A Decentralized Variable Ordering Method for Distributed Constraint Optimization

Anton Chechetka, Katia Sycara
Carnegie Mellon University
{antonc, katia+}@cs.cmu.edu

ABSTRACT

Many different multi-agent problems, such as distributed scheduling, can be formalized as distributed constraint optimization problems (DCOP [1]). Ordering the constraint variables is an important preprocessing step of the ADOPT algorithm [1], the state of the art method of solving DCOP. Currently ADOPT uses depth-first search (DFS) trees for that purpose. For certain classes of tasks DFS ordering does not exploit the problem structure as compared to pseudo-tree ordering [3]. Also the variables are currently ordered by using a centralized scheme, which requires global information about the problem structure.

We present a variable ordering algorithm, which is both decentralized and makes use of pseudo-trees, thus exploiting the problem structure when possible. This allows to apply ADOPT to domains, where global information is unavailable, and find solutions more efficiently. The worst-case pseudo-tree depth resulting from our algorithm is $\sqrt{2k|V|}$, where V is the set of variables, and k is maximum cluster size in constraint graph. The algorithm has space and time complexity polynomial in size of the constraint graph.

Categories and Subject Descriptors

I.2 ARTIFICIAL INTELLIGENCE [**I.2.11 Distributed Artificial Intelligence**]: Coherence and coordination

General Terms

Algorithms

Keywords

Constraint satisfaction

1. INTRODUCTION

The Distributed Constraint Optimization Problem [1] is increasingly used as an underlying framework for modelling agents coordination problems. Applications of DCOP include sensor networks and distributed scheduling.

DCOP consists of variables, each having its own discrete finite domain. Every variable's value is controlled by exactly one agent. The agents must coordinate in order to minimize the global cost function, which depends on the variables. The cost function is modeled as a set of valued constraints, and every agent has knowledge only about the constraints depending on its variable. The connectivity between the agents can be represented in form of a constraint graph. The vertices of the graph are agents, and two agents have an edge between them, iff they share a constraint.

Recently an algorithm for solving DCOP, ADOPT [1] was proposed. It performs a systematic backtracking with agents acting asynchronously. To achieve that, the agents are ordered a priori in a tree such that the agents in different branches of the tree do not share any constraints. ADOPT itself does not address the process of ordering and regards it as a preprocessing step. Currently a centralized algorithm that performs depth-first traversal of the constraint graph, is used for that purpose. It is known [1] that the performance of the algorithm crucially depends on the variable ordering, thus making important the task of constructing the tree.

It is hard to find an optimal tree for ADOPT without actually solving the DCOP, but the depth of the tree is one possible approximation of the optimality criterion. Finding the minimum depth DFS tree of the graph is NP-complete task, so different heuristics are employed to obtain an approximate solution. Distributed algorithm for constructing DFS trees can be found in [2].

For certain classes of tasks restricting the possible orderings to DFS trees can have strong negative impact on ADOPT performance as compared to pseudo-trees [3]. The centralized iterative algorithm for constructing pseudo-trees was presented in [3]. It was shown for certain classes of CSP to yield better performance of the backtracking algorithm than DFS trees. Because of the similarity of CSP and DCOP algorithms, one can expect similar results for DCOP.

The centralized variable ordering algorithm does not allow to apply ADOPT to domains in which information about the constraint graph cannot be gathered and processed by a single unit.

In this paper we present an algorithm which is both decentralized and makes use of pseudo-trees. It enables one to solve DCOP problems without having to process any global information, and increases the efficiency of the solution search if the problem structure allows it. Our algorithm is a decentralized modification of the general iterative algorithm from [3]. It takes into account both the depth of the resulting pseudo-tree, and the maximum message travel time

Figure 1: Algorithm for agent x_i in the acyclic case

```
1:  d = computeDepth($x_i$);
2:  while $\neg$ ($\forall x_j \in (N_i \setminus$ ancestors) d < depth($x_j$))
3:      a = receiveAncestorInfo();
4:      append a to ancestors;
5:      d = computeDepth($x_i$);
6:  end;
7:  broadcastAncestorInfo();
8:  removeFromGraph($x_i$);
```

from root to leaves given the local communication assumptions. It yields an optimal pseudo-tree in terms of message travel time for acyclic graphs and performs no worse than DFS traversal for general graphs.

Because of lack of space, only a brief outline of the algorithm and its properties is presented, please refer to [4] for formal proofs and definitions.

2. THE ALGORITHM

2.1 Acyclic case

Suppose that the constraint graph $X =< V, E >$ is acyclic. Let N_i be the set of neighbors of variable x_i. Each agent executes the algorithm outlined in Fig. 1.

Function $\texttt{computeDepth}(\mathtt{x_i})$ on lines `1:, 5:` computes the maximum length of a simple route starting in node $\mathtt{x_i}$ in the graph, according to

$$\texttt{depth}(\mathtt{x_i}) = \max_{\mathtt{x_j} \in \mathtt{N_i}} \texttt{partialDepth}(\mathtt{x_i}, \mathtt{x_j})$$
$$\texttt{partialDepth}(\mathtt{x_i}, \mathtt{x_j}) =$$
$$\max\{0, \max_{\mathtt{x_k} \in (\mathtt{N_j} \setminus \mathtt{x_i})}(1 + \texttt{partialDepth}(\mathtt{x_j}, \mathtt{x_k}))$$

The values of $\texttt{depth}()$ and $\texttt{partialDepth}()$ are announced by a node to its neighbors as soon as they are known. The process begins with leaves announcing $\texttt{partialDepth}() = 0$ to their respective single neighbors.

On each iteration the node with minimal `depth` becomes the ancestor of all remaining nodes. Then it is removed from the graph, splitting it into several separate connected subgraphs. The process is then repeated iteratively for each of the connected subgraphs. We have proved that locally minimal `depth` value is also globally minimal and ties are possible only between neighbors, so it is enough to compare $\texttt{depth}(\mathtt{x_i})$ with its neighbors to decide whether $\mathtt{x_i}$ is the node with minimum depth.

2.2 Limited-cluster case

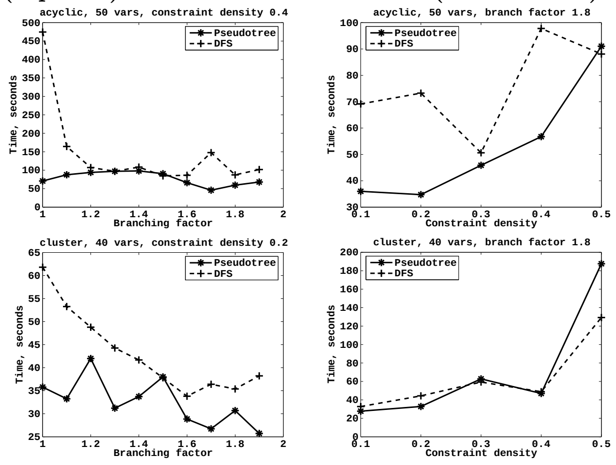
DEF. 1. *Agents x_i, x_j belong to the same cluster iff $\exists$ simple cycle in the constraint graph, containing both x_i and x_j.*

Space restrictions do not allow to give details of the algorithm in this general case. It is similar to the acyclic case, with clusters acting as meta-nodes in the constraint tree. Within the clusters the agents are ordered in a DFS tree.

2.3 Complexity and results quality

Denote k the maximum cluster size. Memory complexity of the algorithm for agent i is $O(|N_i| + \log |V|)$ for acyclic case and $O(k|V|)$ in limited-cluster case. Time complexity is $O(|V|)$ cycles for acyclic case and $O(|V| + |E| + k^{\frac{3}{2}}|V|^{\frac{1}{2}})$ for limited cluster case. The worst-case pseudo-tree depth resulting from the presented algorithm is $\sqrt{2k|V|}$.

Figure 2: ADOPT performance using DFS and pseudotree orderings. Acyclic constraint graphs (top row) and limited-cluster case (bottom row)



3. EVALUATION

We provide a comparison of the performance of ADOPT algorithm using DFS and pseudotree ordering of the variables (Fig. 2). Experiments with ADOPT were performed using agents running on 3 networked computers. Branching factor of the constraint tree (meta-tree in case of limited-cluster graphs) and average constraint density (fraction of the variables assignments that cause constraint violation) were varied. One can conclude that while neither ordering method dominates another for all types of problem structure, pseudotree ordering results in much faster DCOP solutions for problems with low constraint density. The actual ordering time for these problems was < 2 *sec.* and varied little across the problems. Most of the computation was spent on KQML messages parsing.

4. CONCLUSION

We have presented a new algorithm for variable ordering, which allows to eliminate the need to process global information when solving a DCOP. It also increases the solution search efficiency by exploiting the problem structure when possible. This extends the class of problems for which employing ADOPT is feasible and allows to apply it to completely decenralized problems. The algorithm has provable theoretical guarantee on the resulting ordering depth and polynomial time and space complexity.

5. REFERENCES

[1] P. J. Modi, W.-M. Shen, M. Tambe, M. Yokoo. "ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees". Artificial Intelligence, vol. 161, pp. 149-180, 2005

[2] N. Lynch. "Distributed Algorithms" Morgan Kaufmann Publishers, Inc., 1996.

[3] E. C. Freuder, M. J. Quinn. "Taking Advantage of Stable Sets of Variables in Constraint Satisfaction Problems". In Proc. of IJCAI-85, vol. 2, pp. 1076-1078.

[4] A. Chechetka, K. Sycara. "A Decentralized Variable Ordering Method for Distributed Constraint Optimization". Technical Report CMU-RI-TR-05-18, Robotics Institute, Carnegie Mellon University, 2005