Discrepancy-Based Approach for Solving Distributed Constraint Optimization Problems

William Yeoh[†] Roie Zivan[‡] Sven Koenig[†]

[†]Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
{wyeoh,skoenig}@usc.edu

[‡]Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
zivanr@cs.cmu.edu

Abstract. The distributed constraint optimization (DCOP) model is a popular model for formulating and solving agent-coordination problems. The cost-minimal solution that most DCOP algorithms look for might not be desirable in dynamically changing environments where agents are committed to an existing solution. In such situations, it might be more desirable to find either (a) a cost-minimal solution where at most k agents need to break their commitments, where k is a user-defined constant; or (b) any solution within a user-defined error bound that minimizes the number of agents that need to break their commitments. Limited discrepancy search searches for solutions in the order of increasing number of discrepancies, i.e. number of agents that need to break their commitments, and is thus ideally suited for finding the two types of solutions mentioned above. We describe how one can transform a DCOP algorithm that employs depth-first branch-and-bound to employ limited discrepancy search and, as an example, transform BnB-ADOPT to Limited Discrepancy Search ADOPT.

Key words: LDS; Limited Discrepancy Search; BnB-ADOPT; DCOP; Distributed Constraint Optimization; Distributed Search Algorithms

1 Introduction

A distributed constraint optimization (DCOP) problem is a problem where several agents coordinate to take on values such that the objective of the DCOP problem is achieved. Typically, optimizing the objective of the DCOP problem means finding a complete solution (= an assignment of values to all agents) that is cost-minimal (= the minimal sum of the constraint costs). DCOP problems are a popular way of formulating and solving agent-coordination problems, including scheduling meetings [8], coordinating traffic lights [7] and allocating

targets to sensors [9]. As a result, researchers have developed a variety of DCOP algorithms [14, 9, 11, 3], including BnB-ADOPT [12].

Although finding a cost-minimal complete solution is a noble goal, a cost-minimal complete solution might not always be desirable. For example, assume that the problem at hand is to schedule meetings for employees in a company. Assume that, after a cost-minimal complete solution was found, there is a change in the constraints regarding one of the meeting. For example, the current time slot for the meeting is no longer available due to a fire drill. As a result, a new solution needs to be found. The cost-minimal complete solution to this new DCOP problem might not be desirable if it is significantly different from the previous solution especially if a large number of meetings need to be rescheduled. The reason is that the employees might already have committed to their previous schedules by making other arrangements with their schedules as constraints. In such a situation, it might be more desirable to find either (a) a cost-minimal complete solution where at most k meetings need to be rescheduled, where k is a user-defined constant, or (b) any complete solution whose cost is within a user-defined error bound that minimizes the number of meetings that need to be rescheduled.

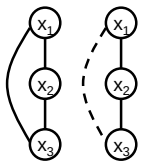
To obtain such solutions, we propose that DCOP algorithms employ limited discrepancy search (LDS) [5] to search through the space of possible solutions. LDS searches for complete solutions in an increasing order of the number of discrepancies, i.e. number of agents that have to break their commitments, and is thus well suited for finding the two types of solutions mentioned above. LDS has been successfully extended in [2] to solve Hierarchical DCOP problems, where hierarchical subproblems need to be solved in sequence. However, this DCOP algorithm cannot solve more general DCOP problems and, to the best of our knowledge, there does not exist a DCOP algorithm that finds either of the two types of solutions mentioned above. We describe how one can transform a DCOP algorithm that employs depth-first branch-and-bound (DFBnB) search to employ LDS by transforming BnB-ADOPT to Limited Discrepancy Search ADOPT (LDS-ADOPT).

2 Background

We now define DCOP problems and describe existing DCOP algorithms.

2.1 DCOP Problems

A DCOP problem is defined by a finite set of agents $X = \{x_1, x_2, ..., x_n\}$; a set of finite domains $D = \{Dom(x_1), Dom(x_2), ..., Dom(x_n)\}$, where the domain $Dom(x_i)$ is the set of possible values for agent $x_i \in X$; and a set of binary constraints $F = \{f_1, f_2, ..., f_m\}$, where each constraint $f_i : Dom(x_{i_1}) \times Dom(x_{i_2}) \rightarrow \mathbb{R}^+ \cup \infty$ specifies its non-negative cost as a function of the values of distinct agents $x_{i_1} \in X$ and $x_{i_2} \in X$ that share the constraint.



(a) (b)

for $i < j$

x_i	x_j	Cost
0	0	5
0	1	8
1	0	20
1	1	3

(c)

Fig. 1. Example DCOP Problem

Each agent is responsible for assigning itself (= taking on) values from its domain. The agents coordinate these value assignments via messages that they exchange with other agents. A complete solution is an agent-value assignment for all agents, while a partial solution is an agent-value assignment for a subset of agents. The cost of a complete solution is the sum of the constraint costs of all constraints, while the cost of a partial solution is the sum of the constraint costs of all constraints shared by agents with known values in the partial solution. Solving a DCOP problem optimally means to achieve its objective.

It is common to visualize a DCOP problem as a *constraint graph* where the vertices are the agents and the edges are the constraints. Most DCOP algorithms operate on a *pseudo-tree*, which is a spanning tree of the constraint graph with the property that no two vertices in different subtrees are connected by an edge in the constraint graph. Figure 1(a) shows the constraint graph of an example DCOP problem with three agents that can each take on the values zero or one, Figure 1(b) shows one possible pseudo-tree (the dotted line is part of the constraint graph but not the pseudo-tree), and Figure 1(c) shows the constraint costs. To be precise, the pseudo-tree is actually a pseudo-chain since the constraint graph is fully connected.

2.2 DCOP Algorithms

There are two classes of DCOP algorithms: *Complete* DCOP algorithms such as SynchBB [6], ADOPT [9], DPOP [11], NCBB [1], AFB [3] and BnB-ADOPT, find globally optimal solutions; and *incomplete* DCOP algorithms [14, 10] find locally optimal solutions. We use the term DCOP algorithms to refer to complete DCOP algorithms and the terms optimal/cost-minimal to refer to globally optimal/cost-minimal since we are interested in finding globally optimal solutions.

DPOP is a DCOP *inference algorithm* that employs dynamic programming to propagate the constraint costs of all agents to the root agent, which then finds the cost-minimal complete solution. SynchBB, ADOPT, NCBB, AFB and BnB-ADOPT are DCOP *search algorithms* that employ different search strategies to search through the space of solutions to find a cost-minimal solution. ADOPT employs *best-first* search while SynchBB, NCBB, AFB and BnB-ADOPT employ *depth-first branch-and-bound* (DFBnB) search. Since DCOP algorithms that employ DFBnB are often faster than DCOP algorithms that employ best-first

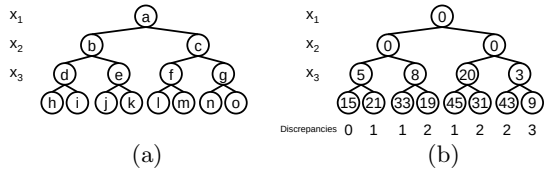


Fig. 2. Search Tree

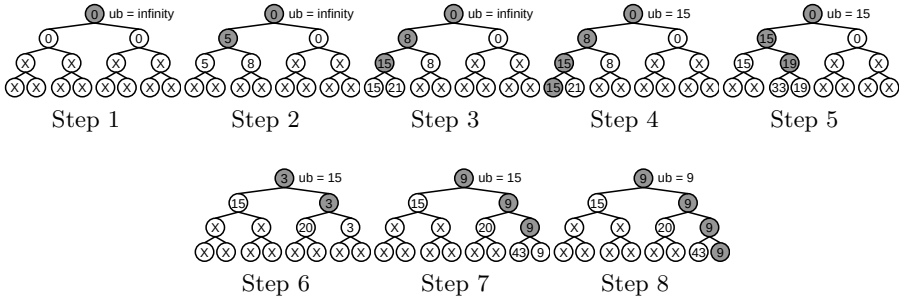


Fig. 3. Execution Trace of Depth-First Branch-and-Bound Based DCOP Algorithms

search [3, 12], we transform DCOP algorithms that employ DFBnB to employ LDS.

3 Discrepancy-Based DCOP Algorithms

In this section, we first describe the limited discrepancy search (LDS) strategy. Then, we describe the general idea behind DFBnB-based DCOP algorithms, before proceeding to describe how one can transform these DCOP algorithms to discrepancy-based DCOP algorithms.

3.1 Limited Discrepancy Search

To illustrate the search process of LDS, we use the search trees and terminology of A* [4]. Figure 2 shows the search tree for our example DCOP problem.

Each level of a search tree corresponds to an agent. For our example DCOP problem, the level of depth 1 corresponds to agent x_1. A left branch that enters a level means that the corresponding agent takes on the value 0. A right branch means that the corresponding agent takes on the value 1. A node in the search tree thus corresponds to a solution. For our example DCOP problem, the partial solution of node e is $(x_1 = 0, x_2 = 1)$. The numbers inside the nodes in Figure 2(b) are the costs of the partial solutions associated with the nodes. These numbers correspond to the f-values of an A* search, if we assume for simplicity

that the zero heuristics are used. The letters inside the nodes in Figure 2(a) are identifiers that allow us to refer to the nodes easily.

Assume that every agent maintains a user-defined *default value*. When an agent takes on a non-default value, this action results in a *discrepancy*. The number of discrepancies in a solution then is the number of agents that take on non-default values in that solution. The number of discrepancies for the eight possible complete solutions of our example DCOP problem is shown under the search tree in Figure 2(b) if we assume that the default value is zero for all agents. LDS operates as follows: First, it searches for the complete solution with zero discrepancies ($x_1 = 0, x_2 = 0, x_3 = 0$). Then, it searches for complete solutions with one discrepancy by allowing one agent at a time to take on a non-default value. The agents do so sequentially from the root agent to the leaf agent of the pseudo-tree. Thus, the complete solutions with one discrepancy are found in the following order: ($x_1 = 1, x_2 = 0, x_3 = 0$), ($x_1 = 0, x_2 = 1, x_3 = 0$), ($x_1 = 0, x_2 = 0, x_3 = 1$). It continues the search with increasing numbers of discrepancies until a cost-minimal complete solution has been found or all complete solutions with the user-defined maximum number of discrepancies have been found.

3.2 Depth-First Branch-and-Bound-Based DCOP Algorithms

We now describe DFBnB and the general idea behind DCOP algorithms that employ this search strategy.

Depth-First Branch-and-Bound

To illustrate the search process of DFBnB-based DCOP algorithms, we use the search trees shown in Figure 3. For simplicity, we assume that the agents operate sequentially and that information is propagated instantaneously, which is sufficient to describe the general search strategy. The nodes that are being expanded and their ancestors are shaded grey. The agents of each level maintain lower bounds for the grey nodes and their children, shown as the numbers in the nodes. Lower bounds that are not maintained are shown as crosses in the nodes. The agents initialize the lower bounds of the nodes that they maintain with the f-values and then set them to the minimum of the lower bounds of the children of the nodes. Therefore the lower bound of the root node is an underestimate of the cost of a cost-minimal solution. The agents of each level also maintain upper bounds for grey nodes and their children. However, we only show the upper bound of the root node, shown as ub, as that is the only upper bound that is essential for our discussion. The upper bound of the root node is always set to the smallest cost of all complete solutions found so far. Therefore, the upper bound of the root node is an overestimate of a cost of the cost-minimal solution.

DFBnB expands the children of a node in the order of increasing f-values and prunes those nodes whose f-values are no smaller than the upper bound of the root node. Thus, it expands nodes in the order a, b, d, h, i^*, e, k^*, j^*, c, g and o, where the nodes with asterisks are pruned. DFBnB terminates when the

lower bound of the root node equals the upper bound of the root node, indicating that a cost-minimal solution has been found.

Outline of Algorithms

The description so far is very simplistic as it assumes that agents propagate information instantaneously. In reality, agents do not have complete knowledge of the values taken on by their ancestor agents (= agents higher up in the pseudo-tree) at all times. These sets of agent-value pairs are called the *contexts* of the agents. Generally, an agent sends *VALUE* messages to its descendant agents that it is connected to in the constraint graph with the value that it currently takes on.[1] Upon receiving these messages, the receiving agents update their contexts to reflect the value of the sending agent contained in the messages. Neither do agents have complete knowledge of the lower and upper bounds of their children agents at all times. Generally, an agent sends *COST* messages to its parent agent with its lower and upper bounds.[2] Upon receiving these messages, the parent agent updates its own lower and upper bounds accordingly. For example, the lower bound of a node in Figure 3 is always set to the minimum of the lower bounds of its children.

Generally, the agents operate in the following fashion. They first take on an initial value and send VALUE messages. They then operate the following loop: They receive and process incoming messages. If they received a VALUE message, they update their contexts. If they received a COST message, they update their lower and upper bounds. After that, they take on a new value if they can prune the subtree rooted at the node that correspond to their current value. Subsequently, they send VALUE and COST messages, if necessary, and wait for new incoming messages.

It is important to note that there are differences in both the content of the messages and the communication links between agents for the different algorithms. Agents in AFB send their entire contexts together with their current values in VALUE messages, while agents in BnB-ADOPT only send their current values in VALUE messages. Agents in NCBB send COST messages to all ancestor agents that they are connected to in the constraint graph, while agents in BnB-ADOPT only send COST messages to their parent agents.

3.3 LDS-ADOPT

We now describe how one can transform a DCOP algorithm that employs DF-BnB to employ LDS. As an example, we transform BnB-ADOPT to Limited Discrepancy Search ADOPT (LDS-ADOPT).

Notation

[1] These messages are called *SEARCH* messages in NCBB and *CPA_MSG* messages in AFB.

[2] These messages are called *FB_ESTIMATE* messages in AFB.

We use the following notation from BnB-ADOPT to describe LDS-ADOPT: $ValInit(x) \in Dom(x)$ is the default value of agent $x \in X$. $C(x) \subseteq X$ is the set of children agents of agent x in the pseudo-tree, and $CD(x) \subseteq X$ is the set of its descendant agents that it is connected to in the constraint graph. $pa(x) \subseteq X$ is the parent agent of agent $x \in X$, $A(x) \subseteq X$ is the set of its ancestor agents (including its parent agent), $SCA(x) \subseteq A(x)$ is the set of its ancestor agents (including its parent agent) that it or one of its descendant agents is connected to in the constraint graph, and $CA(x) \subseteq SCA(x)$ is the set of its ancestor agents (including its parent agent) that it is connected to in the constraint graph.

Brief Description of Transformation

There are two major differences between a DCOP algorithm that employs DFBnB and a DCOP algorithm that employs LDS. The first difference is the notion of *tokens* and *anchor agents*. Tokens are used to restrict the number of discrepancies in the solutions found. An agent in an LDS-based DCOP algorithm can take on a non-default value only when it has a token. An agent is an anchor agent if it has a token and there is at least one token that is passed around among its descendant agents. The second difference is the notion of *pruning*. An agent in a DFBnB based DCOP algorithm is able to prune the subtree rooted at a node that corresponds to its value when no solution in that subtree can have a smaller cost than the best solution found so far. On the other hand, an agent in an LDS based DCOP algorithm can prune the subtree rooted at a node that corresponds to its value when no complete solution in that subtree can have a smaller cost than the best complete solution found so far, given the constraint that the maximum number of non-default values its descendant agents take on is at most the number of tokens that they have. Note that, when an agent in an LDS-based DCOP algorithm prunes a subtree, there might exist a complete solution in that subtree whose cost is smaller than the best complete solution found so far, but it must have a larger number of descendant agents take on non-default values than the number of tokens that it has.

Operation of LDS-ADOPT

LDS-ADOPT transforms the constraint graph to a pseudo-chain in a pre-processing step. Then, it performs the following operations.

- **Zero discrepancies:** Initially, every agent has *zero* tokens and thus takes on its default value. It updates its bounds, checks to see if it can prune its current branch of the search tree and sends its updated bounds to its parent agent. Once the bounds have been propagated up to the root agent, indicating that the current branch of the root agent can be pruned, the complete solution with zero discrepancies has been found.
- **One discrepancy:** The root agent increases the number of its tokens by one. Thus, the root agent now has *one* token, allowing it to take on a non-default value. The root agent takes on a non-default value, prunes the branch corresponding to its current value once the bounds have been propagated up

to it and takes on a new non-default value. It repeats this process for all values in its domain. Finally, the root agent takes on its default value and passes the token down to its child agent. The child agent repeats the process until the token is passed to the leaf agent. After the leaf agent has taken on all values in its domain, all complete solutions with one discrepancy have been found. Then, the leaf agent takes on its default value and the token is propagated back to the root agent.

- **Two discrepancies:** The root agent increases the number of its tokens by one. Thus, the root agent now has *two* tokens. It declares itself to be an anchor agent, passes *one* token down to its child agent and takes on a non-default value. The child agent repeats the process described as above for one discrepancy. Then, all complete solutions with two discrepancies where the root agent takes on its current value have been found. Thus, the root agent takes on a new non-default value and repeats the process. Once the root agent has taken on all non-default values in its domain, it takes on its default value, declares itself to no longer be an anchor agent and passes both tokens down to its child agent. The child agent repeats the process until both tokens have been passed down to the leaf agent. At this point, all complete solutions with two discrepancies have been found. The leaf agent takes on its default value and both tokens are propagated back to the root agent.
- **Three to $|X|$ discrepancies:** The root agent increases the number of its tokens by one, and recursively repeats the above process until it has introduced $|X|$ tokens to the system. LDS-ADOPT terminates when a cost-minimal complete solution has been found or when all complete solutions with the user-defined maximum number of discrepancies have been found.

4 Experimental Evaluation

We use the same experimental setup as [8, 12] and conduct experiments with two DCOP problem types, namely *graph coloring problems* with 10 vertices/agents, density 2 and domain cardinality 3; and *meeting scheduling problems* with 10 meetings/agents and domain cardinality 3. We average the experimental results over 50 DCOP problem instances. The objective is to find a cost-minimal complete solution under the constraint that at most k agents can take on non-default values. The default value for every agent is 0. Since, to the best of our knowledge, no other DCOP algorithms exists for this objective, we compare LDS-ADOPT to a DCOP algorithm that is similar to SynchBB, but performs limited discrepancy search instead of depth-first branch-and-bound search. We call this DCOP algorithm SynchLDS. It assigns values to agents sequentially from the root agent to the leaf agent in the pseudo-chain and backtracks after it has assigned a value to the leaf agent. It uses LDS to determine which agent to backtrack to. We measure the runtimes of both DCOP algorithms in time slices called cycles [9], where smaller numbers of cycles indicate smaller runtimes. During each cycle, all agents receive and process all their incoming messages and send all outgoing messages. A new cycle starts immediately after the last agent sends its outgoing messages.

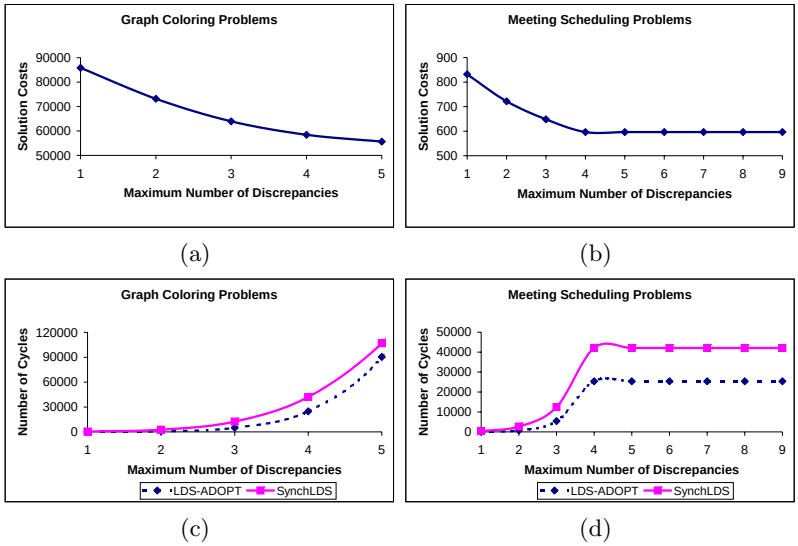


Fig. 4. Experimental Results

Figures 4(a) and 4(b) plot the costs of the complete solutions found by LDS-ADOPT and SynchLDS against the maximum number of agents that can take on non-default values (= maximum number of discrepancies). Both DCOP algorithms find complete solutions with lower costs as the maximum number of discrepancies increases. Figures 4(c) and 4(d) compare the runtimes of LDS-ADOPT and SynchLDS. LDS-ADOPT is faster than SynchLDS for both DCOP problem types. The speedup is due to the asynchronous and concurrent execution of agents in LDS-ADOPT. Agents in SynchLDS operate synchronously and sequentially and are thus often idle while waiting for activation messages from other agents. Paired Student's t-tests determine all results to be statistically significant ($p < 0.05$). The runtimes of both DCOP algorithms increase as the maximum number of discrepancies increases. There are two phases in the runtime of both DCOP algorithms: When the maximum number of discrepancies is small, their runtime grows exponentially, and when the maximum number of discrepancies is sufficiently large, their runtime stays constant. For example, the phase transition is at four discrepancies in Figure 4(d). This behavior is due to the DCOP algorithms finding a cost-minimal complete solution at the phase transition. Thus, larger numbers of allowed discrepancies do not affect their runtime. This result is consistent with the result in Figure 4(b) where the cost of the complete solution found remains constant when the maximum number of discrepancies is at least four.

5 Conclusions

Although DCOP problems are a popular way of formulating and solving agent-coordination problems, the typical objective of DCOP algorithms is to find a cost-minimal complete solution. However, a cost-minimal complete solution might not be desirable in dynamically changing environments where agents are committed to a previous complete solution. In such a situation, it might be more desirable to find either (a) a cost-minimal complete solution where at most k agents need to break their commitments, where k is a user-defined constant, or (b) any complete solution within a user-defined error bound that minimizes the number of agents that need to break their commitments.

To obtain such solutions, we proposed that DCOP algorithms employ limited discrepancy search to search through the space of possible solutions. Limited discrepancy search searches for complete solutions in an increasing order of discrepancies, i.e. number of agents that have to break their commitments by taking on a non-default value, and is thus well suited for finding the two types of solutions mentioned above. We illustrated how one can transform a DCOP algorithm that employs depth-first branch-and-bound search to employ limited discrepancy search by transforming BnB-ADOPT to Limited Discrepancy Search ADOPT (LDS-ADOPT).

Although we provided experimental results for the first type of solutions only, namely cost-minimal complete solutions where at most k agents need to break their commitments, the second type of solutions can be found with only a small change to the termination condition of LDS-ADOPT, namely by letting it terminate when it finds a complete solution whose cost is within the user-defined error bound instead of when it reaches user-defined the maximum number of discrepancies. Our experimental results showed that LDS-ADOPT is indeed faster than SynchLDS for graph coloring and meeting scheduling problems. In the future, we plan to extend LDS-ADOPT to operate on pseudo-trees instead of pseudo-chains. We also plan to further optimize LDS-ADOPT by incorporating the tradeoff mechanisms from [13] to find complete solutions whose costs are within a user-defined error bound as well as variable reordering methods [15] to increase its efficiency.

Acknowledgments

This material is based upon work supported by, or in part by, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-08-1-0468 and by NSF under contract 0413196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

References

1. A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimization. In *Proceedings of AAMAS*, pages 1427–1429, 2006.
2. J. Gaudreault, J. Frayret, and G. Pesant. Discrepancy-based method for hierarchical distributed optimization. In *Proceedings of ICTAI*, pages 29–31, 2007.
3. A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward-bounding for distributed constraints optimization. In *Proceedings of ECAI*, pages 103–107, 2006.
4. P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC4(2):100–107, 1968.
5. W. Harvey and M. Ginsberg. Limited discrepancy search. In *Proceedings of IJCAI*, pages 607–615, 1995.
6. K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of CP*, pages 222–236, 1997.
7. R. Junges and A. Bazzan. Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In *Proceedings of AAMAS*, pages 599–606, 2008.
8. R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *Proceedings of AAMAS*, pages 310–317, 2004.
9. P. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.
10. J. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of IJCAI*, pages 1446–1451, 2007.
11. A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of IJCAI*, pages 1413–1420, 2005.
12. W. Yeoh, A. Felner, and S. Koenig. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. In *Proceedings of AAMAS*, pages 591–598, 2008.
13. W. Yeoh, S. Koenig, and X. Sun. Trading off solution cost for smaller runtime in DCOP search algorithms (short paper). In *Proceedings of AAMAS*, pages 1445–1448, 2008.
14. W. Zhang, G. Wang, Z. Xing, and L. Wittenberg. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005.
15. R. Zivan and A. Meisels. Dynamic ordering for asynchronous backtracking on disCSPs. In *Proceedings of CP*, pages 32–46, 2005.