages 401–408, 1996.
- [26] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.
- [27] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. In *Proceedings of Autonomous Agents and Multi-agent Systems*, 2000.