Anytime Local Search for Distributed Constraint Optimization*

Roie Zivan
Department of Industrial Engineering and Management
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
zivanr@bgu.ac.il

Abstract

Most former studies of Distributed Constraint Optimization Problems (DisCOPs) search considered only complete search algorithms, which are practical only for relatively small problems. Distributed local search algorithms can be used for solving DisCOPs. However, because of the differences between the global evaluation of a system's state and the private evaluation of states by agents, agents are unaware of the global best state which is explored by the algorithm. Previous attempts to use local search algorithms for solving DisCOPs reported the state held by the system at the termination of the algorithm, which was not necessarily the best state explored.

A general framework for implementing distributed local search algorithms for DisCOPs is proposed. The proposed framework makes use of a BFS-tree in order to accumulate the costs of the system's state in its different steps and to propagate the detection of a new best step when it is found. The resulting framework enhances local search algorithms for DisCOPs with the *anytime* property. The proposed framework does not require additional network load. Agents are required to hold a small (linear) additional space (beside the requirements of the algorithm in use). The proposed framework preserves privacy at a higher level than complete DisCOP algorithms which make use of a pseudo-tree ($ADOPT$, $DPOP$).

Introduction

The Distributed Constraint Optimization Problem (DisCOP) is a general model for distributed problem solving that has a wide range of applications in Multi-Agent Systems and has generated significant interest from researchers (Mailer and Lesser 2004; Modi et al. 2005; Petcu and Faltings 2005; Zhang et al. 2005).

A number of complete algorithms were proposed in the last few years for solving DisCOPs (Gershman, Meisels, and Zivan 2006; Modi et al. 2005; Petcu and Faltings 2005). While the completeness of these algorithms is an advantage in the sense that they guarantee to report the optimal solution, this is also a drawback since in order to validate that an acquired solution is optimal they must traverse the entire search space. This drawback limits the use of these algorithms to relatively small problems.

In the case of centralized optimization problems, local search techniques are used when the problems are too large to perform a complete search. Traditionally, local search algorithms maintain a complete assignment for the problem and use a goal function in order to evaluate this assignment. Different methods which balance between exploration and exploitation are used to improve the current assignment of the algorithm (Schaerf 1999). An important feature of most local search algorithms is that they hold the best assignment that was found throughout the search. This makes them *anytime* algorithms, i.e., the quality of the solution can only increase or remain the same if more iterations of the algorithm are performed (Zilberstein 1996). This feature cannot be applied in a distributed environment where agents are only aware of the cost of their own assignment (and maybe their neighbors too) but no one actually knows when a good global solution is obtained.

In (Maheswaran, Pearce, and Tambe 2004; Pearce and Tambe 2007), a new approach was presented to perform local search on DisCOPs in which agents can form coalitions of the size of k. By changing in each iteration the maximal positive gain a k size coalition proposes, the algorithm reaches a *k optimal* state when no further positive gain offers are proposed (Pearce and Tambe 2007). In this solution the exploration of the search space by the algorithm is dependent on the size of the allowed coalitions k. Since the quality of the state held by the agents is monotonically improving, it can be considered *anytime*. However, it does not offer classical *anytime* local search in which the current state of the system can deteriorate in order to achieve exploration while the reported solution monotonically improves.

Several local search algorithms were proposed for Distributed Constraints *Satisfaction* problems. Most of them apply a synchronous framework which in each synchronous step of the algorithm agents propagate their assignments to all their neighbors in the constraint graph, collect all the assignments of their neighbors, and decide whether to change their assignment (Yokoo and Hirayama 2000; Zhang et al. 2005).

Such local search algorithms are applicable for DisCOPs. However, they cannot report the best solution they traverse (i.e., they are not *anytime* algorithms (Zilberstein 1996)).

*The research was supported by the Lynn and William Frankel Center for Computer Science, and by the Paul Ivanier Center for Robotics.

The reason is that in contrast to a satisfaction scenario where in the global optimal state all agents are also in a local (e.g., private) optimal state, in a distributed optimization scenario agents might have a local (private) assignment to their variables which is not optimal but the system is in an optimal (global) state.

In (Zhang et al. 2005), distributed local search algorithms (DSA and DBA) were evaluated solving sensor network DisCOPs. Apparently these algorithms perform well on this application even without a pure *anytime* property. The algorithms were compared by evaluating the state held by agents at the end of the run. Such evaluation limits the chances of local search algorithms which implement an exploring heuristic to be successful.

In order to implement local *anytime* search algorithms which follow the same general structure of DSA and DBA for Distributed Optimization problems, the global result of every synchronous iteration must be calculated and the best solution must be stored. A trivial solution would be to centralize in every iteration the costs calculated by all agents to a single agent which will inform the other agents each time a solution which improves the results on all previous solutions is obtained. However, this method has drawbacks both in the increase in the number of messages and in the violation of privacy caused from the need to inform a single agent (not necessarily a neighbor of all agents) of the quality of all other agents' states in each of the iterations of the algorithm.

The present paper proposes a general framework for enhancing local search algorithms which follow the general synchronous structure of DSA and DBA and solve DisCOPs with the *anytime* property. In the proposed framework the quality of each state is accumulated via a *Breadth First Search tree* (BFS-tree) structure. Agents receive the information about the quality of the recent states of the algorithm from their $children$ in the BFS-tree, calculate the resulting quality including their own contribution according to the goal function, and pass it to their parents. The *root agent* makes the final calculation of the cost for each state and propagates down the tree the index number of the most successful state. When the search is terminated, all agents hold the assignment of the best state according to the global goal function.

In order to produce the best state out of m steps, the algorithm must run $m + (2 * h)$ synchronous steps where h is the height of the tree used. Since the only requirement of the tree is to maintain a parent route from every agent to the root agent, the tree can be a BFS-tree and its height h is expected to be small. The proposed framework does not require agents to send any messages beside the messages sent by the original algorithm. The space requirements for each agent are $O(h)$. In terms of privacy, the proposed framework preserves a higher level of privacy than state-of-the-art algorithms for solving DisCOPs which use a pseudo-tree

DSA
1. $value \leftarrow$ ChooseRandomValue()
2. **while** (no termination condition is met)
3. send value to neighbors
4. collect neighbors' values
5. **if** (ReplacementDecision())
6. select and assign the next value

Figure 1: Standard DSA.

Local Search for DisCOPs

[1] The general design of local search algorithms for Distributed Constraint Problems is synchronous. In each step of the algorithm an agent sends its assignment to all its neighbors in the constraint network and receives the assignment of all its neighbors. For lack of space we only present one algorithm that applies to this general framework, the *Distributed Stochastic Algorithm* (DSA). It is presented following the recent version of (Zhang et al. 2005). [2]

The basic idea of the DSA algorithm is simple. After an initial step in which agents pick some value for their variable (random according to (Zhang et al. 2005)), agents perform a sequence of steps until some termination condition is met. In each step, an agent sends its value assignment to its neighbors in the constraints graph and receives the assignments of its neighbors. [3] After collecting the assignments of all its neighbors, an agent decides whether to keep its value assignment or to change it using a stochastic strategy (see (Zhang et al. 2005) for details on the possible strategies and the difference in the resulting performance). A sketch of DSA is presented in Figure 1.

Anytime framework

Local search algorithms combine exploration and exploitation properties in order to converge to local minimas, and escape from them in order to explore other parts of the search space. When a centralized constraint optimization local search algorithm is performed, the quality of the states of the algorithm are completely known and therefore there is no difficulty in holding the best state which was explored. In a distributed constraint optimization problem, agents are only aware of their own private state (the violated constraints which they are involved in and their costs) and, thus, a state which can seem to have high quality to a single agent might be of low global quality and vise versa.

We propose a framework that will enhance DisCOP local search algorithms with the *anytime* property. In the proposed

[1] A detailed description of Distributed Constraint Optimization was left out for lack of space. A detailed description can be found in (Gershman, Meisels, and Zivan 2006). We assume that agents are aware only of their own topology (constraints that they are involved in personally and privately hold).

[2] In our description we consider an improvement a decrease in the number of violated constraints (as in Max-CSPs).

[3] In this paper we follow the general definition of a DisCOP and a DisCSP which does not include a synchronization mechanism. If such a mechanism exists, agents in DSA can send value messages only in steps in which they change their assignments.

DSA_DisCOP

1. $height \leftarrow$ height in the BFS- tree
2. $dist \leftarrow$ distance from root
3. $best \leftarrow null$
4. $best\_index \leftarrow null$
5. $current\_step \leftarrow 0$
6. **if** (root)
7. $\quad best\_cost \leftarrow \infty$
8. $value\_current \leftarrow$ ChooseRandomValue()
9. **while** $(current\_step < (m + dist + height))$
10. $\quad$ send value and $cost\_i$ to parent
11. $\quad$ send value to non tree neighbors
12. $\quad$ send value and $best\_index$ to children
13. $\quad$ collect neighbors' values
14. $\quad cost\_i \leftarrow$ CalculateStepCost($current\_step - height$)
15. $\quad$ **if**(root)
16. $\quad\quad$ **if**($cost\_i < best\_cost$)
17. $\quad\quad\quad best\_cost \leftarrow cost\_i$
18. $\quad\quad\quad best \leftarrow value\_i$
19. $\quad\quad\quad best\_index \leftarrow i$
20. $\quad$ **if** (message from parent includes a new $best\_index\ j$)
21. $\quad\quad best \leftarrow value\_j$
22. $\quad\quad best\_index \leftarrow j$
23. $\quad$ **if** (ReplacementDecision())
24. $\quad\quad$ select and assign the next value
25. $\quad$ delete $value\_(current\_step - (2 * dist))$
26. $\quad$ delete cost of step $(current\_step - height)$
27. $\quad current\_step++$
28. **for** (1 to $dist + height$)
29. $\quad$ receive message from parent
30. $\quad$ **if** (message from parent includes a new $best\_index\ j$)
31. $\quad\quad best \leftarrow value\_j$
32. $\quad\quad best\_index \leftarrow j$
33. $\quad$ send $best\_index$ to children

Figure 2: DSA in the ALS_DisCOP framework.

Anytime Local Search framework, ALS_DisCOP, a tree is used as in $ADOPT$ (Modi et al. 2005) and $DPOP$ (Petcu and Faltings 2005). In contrast to $ADOPT$ and $DPOP$ that require the use of a pseudo-tree, the only requirement in ALS_DisCOP is that every agent has a parent route to the root agent. Thus, a *Breadth First Search* (BFS) tree on the constraints graph can be used. The BFS-tree structure is used in order to accumulate the cost of agents' states in the different steps during the execution of the algorithm. Each agent calculates the cost of the sub-tree it is a root of in the BFS-tree and passes it to its parent. The root agent calculates the complete cost of each state and if it is better than the best state found so far, propagates its index to the rest of the agents. Each agent A_i is required to hold its assignments in the last $2*dist_i$ steps where $dist_i$ is the length of the route of parents in the BFS-tree from A_i to the root agent and is bounded by the height of the BFS-tree (h).

In each step of the algorithm an agent collects from its children in the BFS-tree the calculation of the cost of the sub-tree of which they are the root. When it receives the costs for a step j from all its children, it adds its own cost for the state in step j and sends the result to its parent. When the root agent receives the calculations of the cost of step i from all its children, it calculates the global state cost. If it is better than the best state found so far, in the next step it will inform all its children that the state in step j is the new best state. Agents which are informed of the new best step store their assignment in that step as the best assignment and pass the information about the best index to their children in the next step. After every synchronous step the agents can delete the information stored about any of the steps which were not the best and are not of the last $2 * dist$ steps. When the algorithm is terminated, the agents must perform another h steps (again, h is the height of the BFS-tree) in which they do not replace their assignment to make sure that all the agents are aware of the same index of the best step.

The code for DSA in the ALS_DisCOP framework is presented in Figure 2 [4]. The structure of the framework is homogeneous for all algorithms with a distributed synchronous local search general structure (such as DSA and DBA). It is interleaved in the algorithm execution as follows:

1. In the initialization phase, besides choosing a random value for the variable, agents initialize the parameters which are used by the framework. The root initializes an extra integer variable to hold the cost of the best step (lines 1-7 in Figure 2).
2. In order to get the best out of m steps of the algorithm, $m + h$ steps are performed (notice that for each agent the sum of $height$ and $dist$ is equal to h which is the height of the global BFS-tree). This is required so all the information needed for the root agent to calculate the cost of the m steps will reach it (line 9 in Figure 2).
3. After values are exchanged, each agent calculates the cost of the state according to its height. An agent with height h_i calculates the cost of the state in which its sub-tree was in h_i steps ago. The root agent checks if the cost it calculated is smaller than the best cost found so far and if so saves its information. All other agents check if the best index received from their parent is new. If so they save the information (index and assignment) of the step with the corresponding index (lines 16-22 in Figure 2).
4. Before the step is over, the agent deletes the information that has become redundant. This includes the information on the cost which it passed to its parent on this step and the assignment of the step which its index should have been received on this step in case it was found to be better than previous steps by the root agent (lines 25,26 in Figure 2).
5. On the next step, the value message an agent sends to its parent will include the cost calculation it had performed in this step and the messages to its children will include the index of the best step it knows of.
6. When the termination condition of the algorithm is met, the agents perform additional h steps in which only the best index is propagated down the tree. This way, if the last step cost calculated by the root agent is found to be best, its propagation to all agents is completed. Furthermore, by performing these steps, the possibility that different agents hold assignments of their best steps which

[4] We assume the existence of a BFS tree when the algorithm begins.

belong to different steps is prevented (lines 28-33 in Figure 2).

Properties of ALS_DisCOP

ALS_DisCOP is a framework for implementing local search algorithms for DisCOPs. Regardless of the algorithm being used, the ALS_DisCOP framework offers properties which ensure the preservance of the algorithm's behavior.

Anytime property

The main goal of the ALS_DisCOP framework is to enhance a distributed local search algorithm with the *anytime* property (i.e. that the cost of the solution held by the algorithm at the end of the run would monotonically decrease if the algorithm is allowed to run for additional iterations (Zilberstein 1996)). In order to prove that ALS_DisCOP is an *anytime* framework for distributed local search algorithms, we first prove the following Lemma:

Lemma 1 *When the algorithm terminates, all the assignments of the best state held by all the agents in the system are assignments they held in the same step (in other words the $best\_index$ of all agents is equal).*

proof: The proof for this Lemma derives directly from the last part of the framework which includes h steps in which messages including only the $best\_index$ are passed. Since no new $best\_index$ is found by the root in these steps and h steps are enough for all the agents in the BFS-tree to receive a new $best\_index$, then even if a $best\_index$ was found in the last (standard) step of the algorithm its propagation is completed. □

Next, we prove that at the end of the run of $m + h$ iterations (synchronous steps of the algorithm in the framework), the index of the step with the best state which was held by the system in the m first steps of the algorithm is held by the root agent. To this end we present the following Lemma (the simple proof was left out for lack of space):

Lemma 2 *At the $i + h$ step, the root agent holds all the needed information for calculating the quality of state i.*

The *anytime* property of the ALS_DisCOP framework derives directly from the Lemmas 1 and 2. h steps after each step is performed, the root agent holds the information needed to evaluate its quality and can propagate its index to all other agents in case it is the best. Since we require that agents hold the value assignments of the last $2 * dist$ ($dist$ is the length in a rout of parents of an agent from the root which is at most h) steps they can get the update on the index of the best step before they delete the relevant assignment and hold it until they receive another update. Thus, after $m + h$ steps, the root agent holds the index of the step with the best state among the first m steps and according to Lemma 1, at the end of the algorithm run, all agents hold the value assignment of the step which was found by the root to be the best. If the algorithm is run for k more steps (the termination condition is met after $m + k + h$ steps), in the k steps performed after the first m steps, either a better solution is found or the same solution which was found best in the first m steps is reported. □

Performance analysis

Distributed algorithms are commonly measured in terms of time for completion and network load (Lynch 1997). When considering a local search algorithm for DisCOPs, since it is not complete and the agents cannot determine that they are in an optimal state, there is no point in measuring time. We note that in the proposed framework, in order to get the best state among m steps the algorithm needs to run for $m + (2 * h)$ steps. However, the tree which is used by ALS_DisCOP has different requirements than the pseudo-trees which are used by complete algorithms (like $ADOPT$ and $DPOP$). In a pseudo-tree which is used in complete algorithms, constrained agents must be on the same parent route (in every binary constraint, one of the constrained agents is an ancestor of the other). This requirement makes the pseudo-tree less effective when the constraint graph is dense. In contrast, the only requirement in ALS_DisCOP is that every agent has a parent route to the root agent. Thus, a *Breadth First Search* (BFS) tree can be used. Since a BFS-tree includes the shortest route from each node to the root, the height of the resulting tree (especially when the constraint graph is dense) is expected to be small.

In terms of network load, the ALS_DisCOP framework does not require any additional messages. In standard local search algorithms (such as DSA and DBA), agents at each step send at least one message to each of their neighbors. The ALS_DisCOP framework does not require more. The additional information that agents add to messages is constant (a single cost of a step or a best index).

In terms of space, an agent i is required to hold the value assignments of the last $2 * dist$ steps (again $dist$ is the distance in tree arcs of an agent from the root) and to hold the cost of the last $height$ states (where $height$ is the height of the agent in the BFS-tree). This results in an $O(h)$ additional space requirement (linear in the worst case) for each agent.

Privacy of ALS_DisCOP

When a complete algorithm is being performed, agents must be able to announce that parts of the search space were already scanned and do not include a better solution than the best solution which was already found by the algorithm. This is done by declaring lower bound costs on partial assignments ($Nogoods$). While this report of costs of partial assignments is necessary for the algorithm's completeness, it is also a drawback when it comes to privacy. In a naive algorithm like synchronous $B\&B$, agents are required to reveal their assignments to non-neighboring agents. In the case of algorithms which exploit the structure of the constraint network, such as $ADOPT$ and $DPOP$, agents report costs with respect to their contexts, which means, for example in $ADOPT$, that a parent of a leaf agent is informed not only of the lowest cost of any of the leaf's possible value assignments but of the assignments of agents with which it is constrained and caused this cost.

In a local search algorithm, partial assignments can be revisited and therefore a cost of a state must be reported, but the context which caused this cost can remain concealed.

The trade-off that is commonly considered between complete search and local search is time versus completeness. It turns out that there is another trade-off to consider, which is the privacy loss required by a complete algorithm and is not required in a local search algorithm.

In addition to the information which the local search algorithm requires agents to exchange between them, ALS_DisCOP requires that each agent will pass the cost of a state in the sub-tree of which it is the root. As in other algorithms which use a tree (as $ADOPT$ and $DPOP$), the main problem in ALS_DisCOP is with the information passed by leafs in the tree to their parents (Greenstadt, Grosz, and Smith 2007).

When a non-leaf agent A_j passes the cost of its sub-tree to its parent, the parent does not know how many children A_j has and the contribution of each of these agents to the reported cost. On the other hand, when a leaf agent reports a cost, its parent knows that it is the cost of a single agent (the leaf itself). However, agents are not aware of the system's topology except for their own neighbors. So in fact, even though the parent of a leaf receives its cost in every step of the algorithm, the parent does not know how many neighbors of its leaf child has in the constraint network and which constraints were violated. Thus, the privacy violation is minor.

Experimental Evaluation

In order to emphasize the impact of the ALS_DisCOP framework on distributed local search, a set of experiments that demonstrate the effect of the proposed framework on the DSA algorithm is presented.

The experiments were performed on realistic *Meeting Scheduling Problems* $MSPs$ (Gent and Walsh 1999; Meisels and Lavee 2004; Modi and Veloso 2004). The agents' goal in a MSP is to schedule meetings among them. Each meeting which is scheduled is assigned a utility according to the participating agents' preferences on the time-slot it was scheduled in. The experiments setup included $MSPs$ with 20 agents and 20 meetings. There were 12 time slots available (3 days, 4 time slots in each day). In each meeting there are 5 participating agents. 50 different problems were generated in which agents were assigned randomly to meetings. [5]

Four versions of DSA were implemented. In each version the probability to change an already set time-slot of a meeting was different. This parameter (*p_explore*) determined the level of the exploration property of the algorithm. It enables agents to perform steps which deteriorate their current utility (in contrast to previous versions of DSA (Zhang et al. 2005)).

Each of the 50 problems was solved by each of the algorithm's versions 20 times. In each time the algorithm performed 1000 steps. For each run of the algorithm we calculated the sum of all meeting utilities. The maximal utility

[5]Our paper which describes in details the representation of an MSP problem as a DisCOP was submitted to the DCR-2008 workshop. We will add the reference in the final version of the paper.

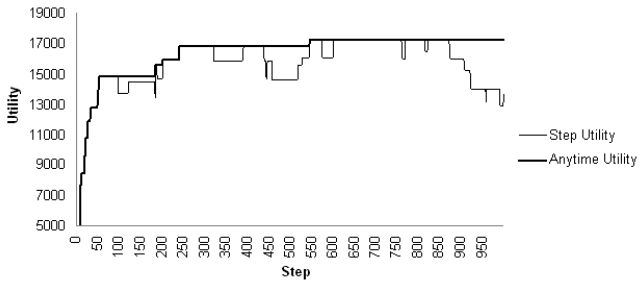

Figure 3: Step utility vs. anytime utility.

of a schedule (if all agents got all their meeting scheduled according to their preferences) is 25000. The results presented in Table 1 represent the average sum of utilities of all 1000 runs for each of the versions of the algorithm with and without using the anytime framework.

Notice that the meeting scheduling problem is a maximization problem (it is concerned with maximizing utilities and not with minimizing costs). Although our presentation of the ALS_DisCOP framework was with accordance to the classic minimization DisCOP (Yokoo and Hirayama 2000; Modi et al. 2005; Petcu and Faltings 2005; Gershman, Meisels, and Zivan 2006), the framework is compatible (with very small changes) for maximization problems as well.

p_explore	state utility	anytime utility
0	18387	18387
0.0001	18381	18523
0.0005	18196	18851
0.001	17672	189161
0.005	13170	17642

Table 1: DSA with different level of exploration solving MSPs

As expected, there is a correlation in the results presented in Table 1 between the quality of the state at the end of the search and the exploration probability factor. The higher the probability for exploration, the lower the quality of the final state. However, when the anytime framework is used, the algorithm reports the state with the highest quality which was found. In this case, the quality of the reported state increases with the increment of the exploration probability up to some point and then drops (this phase transition which occurs for high level of exploration in DSA was reported in (Zhang et al. 2005)). The difference between the quality of the best (anytime) reported state and the quality of the final state monotonically increases in correlation with the increment of the exploration factor.

In order to demonstrate the behavior of the algorithm which leads to the results in Table 1 we present the change in the utility in a single run of the algorithm. Figure 3

presents the difference between the state utility at each step and the anytime utility at each step of a single run of the DSA algorithm with the most successful exploration probability (*p_explore* = 0.001). It is clear that the exploration of the algorithm enables an improvement in the resulting utility. Furthermore, the anytime property eliminates the dependency on the quality of the last state performed.

Conclusions

DisCOPs are hard optimization problems which require exhaustive search. Therefore, complete search algorithms are limited for solving relatively small problems.

Distributed local search algorithms were proposed for Distributed CSPs and were applied for DisCOPs (Zhang et al. 2005). However, these algorithms failed to report the best state traversed by the algorithm.

In order to enhance local search algorithms for DisCOPs with the *anytime* property, a general framework for performing distributed local search algorithms in DisCOPs was proposed in this paper. In the proposed framework, agents use a BFS-tree structure in order to accumulate the costs of a state of the system to the root agent which compares the cost of each state with the cost of the best state found so far and propagates the index of a new best state, once it is found, to all other agents. At the end of the run, the agents hold the best state which was traversed by the algorithm.

Apart from a small number of idle steps at the end of the run of the algorithm (two times the height of the BFS-tree), the framework does not require any additional slowdown in the performance of the algorithm. In contrast to complete algorithms which use a pseudo-tree, the tree used in ALS_DisCOP can be a *Breadth First Search* (BFS) tree. Thus, the height of the tree is expected to be small. In terms of network load, the only messages used in the ALS_DisCOP framework are the algorithm's messages (i.e., no additional messages are required by the framework). Agents are required to use small (linear in the worst case) additional space. In terms of privacy, ALS_DisCOP preserves a higher level of privacy than complete DisCOP algorithms which use a pseudo-tree, since an agent is not required to reveal to its neighbor with which of the other agents it is constrained (who are its other neighbors in the constraints graph).

Our experimental study demonstrates the potential of distributed anytime local search algorithms which implement an exploration heuristic. We hope that this paper will encourage studies of distributed local search algorithms for DisCOPs which will combine stochastic and systematic methods for exploration and exploit the special properties of a distributed optimization problem.

Acknowledgment: I thank Alon Grubshtein for producing the experiments presented in this paper.

References

Gent, I., and Walsh, T. 1999. Csplib: a benchmark library for constraints. Technical report, Technical report APES-09-1999. Available from http://csplib.cs.strath.ac.uk/. A shorter version appears in the Proceedings of the 5th International Conference on Principles and Practices of Constraint Programming (CP-99).

Gershman, A.; Meisels, A.; and Zivan, R. 2006. Asynchronous forward-bounding for distributed constraints optimization. In *Proc. ECAI-06*, 103–107.

Greenstadt, R.; Grosz, B. J.; and Smith, M. D. 2007. Ssdpop: Improving the privacy of dcop with secret sharing, distributed constraint reasoning workshop (dcr), providence, rhode island, september 2007. In *Distributed Constraint Reasoning Workshop (DCR), CP-07*.

Lynch, N. A. 1997. *Distributed Algorithms*. Morgan Kaufmann Series.

Maheswaran, R. T.; Pearce, J. P.; and Tambe, M. 2004. Distributed algorithms for dcop: A graphical-game-based approach. In *Proc. Parallel and Distributed Computing Systems PDCS)*, 432–439.

Mailer, R., and Lesser, V. 2004. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. AAMAS-2004*, 438–445.

Meisels, A., and Lavee, O. 2004. Using additional information in discsp search. In *Proc. 5th workshop on distributed constraints reasoning, DCR-04*.

Modi, J., and Veloso, M. 2004. Multiagent meeting scheduling with rescheduling. In *Proc. of the Fifth Workshop on Distributed Constraint Reasoning (DCR), CP 2004*.

Modi, P. J.; Shen, W.; Tambe, M.; and Yokoo, M. 2005. Adopt: asynchronous distributed constraints optimization with quality guarantees. *Artificial Intelligence* 161:1-2:149–180.

Pearce, J. P., and Tambe, M. 2007. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*.

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *IJCAI*, 266–271.

Schaerf, A. 1999. Local search techniques for large high-school timetabling problems. *IEEE Transactions on Systems, Man, and Cybernetics— Part A: Systems and Humans* 29(4):368–377.

Yokoo, M., and Hirayama, K. 2000. *Distributed Constraint Satisfaction Problems*. Springer Verlag.

Zhang, W.; Xing, Z.; Wang, G.; and Wittenburg, L. 2005. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence* 161:1-2:55–88.

Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI Magazine* 17(3):73–83.