

Recent Advances in Dynamic, Distributed Constraint Optimization

Adrian Petcu

Artificial Intelligence Laboratory

adrian.petcu@epfl.ch, liawww.epfl.ch/People/apetcu/
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract

Many new technologies like sensor networks, RFID and the internet provide companies with a wealth of interconnected, real-time data sources and have significant potential for automating certain parts of their business processes. The potential of these technologies goes well beyond the limited role of simply collecting, storing and sending data to a central location for processing. We believe that these "smart items" will certainly evolve to more complex embedded systems that can also communicate between themselves, aggregate data and cooperate to take optimal decisions *locally*, with little or no influence from a central system.

Furthermore, at a macro-level, companies themselves act as "intelligent agents" in the marketplace, and often times cooperate in complex supply chains in order to better cope in dynamic and highly competitive environments. In such a context, efficient optimization techniques are essential to allow companies to effectively coordinate to better match their mutual business needs. Distributed solution processes are desirable because they allow the participating actors to keep control of their data, and can offer strong privacy guarantees.

We believe that many key enabling technologies will come from the field of Distributed Artificial Intelligence. Specifically, within that field, Distributed Constraint Optimization is an excellent formalism for problem solving in multiagent systems. Many real-life problems like planning, scheduling and resource allocation can be modeled in this framework, and effectively solved in a distributed fashion.

This paper is a brief introduction to this area. We present some important issues that arise in this domain, and some of our recent results.

Keywords: distributed AI, constrained optimization, resource allocation, scheduling

1 Introduction

The ever-increasing data flows from different sources, shorter response times required from decision makers, privacy concerns and the ever increasing dynamics of the real-world processes, all point into the direction of decentralizing information flows and decision making. Therefore, it is becoming more and more apparent that inexpensive sensors should not only collect and send data to a central location for processing, but more interestingly, evolve into embedded systems that also have processing power, and can cooperate with their peers for local decision making or accomplishing various tasks. Pushing as much business logic as possible towards *smart items* at the edges of the network, close to the data sources is rapidly becoming the only real alternative [Haller and Hodges, 2002; Bornhövd *et al.*, 2005].

However, apart from the obvious benefits, this decentralization also poses a number of challenges. The most important one is that the many actors involved in these distributed decision processes do not have the global knowledge and overview that is available to a centralized process, that has available full knowledge about the problem. The question then is how does one develop techniques that can ensure globally optimal outcomes with only local knowledge? Then, it is also important to guarantee other desirable properties, like robustness in the face of failures, adaptability to changing conditions, preserving privacy, etc.

We believe that a good number of techniques from the artificial intelligence area can provide good solutions to many of these problems. Specifically, *constraint satisfaction/optimization* is a powerful paradigm for solving numerous tasks in distributed AI, including planning, scheduling and resource allocation.

We have recently developed a series of new techniques for *distributed constraint optimization*, based on dynamic programming. Unlike the methods that existed before, our techniques require a very small number of messages (linear in the size of the problem). The maximal message size depends on a parameter of the problem graph, called the *induced width*. Our methods are sometimes orders of magnitude more efficient than previous work when applied to large but loose problems.

Additionally, we have proposed a whole range of techniques that deal with the most common problems in multi-agent environments, related to issues of efficiency, dynamics

of the environment, privacy and the self interest of the agents involved.

We believe that these methods are a particularly interesting foundation for distributed problem solving in dynamic, distributed environments.

The rest of this paper is organized as follows: we will present the constraint optimization framework and then we briefly introduce the techniques we have developed so far for dealing with various aspects of distributed constraint reasoning. Then we describe 3 application domains we have considered, and briefly present the FRODO simulation platform that we have developed. We present some ideas for future work, and then conclude.

2 Constraint Optimization Problems

Constraint programming has been identified by ACM as one of the strategic directions for computing research.

A constraint optimization problem (COP) consists of a set of variables, each with a possible set of values, and a set of constraints that assign utilities or costs to combinations of value assignments. The goal is to find the best assignment of values to variables, such that the sum of utilities is maximized (alternatively, the sum of costs is minimized). Constraints offer a declarative framework for implementing general solution methods and applying them to a wide range of problems. Examples include planning, scheduling, resource allocation, etc. There is a wide body of research that spans a few decades and deals with constraint reasoning problems and their applications in real-life problems.

The main issue in solving constraint problems is computational complexity, as COPs are NP-hard problems.

2.1 Distributed Constraint Optimization Problems

In many real applications, parts of the problem are known to and controlled by individual actors (agents), and there is no central authority that has access to the whole problem. Traditionally, such problems were artificially gathered into a single place, and a centralized algorithm was applied in order to find a solution.

However, this approach may not be feasible or desirable in all circumstances:

- When the actors involved hold sensitive internal information (e.g. production costs), then **privacy** is a big concern, because agents may be reluctant to share all this information with a third party (the centralized solver).
- Many real world problems are **hard to bound**, such that a central authority is feasible. For example, contractors in a construction project simultaneously work on other projects, thus creating an unbounded web of dependencies that cannot be optimized in a centralized fashion.
- **Dynamic problems:** in dynamic systems, problems change at runtime. A centralized solving approach would require that each and every change in the problem is reported to the center, the center computes the new solution, and then redistributes it to the agents. This approach then suffers from latency problems: by the time the problem is re-centralized and solved, it may have already changed! A distributed approach is better suited

to deal with changes in a localized fashion, thus being potentially more responsive to rapid changes in the environment.

As a result, a formalism called Distributed Constraint Optimization Problem (DCOP) [Yokoo *et al.*, 1992] has been recently developed in response to the need of solving such problems which are naturally distributed. In this formalism, each agent knows and is responsible for its own subproblem. Agents then can execute various message-passing algorithms in the quest for the optimal solution.

Figure 1 presents an optimization problem in its two formalizations side by side: (a) presents the centralized model, where the problem data is collected by a center which also solves the problem, and (b) presents the distributed model, where all agents are responsible for their own subproblems, and they communicate in order to find the optimal solution.

The main bottleneck in DCOP problems is no longer computational complexity, but rather communication. Until recently, all distributed algorithms for solving DCOPs required an exponential number of small messages, thus producing important communication overheads.

3 Distributed Optimization Techniques

Previously, essentially all distributed algorithms for constraint optimization were based on some form of backtrack search. Typically, these algorithms have the advantage of a low memory footprint: they require only linear memory (with the notable exception of AWC [Yokoo, 1995], which requires exponential memory as well).

However, they have the important drawback that they typically generate high communication overheads because they require in the worst case an exponential number of small messages. Typical numbers of messages for these types of algorithms are in the range of millions of messages for problems of modest sizes. Considering the performance of the current network technologies, this implies that the runtime of the best search-based algorithms currently available is well above a few weeks even for problems of modest sizes.

3.1 DPOP - a dynamic programming algorithm

We have developed a new technique for distributed constraint optimization, based on *dynamic programming* - DPOP [Petcu and Faltings, 2005b]. DPOP is a utility propagation mechanism, and works on arbitrary constraint problems. It is a complete algorithm, i.e. it guarantees that the optimal solution is found.

The most important feature of DPOP is that unlike previous approaches, it only requires a *linear number of messages* to find the optimal solution, thus generating far less communication overhead. DPOP is a technique that is likely to work very well on large but loose problems.

In fact, experimental results on meeting scheduling problems and resource allocation in sensor networks show that DPOP can handle problems orders of magnitude larger than its predecessors. For more details, please refer to [Petcu and Faltings, 2005b].

3.3 DPOP extensions for dynamic problems

Many real systems are dynamic, i.e. problems change at run-time. For example, new actors may join a negotiation, trucks must be re-routed because of poor weather conditions, production needs to be re-planned because a machine breaks down, etc. Distributed optimization methods are better suited to deal with changes in a localized fashion, thus being potentially more responsive to rapid changes in the environment.

Self-stabilization and fault containment

We believe that **self stabilization** [Dijkstra, 1974; Dolev, 2000] is the key when designing distributed optimization algorithms that must deal with dynamic, error-prone environments. In our context, a self-stabilizing algorithm is one that guarantees that given enough time between changes/failures, the system always converges to the optimal solution of the optimization problem, and then maintains that state. This makes such systems particularly interesting because they can tolerate faults, and are able to cope with dynamic environments.

We have proposed in [Petcu and Faltings, 2005e] an extension of DPOP, the first *self-stabilizing* mechanism for distributed combinatorial optimization. This technique always stabilizes in a state corresponding to the optimal solution of the optimization problem, even upon faults or dynamic changes to the problem.

A *super-stabilizing* extension of this technique can guarantee consistent updates to the variable assignments, even while transiting from one stable state to the next. This means that until the new optimal solution is found, the last-known-good state is preserved. When the new solution is found, all the decision variables are switched *atomically* to their new optimal values. This may be an important *safety feature* in certain applications, where we want to ensure that different controls are applied to a system in a consistent manner, even if they are performed by different actors.

Furthermore, we described a general scheme for *fault containment* and fast response time upon low impact failures/changes. The idea is that it is desirable to confine the effects of small changes that happen in the environment to limited areas, and avoid recomputing the whole solution. This scheme allows for multiple, isolated failures/changes to be handled effectively, without solving the problem from scratch.

Continuous-time optimization

In [Petcu and Faltings, 2005d] we define the *distributed continuous-time optimization problem*. This formalism allows for a *continuous optimization process* in a dynamic system that evolves continuously.

In such a system, optimal decisions are continuously made based on the current state of the environment. However, subsequent changes in the environment may make these decisions not optimal anymore, and it may become more profitable to revise some of them, even if we must pay a price to do so.

For example, an assembly line may experience a failure, and needs to be taken down for unscheduled maintenance. The plant owner may then decide to reassign a maintenance

crew from another (previously scheduled!) task and pay the associated cost, *if* this means saved costs from penalties for not observing contractual duties to customers.

[Petcu and Faltings, 2005d] introduces this formalism that allows for reasoning about optimality in these circumstances, and a corresponding algorithm that can operate in these conditions.

3.4 M-DPOP for systems with self-interested users

In a setting where the participating agents are self-interested, it is possible that they would try to manipulate the optimization process such that the outcome is not the globally optimal solution, but a suboptimal one that is more favorable to themselves. Such manipulations steer the final outcome away from a global optimum which is otherwise achievable.

We have designed a *faithful* optimization protocol (M-DPOP, see [Petcu *et al.*, 2006]) that ensures that it is always in the best interest of the agents to correctly execute the optimization algorithm, without introducing any manipulations. M-DPOP uses the Vickrey-Clarke-Groves (VCG) tax mechanism in such a way that it ensures that the best interest of each agent in the system is always aligned with the best interest of the group, thus ensuring the agents have no incentive to manipulate.

3.5 DPOP extensions for maintaining privacy

In [Silaghi *et al.*, 2006] we introduced a cryptographic mechanism that can be applied on top of DPOP and results in a completely secure protocol. The optimal solution is still found, and no private information (e.g. internal valuations or costs) is leaked, except the final solution.

4 Applications considered so far

We have considered several possible applications of DCOP techniques, from different domains: resource allocation, scheduling, coordination, distributed control. In the following sections we briefly describe these applications.

4.1 Multiagent Meeting Scheduling

We experimented with distributed meeting scheduling in an organization with a hierarchical structure (a tree with departments as nodes, and a set of agents working in each department). Random meetings are generated, each with a certain utility for each agent. The objective is to find the schedule that maximizes the overall utility.

We experimented with problems with hundreds of agents and meetings, which are far larger than the previous state of the art for distributed optimization algorithms. For more details, please refer to [Petcu and Faltings, 2005b].

4.2 Resource Allocation

Resource allocation is the problem of assigning a set of resources to tasks which need to be carried out. This assignment must respect two constraints: the resources must be capable of executing the task, and they must be available during the time that the task must be executed. Resource allocation is a fundamental problem with many practical applications.

Structure of the multiagent optimization framework

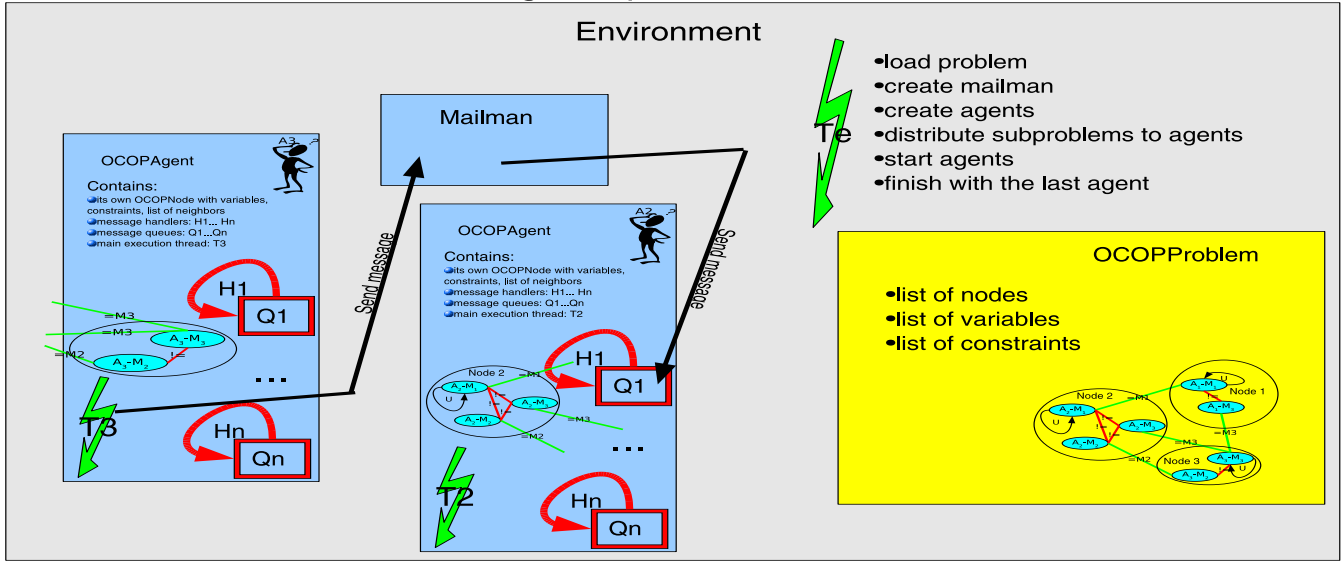


Figure 2: *FRODO* platform: agents simulated as threads that exchange messages with their peers. New optimization algorithms can be easily implemented and tested. The environment provides monitoring and visualization support.

Distributed resource allocation is a special case of resource allocation, where the task of allocating the resources is distributed among a set of agents. We have experimented with two different scenarios: allocating sensors to targets in a sensor network, and allocating servers to user queries in a peer-to-peer network with self-interested users.

Distributed Resource Allocation in Sensor Networks

The distributed sensor network problem formalized in [Gomes *et al.*, 2002] consists of a set of sensors dispersed in the field, and a set of targets that need to be tracked. Each sensor has a “range” parameter that expresses the maximum distance that it can cover; in order to successfully track a target, 3 sensors have to be assigned to that target (triangulation can be applied using the data coming from those 3 sensors). However, some restrictions apply:

- the sensors in the field can communicate among themselves, but not necessarily every sensor with every other sensor (the sensor connectivity graph is not fully connected). The 3 sensors tracking a given target must be able to communicate among themselves;
- any one sensor can only track one target at a time;

This is a hard combinatorial problem that is difficult when the instances considered are large. In [Petcu and Faltings, 2003] we have proposed a number of techniques that increase the scalability of the Distributed Breakout Algorithm [Yokoo and Hirayama, 2000]. Specifically, we were able to solve practical allocation problems with **hundreds of sensors and targets** (see [Petcu and Faltings, 2003] for more details). For more details, screenshots and source code of the simulator, please visit <http://liawww.epfl.ch/Research/sensornets/>.

Distributed Server Allocation in a Peer-to-Peer Network

In this setting we have considered problems where a set of self-interested agents want to each place in-network processing operators on a set of servers. Each agent has its own private preferences about different combinations of operator placements. Each server in the overlay network is able to perform some set of operators, and servers differ in their network and computational characteristics. The task is to provide a distributed algorithm that finds the globally optimal allocation of operators to servers, such that the overall utility of all users is maximized. For more details, please refer to [Faltings *et al.*, 2006].

4.3 Distributed Control

In a highway network, many problems like traffic jams or accidents can be avoided with more effective and adaptive speed limitations. Such adaptive control can be provided by intelligent agents, each one responsible for a highway segment. Neighboring agents can communicate with each other to exchange information about traffic conditions, enforced speed limits, etc. The objective is to make the traffic a fluid as possible, and increase safety.

We have developed a DCOP model of this problem, where the agents correspond to highway segments and they control the speed limitation for their respective segments. Constraints between neighboring agents are designed to model safety restrictions (e.g. enforcing a speed limit of 60 km/h on a segment immediately after a segment with 120 km/h is dangerous), and to increase throughput.

4.4 Distributed Coordination of Robot Teams

Cooperative robotics is an area where multiple autonomous agents often have to accomplish a common goal, such as finding an object, moving an object, patrolling, etc. Often times,

the goal is too complex for each one of the individual robots to achieve by itself: the area to patrol may be too large for a single robot, the object to move may be too heavy, etc. In such settings, the robots have to cooperate in order to achieve the goal, and effective coordination is essential.

In [?] we investigate a scenario where a team of robots must find an odor source as fast as possible. They have sensors for odor and for the wind direction on board, and can track the odor source by reasoning about the direction of the wind, and about the possible location of the source. Team work can lead to finding the source much faster than a single robot could do, but requires effective coordination among the robots. Modeling the coordination problem as a DCOP and executing a variant of DPOP to solve it dynamically as the robot teams evolve in the environment can lead to significant improvements in terms of the time required to find the source, and of the total effort spent by the robots to find the source.

5 FRODO: a Platform for Multiagent Constraint Optimization

We have developed and released a "FRamework for Open/Distributed Optimization" (FRODO), that simulates in a single Java virtual machine a multiagent platform geared towards the implementation and testing of (distributed) optimization algorithms. Each agent is simulated with a Java thread, and communicates with its peers via message exchange.

In Figure 2 we present an overview of the architecture of the platform. Briefly, there is an environment that is responsible for creating the agent threads and message delivery. Within the environment, each agent operates in an autonomous fashion: it loads its relevant subproblem, and then participates in a message exchange protocol with (some of) its peers, as dictated by the optimization algorithm.

The environment can monitor the message exchange, and can present a GUI to the user that shows the current state of the solving process. For example, in the resource allocation example in the sensor network environment, the GUI shows the current allocations of sensors to targets, and the conflicts that are still to be resolved. For more details, and screenshots of the simulator, please visit <http://liawww.epfl.ch/Research/sensornets/>.

In the public version there are two implemented algorithms: Distributed Breakout Algorithm - DBA [Yokoo and Hirayama, 2000], and DPOP [Petcu and Faltings, 2005b]. The framework is extensible, and allows for easy implementation and testing of new optimization algorithms, be they centralized or distributed.

There are also available two testbeds: one for resource allocation in a sensor network, and one for meeting scheduling problems. Both have random problem generators, and GUIs to display the problem instances.

More details, documentation, paper and source download can be found in [Petcu, 2006] and at <http://liawww.epfl.ch/frodo/>.

6 Conclusions and Future Work

Distributed Constraint Optimization is a young and promising research area that has many applications in real-life scenarios. Up until now, distributed algorithms for constraint reasoning (especially optimization) have not yet been applied to large-scale systems mainly due to their prohibitive complexity in terms of number of messages being exchanged.

We have recently developed a new technique for distributed constraint optimization, based on *dynamic programming*. This method requires a very small number of messages (*linear in the size of the problem*), thus dramatically improving the current state of the art in terms of scalability for large optimization problems.

We have also proposed a whole range of additional techniques that deal with the most common problems in multiagent environments: efficiency, dynamics of the environment, privacy and the self interest of the agents involved. These methods can already offer a particularly interesting foundation for distributed problem solving in dynamic, distributed environments.

Future work will focus on experimenting with these new techniques in real applications, to better understand their strengths and weaknesses in these contexts, and to try to develop tailor-made techniques for specific applications of interest.

References

- [Bornhövd *et al.*, 2005] Christof Bornhövd, Tao Lin, Stephan Haller, and Joachim Schaper. Integrating smart items with business processes: An experience report. In *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38 2005)*, 2005.
- [Dechter, 2003] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.
- [Dijkstra, 1974] Edsger W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communication of the ACM*, 17(11):643–644, 1974.
- [Dolev, 2000] Shlomi Dolev. *Self-Stabilization*. MIT Press, March 2000.
- [Faltings *et al.*, 2006] Boi Faltings, David Parkes, Adrian Petcu, and Jeff Shneidman. Optimizing streaming applications with self-interested users using M-DPOP. In *COMSOC'06: International Workshop on Computational Social Choice*, pages 206–219, Amsterdam, The Netherlands, December 2006.
- [Gomes *et al.*, 2002] Carla Gomes, Cesar Fernandez, Ramon Bejar, and Bhaskar Krishnamachari. Communication and computation in discsp algorithms. In *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 40–45, Ithaca, NY, USA, 2002.
- [Haller and Hodges, 2002] S Haller and S Hodges. The need for a universal smart sensor network. Whitepaper CAM-AUTOID-WH-007, Auto-ID Center, Nov 2002.

- [Modi *et al.*, 2005] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *AI Journal*, 161:149–180, 2005.
- [Petcu and Faltings, 2003] Adrian Petcu and Boi Faltings. Applying interchangeability techniques to the distributed breakout algorithm - poster. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03*, Acapulco, Mexico, 2003.
- [Petcu and Faltings, 2005a] Adrian Petcu and Boi Faltings. A-DPOP: Approximations in distributed optimization. In *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP'05)*, pages 802–806, Sitges, Spain, October 2005.
- [Petcu and Faltings, 2005b] Adrian Petcu and Boi Faltings. DPOP: A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-05*, pages 266–271, Edinburgh, Scotland, Aug 2005.
- [Petcu and Faltings, 2005c] Adrian Petcu and Boi Faltings. LS-DPOP: A propagation/local search hybrid for distributed optimization. In *CP 2005- LSCS'05: Second International Workshop on Local Search Techniques in Constraint Satisfaction*, Sitges, Spain, October 2005.
- [Petcu and Faltings, 2005d] Adrian Petcu and Boi Faltings. R-DPOP: Optimal solution stability in continuous time optimization. In *IJCAI05 - Distributed Constraint Reasoning workshop, DCR05*, Edinburgh, Scotland, August 2005.
- [Petcu and Faltings, 2005e] Adrian Petcu and Boi Faltings. S-DPOP: Superstabilizing, fault-containing multiagent combinatorial optimization. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-05*, pages 449–454, Pittsburgh, USA, July 2005.
- [Petcu and Faltings, 2007a] Adrian Petcu and Boi Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-07*, Hyderabad, India, Jan 2007.
- [Petcu and Faltings, 2007b] Adrian Petcu and Boi Faltings. PC-DPOP: A new partial centralization algorithm for distributed optimization. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-07*, Hyderabad, India, Jan 2007.
- [Petcu *et al.*, 2006] Adrian Petcu, Boi Faltings, and David Parkes. M-DPOP: Faithful Distributed Implementation of Efficient Social Choice Problems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-06)*, pages 1397–1404, Hakodate, Japan, May 2006.
- [Petcu, 2006] Adrian Petcu. FRODO: A FRamework for Open and Distributed constraint Optimization. Technical Report No. 2006/001, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2006. <http://liawww.epfl.ch/frodo/>.
- [Silaghi *et al.*, 2006] Marius-Calin Silaghi, Boi Faltings, and Adrian Petcu. Secure combinatorial optimization simulating DFS tree-based variable elimination. In *9th Symposium on Artificial Intelligence and Mathematics*, Ft. Lauderdale, Florida, USA, Jan 2006.
- [Yokoo and Hirayama, 2000] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.
- [Yokoo *et al.*, 1992] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *International Conference on Distributed Computing Systems*, pages 614–621, 1992.
- [Yokoo, 1995] Makoto Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *CP '95: Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pages 88–102, London, UK, 1995. Springer-Verlag.