# Dynamic Distributed Optimization for Planning and Scheduling

**Adrian Petcu**
adrian.petcu@epfl.ch, http://liawww.epfl.ch/People/apetcu/
Artificial Intelligence Laboratory
Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

### Abstract

Constraint satisfaction/optimization is a powerful paradigm for solving numerous tasks in distributed AI, including planning and scheduling. However, up to now, distributed algorithms for constraint reasoning (especially optimization) have not been applied to large-scale systems due to their prohibitive complexity in terms of number of messages being exchanged.

We have developed a series of new techniques for distributed constraint optimization, based on *dynamic programming*. These approaches require a *linear number of messages*, whose maximal size depends on a parameter of the problem graph, called the *induced width*. Thus, these methods are likely to work very well on large but loose problems.

We believe that these methods have features that make them particularly interesting for solving planning and scheduling problems in dynamic, distributed environments.

## Distributed optimization

Distributed optimization problems are problems where each variable and constraint is owned by an agent. This formalism has been developed in response to the need of solving problems which are naturally distributed. Traditionally, such problems were gathered into a single place, and a centralized algorithm was applied in order to find a solution. However, distributed optimization has a number of practical advantages over its centralized counterpart. Centralized solving may be infeasible due to privacy and data integration problems. Dynamic systems are another reason: by the time we manage to centralize the problem, it has already changed.

We have developed a new technique for distributed constraint optimization, based on *dynamic programming* - DPOP (Petcu & Faltings 2005c). Our approach is a utility propagation method, and works on arbitrary topologies using a pseudotree arrangement of the problem graph. It requires a *linear number of messages*, whose maximal size depends on the *induced width* along the particular pseudotree chosen. Thus, our method is likely to work very well on large but loose problems.



## DPOP extensions

**Self-stabilization and fault containment** Self stabilization in distributed systems (Dijkstra 1974; Dolev 2000) is the ability of a system to always reach a *legal* state, and maintain it afterwards. This makes such systems particularly interesting because they can tolerate faults, and are able to cope with dynamic environments.

We have proposed in (Petcu & Faltings 2005d) 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 (until the new optimal solution is found, the last-known-good state is preserved).

Furthermore, we described a general scheme for *fault containment* and fast response time upon low impact failures/changes. Multiple, isolated failures/changes are handled effectively, without solving the problem from scratch.

**Continuous-time optimization** In (Petcu & Faltings 2005b) we define the *distributed continuous-time optimization problem*. This formalism allows for a continuous optimization process in the sense that one can specify *commitment* deadlines for the decision variables in the problem, and the *costs* of changing an already assigned variable are also captured. This formalism can be particularly useful for distributed planning or scheduling with user specified deadlines.

**Approximations and anytime optimization** In (Petcu & Faltings 2005a) we present an approximate version of *DPOP*. This method overcomes the message size shortcoming of DPOP. It is a distributed approximation scheme which trades efficiency for computational effort. The size of the largest message can be adapted to the desired approximation ratio.

We also present an anytime version of this algorithm, which provides increasingly accurate solutions while the propagation is still in progress. At any one time, this method provides known bounds on solution quality. This makes it suitable for very large, distributed problems, where the prop-

agation may take too long to complete.

## Experimental evaluation

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 solved problems with up to 200 agents, 101 meetings, 270 variables, 341 constraints. To our knowledge, these are by far the largest optimization problems solved with a complete, distributed algorithm. Previously, (Maheswaran *et al.* 2004) reported on experiments with 33 agents, 12 meetings, 47 variables, 123 constraints. They use a static optimization algorithm, without any of the features that we describe here.

To our knowledge, there are no other results on self-stabilizing, fault-containing, continuous-time or approximate distributed optimization as yet.

## Applications to planning and scheduling

There are obvious examples of applications of our techniques to planning and scheduling. Settings where privacy is an issue (meeting scheduling, combinatorial auctions in multiple markets, etc) lend themselves well to distributed optimization methods.

*Superstabilization* can be vital in some settings: e.g. while controlling an industrial process in real-time, random settings applied to various installations during the search for the optimal solution can be dangerous. Generating random schedules/plans while searching for the optimal one can be costly.

*Fault containment*: low impact perturbations on a dynamic system require just a few messages to reach the new optimal state. In the best case, our system can deal with $n$ simultaneous perturbations in $O(1)$ time. This leads to fast response time in planning/scheduling applications, when small perturbations are applied.

*Continuous optimization* as described in (Petcu & Faltings 2005b) can be used as a framework to deal optimally with a continuous planning or scheduling process, in which decisions have to be made under time constraints, and then revised as new events unfold.

*Approximations* like (Petcu & Faltings 2005a) can be the only way to deal with large multiagent systems, where centralization is not feasible and the required time and space resources are not available. In such a setting, a configurable anytime algorithm provides increasingly good solutions as time goes by, with a predefined maximal amount of effort.

We have already studied the behavior of our techniques on distributed meeting scheduling problems, and the results are very promising. Currently, we are investigating their application to more general and difficult planning and scheduling problems. Certain combinations of the techniques described here can be particularly efficient in solving a number of real-world problems. We intend to pursue this research avenue and identify such problems and settings where these techniques perform well, as well as understanding their limitations.

## References


Dijkstra, E. W. 1974. Self stabilizing systems in spite of distributed control. *Communication of the ACM* 17(11):643–644.

Dolev, S. 2000. *Self-Stabilization*. MIT Press.

Maheswaran, R. T.; Tambe, M.; Bowring, E.; Pearce, J. P.; and Varakantham, P. 2004. Taking DCOP to the real-world: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS-04*.

Petcu, A., and Faltings, B. 2005a. Approximations in distributed optimization. Technical Report 2005018, EPFL, Lausanne,Switzerland.

Petcu, A., and Faltings, B. 2005b. Optimal solution stability in continuous time optimization. In *IJCAI05 - Distributed Constraint Reasoning workshop, DCR05*.

Petcu, A., and Faltings, B. 2005c. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-05*.

Petcu, A., and Faltings, B. 2005d. Superstabilizing, fault-containing multiagent combinatorial optimization. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-05*.