Resource constrained distributed constraint optimization using resource constraint free pseudo-tree

(Short Paper)

Toshihiro Matsui
Nagoya Institute of Technology
matsui.t@nitech.ac.jp

Marius Silaghi
Florida Institute of Technology
msilaghi@fit.edu

Katsutoshi Hirayama
Kobe University
hirayama@maritime.kobe-u.ac.jp

Makoto Yokoo
Kyusyu University
yokoo@is.kyushu-u.ac.jp

Hirohsi Matsuo
Nagoya Institute of Technology
matsuo@nitech.ac.jp

ABSTRACT

The Distributed Constraint Optimization Problem (DCOP) is a fundamental formalism for multi-agent cooperation. A dedicated framework called Resource Constrained DCOP (RCDCOP) has recently been proposed. RCDCOP models objective functions and resource constraints separately. A resource constraint is an n-ary constraint that represents the limit on the number of resources of a given type available to agents. Previous research addressing RCDCOPs employs the Adopt algorithm, which is a basic solver for DCOPs.

In this paper we propose another version of the Adopt algorithm for RCDCOP using a pseudo-tree that is generated ignoring resource constraints. The key ideas of our work are as follows: (i) The pseudo-tree is generated ignoring resource constraints. (ii) Virtual variables are introduced, representing the usage of resources. These virtual variables are used to share resources among subtrees. These ideas are used to extend Adopt. The proposed method reduces the previous limitations in the construction of RCDCOP pseudo-trees. The efficiency of our technique depends on the class of problems being considered, and we describe the obtained experimental results.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence

General Terms

Algorithms

Keywords

constraint reasoning, distributed constraint optimization problem, resource constraint, multi-agent systems

Cite as: Resource constrained distributed constraint optimization using resource constraint free pseudo-tree (Short Paper), Toshihiro Matsui, Marius Silaghi, Katsutoshi Hirayama, Makoto Yokoo and Hiroshi Matsuo, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1405-1408..

1. INTRODUCTION

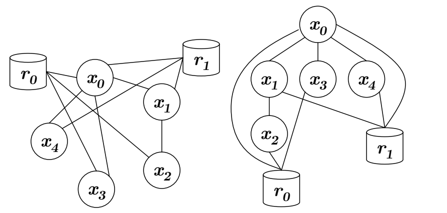
The Distributed Constraint Optimization Problem (DCOP) [2, 3, 4, 6] is a fundamental formalism for multi-agent cooperation in distributed meeting scheduling, sensor networks and other applications including distributed problem solving.

A dedicated framework called Resource Constrained DCOP (RCDCOP) has been recently proposed [1, 5] . RCDCOP models objective functions and resource constraints separately. A resource constraint is an n-ary constraint that represents the limit on the number of resources of a given type available to agents. Multiply-constrained DCOP is formalized in [1]. As an example domain, [1] describes the meeting scheduling problem with privacy requirements. Resource constrained distributed task scheduling modeled as n-ary constrained DCOPs, and the algorithm to solve such problems, are presented in [5]. The previous research addressing RCDCOPs employs the Adopt algorithm [4], which is a basic solver for DCOPs. An important graph structure for Adopt is the pseudo-tree for constraint networks. A pseudo-tree implies a partial ordering of variables. In this variable ordering, n-ary constrained variables are placed on a single path of the tree. Therefore, resource constraints that have large arity augment the depth of the pseudo-tree. This also reduces the parallelism, and therefore the efficiency of Adopt.

In this paper, we propose another version of the Adopt algorithm for RCDCOP using a pseudo-tree that is generated ignoring resource constraints. The key ideas of our work are as follows: (i) The pseudo-tree is generated ignoring resource constraints. (ii) Virtual variables are introduced, representing the usage of resources. These virtual variables are used to share resources among subtrees. These ideas are used to extend Adopt. The proposed method reduces the previous limitations in the construction of RCDCOP pseudo-trees. The efficiency of our technique depends on the class of problems being considered, and we describe the obtained experimental results.

2. PROBLEM DEFINITION: RESOURCE CONSTRAINED DCOP (RCDCOP)

A DCOP is defined by a set A of agents, a set X of variables and a set F of binary functions. Agent i has its own variable x_i. x_i takes a value from discrete finite domain D_i. The value of x_i is controlled by agent i. The cost of an assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by a binary function $f_{i,j}(d_i, d_j) : D_i \times D_j \to \mathbb{N}$. The goal is to find a global optimal solution $\mathcal{A}$ that minimizes the global cost function: $\sum_{f_{i,j} \in F,\ \{(x_i,d_i),(x_j,d_j)\} \subseteq \mathcal{A}} f_{i,j}(d_i, d_j)$.



(a) constraint network (b) resource constraint free pseudo-tree

Figure 1: Resource constrained DCOP

In RCDCOP resource constraints are added to DCOP. Resource constraints are defined by a set R of resources and a set U of resource requirements. A resource $r_a \in R$ has its capacity defined by $C(r_a) : R \rightarrow \mathbb{N}$. Each agent requires resources according to its assignment. For assignment (x_i, d_i) and resource r_a, a resource requirement is defined by $u_i(r_a, d_i) : R \times D_i \rightarrow \mathbb{N}$. For each resource, the total amount of requirement must not exceed its capacity. The global resource constraint is defined as follows: $\forall r \in R, \sum_{u_i \in U,\ \{(x_i,d_i)\} \subseteq \mathcal{A}} u_i(r, d_i) \leq C(r)$. The resource constraint takes arbitral arity. An example of RCDCOP that consists of 5 variables and 2 resources is shown Figure 1(a). In this example, x_0, x_2 and x_3 are constrained by resource R_0. x_0, x_1 and x_4 are constrained by resource R_1.

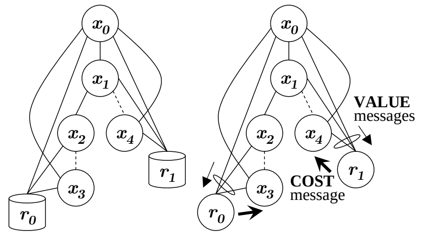
3. BACKGROUND : SOLVING RCDCOP USING ADOPT

The Adopt[4] algorithm depends on a variable ordering defined by a pseudo-tree. The edges of the original constraint network are categorized into tree edges and back edges of the pseudo-tree. The tree edges represent the partial order relation between two variables. There is no edge between different subtrees. By employing this property, Adopt performs search processing in parallel.

The processing of Adopt consists of two phases as follows. (i) **Computation of global optimal cost:** Each node computes the boundary of the global optimal cost according to the pseudo-tree. (ii) **Termination:** After computation of global optimal cost, the boundary of the cost is converged to the optimal value in the root node. Then the optimal solution is decided according to the pseudo-tree in a top-down manner.

Details of the Adopt algorithm are shown in [4]. In this paper, important modifications for Adopt are applied to computation of the global optimal cost. Agent i computes the cost using information as follows. (1) x_i: variable of agent i. Value d_i of x_i is sent to lower neighbor nodes of x_i using **VALUE** message. (2) $current\_context_i$: current partial solution of ancestor nodes of x_i. $current\_context_i$ is updated by **VALUE** message and $context$ of **COST** messages. (3) $context_i(x, d)$, $lb_i(x, d)_i$, $ub_i(x, d)$: boundary of optimal cost for each value d of variable x_i and subtree routed at child node x. These elements are received from child node x using **COST** message. If $current\_context_i$ includes $context_i(x, d)$, upper and lower bounds of cost are $lb_i(x, d)$ and $ub_i(x, d)$ respectively. If $current\_context_i$ is incompatible with $context_i(x, d)$, $context_i(x, d)$, $lb_i(x, d)_i$ and $ub_i(x, d)$ are reset to $\{\}$, 0 and ∞ respectively. (4) $UB_i(d)$, $LB_i(d)$: upper bound and lower bound of cost for value d of variable x_i and the subtree routed at x_i. They are computed using cost information.

In previous work[5], a version of the Adopt algorithm, which serializes resource constrained variables, is proposed. For example, the pseudo-tree shown in Figure 2(a) is generated from the RCDCOP shown in Figure 1(a). In this example, x_0, x_2 and x_3, which are related to resource r_0, are placed on a single path of a pseudo-tree. x_0, x_1 and x_4, which are related to resource r_1, are also placed on a single path. If it is necessary to serialize variables, extra tree edges are inserted between nodes (e. g. (x_2, x_3) and (x_1, x_4) in Figure 2(a)). In the Adopt algorithm, *Resource evaluation nodes*, which evaluate resource constraints, are introduced. A resource evaluation node is added as a child node of the lowest node of serialized nodes. For example, in Figure 2(b), extra nodes r_0 and r_1 are added as child nodes of x_3 and x_4 respectively. Each agent sends its value of variable to resource evaluation nodes using the **VALUE** message. Then the resource evaluation node evaluates the total amount of resource requirement for its resource. If the resource constraint is not satisfied, the resource evaluation node notifies its parent node using the *infinity* **COST** message. In this approach, no modification of the Adopt algorithm is necessary except adding resource evaluation nodes and handling infinity cost. However, large arity of resource constraint increases the depth of the tree, and reduces parallelism in search processing.



(a) pseudo-tree (b) computation

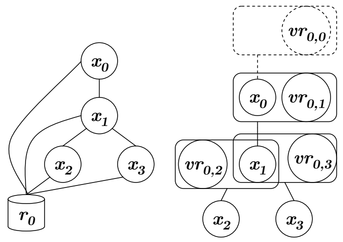
Figure 2: Serializing of resource constrained variables

4. SOLVING RCDCOP WITH RESOURCE CONSTRAINT FREE PSEUDO-TREE

In this work, we propose a novel version of the Adopt algorithm for RCDCOP. The proposed algorithm allows resource constraints related to nodes in different subtrees. The pseudo-tree is generated ignoring resource constraints. For example, the pseudo-tree shown in Figure 1(b) is generated from the RCDCOP shown in Figure 1(a). In this example, there is a constraint edge of r_0 between two different subtrees, which contain x_2 and x_3 respectively. Similarly, there is a constraint edge of r_1 between x_1 and x_4.

4.1 Introduction of virtual variables

The main idea of the proposed method is the introduction of virtual variables, which represent usage of resources. Each node shares resources with its parent node and child nodes using the virtual variables. Virtual variable $vr_{a,i}$ is defined for resource r_a and node x_i, which requires resource r_a in the subtree routed at x_i. $vr_{a,i}$ is owned by the parent node of x_i. $vr_{a,i}$ takes a value from its discrete domain $\{0, 1, \cdots, C(r_a)\}$. As a simple example, a pseudo-tree, which is related to single resource constraint, is shown in Figure 3. In this example, resource r_0 is related to variables x_0, x_1, x_2 and x_3. For these resources and variables, virtual variables $vr_{0,1}$, $vr_{0,2}$ and $vr_{0,3}$ are introduced. Each virtual variable $vr_{a,i}$ is owned by the parent node of x_i. The value of $vr_{a,i}$ is controlled by the parent node. Note that root node x_0 does not have a parent node. Therefore, it is assumed that the value of $vr_{0,0}$ is given from the virtual parent node. In this case, $vr_{0,0}$ takes a constant value that is equal to capacity $C(r_0)$ of resource r_0. Value $dr_{a,j}$ of virtual variable $vr_{a,j}$, which is owned by agent i, is sent to i's child node j using the **VALUE** message. Therefore, the **VALUE** message is modified to contain (x_i, d_i) and additional assignment $(vr_{a,j}, dr_{a,j})$. When node j receives the **VALUE** that contains $(vr_{a,j}, dr_{a,j})$, node j updates its $current\_context_j$ with new $(vr_{a,j}, dr_{a,j})$. In node i, assignments of virtual variables for



(a) pseudo-tree (b) virtual variables

Figure 3: Virtual variables for resource constraint

resource r_a should satisfy a constraint $c_{a,i}$ as follows.

$$c_{a,i}:\ dr_{a,i} \geq u_i(r_a, d_i) + \sum_{\substack{j \in child\ nodes\ of\ i \\ which\ requires\ r_a}} dr_{a,j} \tag{1}$$

Here $dr_{a,i}$ denotes the value of $vr_{a,i}$, which is received from the parent node of i. The assignment $(vr_{a,i}, dr_{a,i})$ is contained in $current\_context_i$. If an assignment does not satisfy the resource constraint $c_{a,i}$, the violation of the resource constraint is represented by infinity cost. Each node i evaluates the boundary of optimal cost for $current\_context_i$. Then the cost information is sent to the parent node of i using the **COST** message. The context of the **COST** message is modified to contain additional assignment for virtual variables of i's parent node.

In a general case, variables are related to one or more resources. Moreover, variables are related to a subset of whole resources. Virtual variables are generated according to rules as follows.

(1) Basically, if a subtree routed at node i's child node j requires resource r_a, then node i owns virtual variable $vr_{a,j}$. However, the following cases are prioritized as special cases.

(2) If node i or multiple subtrees routed at i's child nodes require r_a, then $current\_context_i$ contains assignment $(vr_{a,i}, dr_{a,i})$. In this case, $dr_{a,i}$ is decided as follows. (i) If no i's ancestor node requires r_a, then i is the root node for r_a. In this case, $dr_{a,i}$ is initialized as a constant that takes a value equal to capacity $C(r_a)$ of r_a. (ii) If node i is not the root node for r_a, then i's parent node h owns virtual variable $vr_{a,i}$. Therefore, **VALUE** messages, which are received from h, contain assignment $(vr_{a,i}, dr_{a,i})$.

(3) If node i requires resource r_a and no subtree routed at i's child node requires r_a, then i is a *leaf* node for r_a. In this case, node i has no virtual variables for r_a. Therefore, the resource constraint is defined by $dr_{a,i} \geq u_i(r_a, d_i)$.

(4) If multiple subtrees routed at i's child nodes $j \in A'$ require r_a, then i must share r_a among child nodes $j \in A'$, even if node i does not require r_a. Therefore, node i owns virtual variables $\{vr_{a,j} | j \in A'\}$.

An algorithm to generate virtual variables is shown in Algorithm 1. For the sake of simplicity, the algorithm consists of two phases of processing. As a result, node i generates set $\mathcal{X}_i$ of own variables.

4.2 Growth of search space

Additional virtual variables increase the search space. Node i selects an assignment for a set of variables $\mathcal{X}_i = \{x_i\} \cup \{vr_{a,j} | j \in Children_i, r_a \in R_j\}$. Here R_j denotes a subset of resources that are required in the subtree routed at node j. Cost evaluations in node i are modified to $\delta_i(\mathcal{D}_i)$, $LB_i(\mathcal{D}_i)$ and $UB_i(\mathcal{D}_i)$ respectively. Here $\mathcal{D}_i$ denotes a total set of assignments for $\mathcal{X}_i$. Moreover, cost information of node i's child node j is evaluated for $\mathcal{X}_{i,j} = \{x_i\} \cup \{vr_{a,j} | r_a \in R_j\}$. Therefore, they are modified to $lb_i(j, \mathcal{D}_{i,j})$, $ub_i(j, \mathcal{D}_{i,j})$, $t_i(j, \mathcal{D}_{i,j})$ and $context_i(j, \mathcal{D}_{i,j})$ respectively. As a result of these modifications, the size of the search space increases exponentially with the number of virtual variables.

If an assignment does not satisfy Equation 1, the cost of the as-

Algorithm 1: Generate virtual variables

```
Initiation$_i${
  Generate pseudo−tree ignoring resource constraint.
  if($i$ is not root node) $p_i \leftarrow$ parent node of node $i$.
  $C_i \leftarrow$ a set of child nodes of node $i$.
  $R_i \leftarrow$ a set of resources required by node $i$.
  $\mathcal{X}_i \leftarrow \{x_i\}$.
  if ( $i$ is root node ){ call Rootward$_i$(). call Leafward$_i(\phi)$. } }
Rootward$_i$(){
  $R_i^- \leftarrow R_i$.
  for each $j$ in $C_i${ call Rootward$_j$() and receive $R_j^-$. $R_i^- \leftarrow R_i^- \cup R_j^-$. } }
Leafward$_i(R_{p_i}^+)${
  $R_i^+ \leftarrow \phi$.
  for each $r$ in $R_i^-$ {
    $n \leftarrow$ number of nodes $j$ s.t. $r \in R_j^-$.
    if ($n \geq 2$ or ( $n$ = 1 and ($r \in R_i$ or $r \in R_{p_i}^+$) ) ){ $R_i^+ \leftarrow R_i^+ \cup \{r\}$. } }
  for each $j$ in $C_i${
    for each $r$ in $R_j^-$ { if($r$ is contained in $R_i^+$) $\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{vr_{r,j}\}$. }
    call Leafward$_j(R_i^+)$. } }
```

signment is ∞. A violation of a resource constraint does not depend on evaluation of other resource constraints. If an assignment violates a resource constraint for r_a, the assignment is a violated assignment even if other resource constraints are satisfied. Therefore, the assignment is pruned.

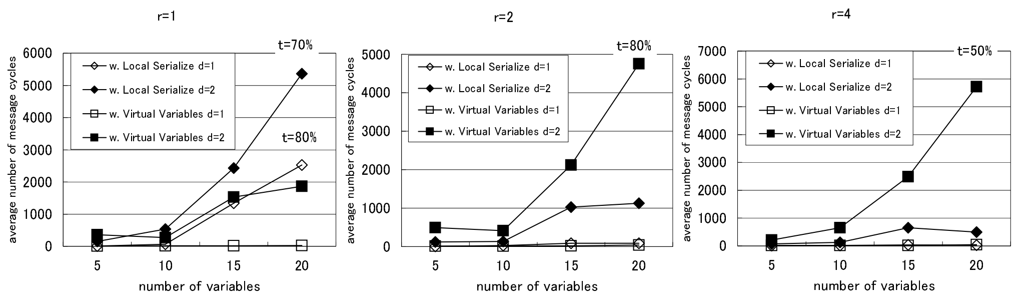
The memory space for node i's child node j increases exponentially with the number of virtual variables that are contained in $\mathcal{X}_{i,j}$. However, in the Adopt algorithm, default initial cost information is used when the cost information has not been received from the child nodes. Therefore, it is unnecessary to store the cost information that takes the initial value.

4.3 Correctness of the algorithm

The proposed method uses additional virtual variables. This modification straightforwardly extends Adopt. In each node, the original variable and virtual variables can be considered as one integrated variable. The cost evaluation and invariants for the integrated variable are the same as the original definition of Adopt. Therefore, the optimality, soundness and termination are the same as Adopt.

5. EVALUATION

The efficiency of the proposed method is evaluated by experiments. As initial experiments, we use a modified graph coloring problem with three colors. The problems are generated using parameters (n, d, r, c, l, u). The number of nodes n and link density d are the basic parameters of the graph coloring problem. The link density d is set to 1 or 2. In original graph coloring problems, this setting of parameters is used to generate a low constrained problem. However, the problem contains additional resource constraints as follows. Parameter r, c, l determines number of resources, capacity of a resource, and arity of a resource constraint respectively. In this problem setting, each variable is related to at least one resource constraint. For the sake of simplicity, the usage of a resource, which is required by an agent, is limited to 0 or 1.This means that each agent requires a unit amount of a resource or does not require one at all. Parameter u represents the ratio of a variable's values that require a resource. In these experiments u is set to $\frac{2}{3}$. Capacity c of a resource is set to $\lceil \frac{n}{2} \rceil$. Each problem instances generated so that at least one assignment globally satisfies resource constraint. 10 instances for each setting are averaged. As a competitive method, Adopt using local serialization of resource constrained variables, is also applied. Each experiment is terminated at 9999 cycles . In that



(t: ratio of correctly terminated instances (others: 100%))

Figure 4: Message cycles

Table 1: Size of pseudo-trees and dimension of assignments

n	d	r	c	l	Depth of pseudo tree		Branching factor		max. dim. of assignment	num. of infinity cost	max. total num. of cost info.	execution time (s) (Itanium2 1.6GHz 32GB Memory/C++)	
					w. Local Serialize	w. Virtual Variables	w. Local Serialize	w. Virtual Variables	w. Virtual Variables			w. Local Serialize	w. Virtual Variables
10	1	1	5	10	10.0	4.3	1.0	2.7	5.3	0	17.4	0.004	0.020
		2	3	5	7.1	4.3	1.1	2.7	6.5	0	19.3	0.001	0.055
		4	2	3	6.4	4.3	1.4	2.7	8.6	3.0	22.3	0.001	0.257
	2	1	5	10	10.0	6.8	1.0	1.5	3.6	0	60.8	0.033	0.132
		2	3	5	8.4	6.8	1.1	1.5	4.7	0	81.5	0.007	0.538
		4	2	3	7.7	6.8	1.3	1.5	6.2	176.5	116.3	0.007	2.637
20	1	1	10	20	20.0	5.3	1.0	3.5	9.6	0	50.3	0.507	32.524
		2	5	10	13.8	5.3	1.1	3.5	10.9	0	65.8	0.011	47.405
		4	3	5	10.8	5.3	1.2	3.5	13.0	0	71.1	0.002	334.243
	2	1	10	20	20.0	11.2	1.0	1.5	3.7	0	176.9	1.089	5.656
		2	5	10	17.3	11.2	1.1	1.5	5.1	0	323.5	0.192	57.735
		4	3	5	15.2	11.2	1.2	1.5	6.8	0	559.4	0.073	490.274

case, the cycle is considered as total number of message cycles. The results are shown in Figure 4 and Table 1. In these results the incorrectly terminated instances are taken into account.

In the case of $r = 1$, message cycles of the competitive method are greater than the proposed methods. In this case, the competitive method generates a linear graph as a pseudo-tree. The linear pseudo-tree causes a delay in the processing of Adopt. On the other hand, the proposed method generates a pseudo-tree ignoring resource constraints. Therefore, processing of Adopt is performed in parallel. However, in the case of $r = 2$ and 4, $d = 2$, the proposed method takes a larger number of cycles than the competitive method. In this problem, the proposed method generates multiple virtual variables for each node of a pseudo-tree. Therefore, the search space of the proposed method is increased. Results related to generated pseudo-trees, the dimension of assignments and execution time are shown in Table 1. In the competitive method, the depth of pseudo-tree increases when the number of resources is small. On the other hand, the depth of pseudo-tree does not depend on the number of resources. In the proposed method, dimension of the assignment for each node increases with the number of resources. The dimension also depends on the branching factor. The total number of cost information that is recorded in each node increases with the dimension of assignment. Computation cost of proposed method is greater than previous method. The main reason of that is internal search processing related to virtual variables.

6. CONCLUSION

We propose a distributed constraint optimization method for RCDCOP using a pseudo-tree that is generated ignoring resource constraints. The main idea is to introduce a special set of virtual variables that represents the usage of resources. The proposed method reduces the previous limitations in the construction of RCDCOP pseudo-trees. The efficiency of our technique depends on the class of problems being considered, and we described the obtained experimental results. The promising class of problems is defined by a large number of agents and few shared resource constraints Analysis of pseudo-trees to improves the efficiency of the proposed method, better representation of boundaries to prune the search processing, will be included in future work.

7. ACKNOWLEDGMENTS

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 19300048, 2007.

8. REFERENCES

[1] E. Bowring, M. Tambe, and M. Yokoo. Multiply constrained distributed constraint optimization. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.

[2] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317, Aug. 2004.

[3] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, July 2004.

[4] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[5] F. Pecora, P. Modi, and P. Scerri. Reasoning About and Dynamically Posting n-ary Constraints in ADOPT. In *7th International Workshop on Distributed Constraint Reasoning, at AAMAS, 2006*, 2006.

[6] A. Petcu and B. Faltings. A Scalable Method for Multiagent Constraint Optimization. In *9th International Joint Conference on Artificial Intelligence*, pages 266–271, Aug. 2005.