Pseudo-Tree Construction Heuristics for DCOPs with Variable Communication Times

Atena M Tabakhi
Department of Computer Science
New Mexico State University
amtabakh@cs.nmsu.edu

Abstract

Empirical evaluations of DCOP algorithms are typically done in simulation and under the assumption that the communication times between all pairs of agents are identical, which is unrealistic in many real-world applications. In this abstract, we incorporate non-uniform communication times in the default DCOP model and propose heuristics that exploit these communication times to speed up DCOP algorithms that operate on pseudo-trees.

Introduction

Distributed Constraint Optimization Problems (DCOPs) are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents (Modi et al. 2005; Yeoh and Yokoo 2012).

Unfortunately, empirical evaluations of DCOP algorithms are typically done in simulation and under the assumption that the communication time between any pair of agents is identical for all pairs. In many coordination problems, this assumption is unrealistic. For example, in a distributed sensor network problem, communication between pairs of sensors may rely on factors such as the distance between the sensors and the topology of the environment. Therefore, in this abstract, we extend the DCOP model to include communication-related information, specifically, the communication times for each constraint as well as propose heuristics that make use of this information to speed up DCOP algorithms that operate on pseudo-trees.

We evaluate our heuristics in random graphs with communication times that depend on physical distances sampled from two distributions (uniform and Gaussian). Our experimental results show that our heuristics find pseudo-trees with smaller depths than the existing max-degree heuristic by up to 20%.

DCOPs with Variable Communication Times

We extend the DCOP model to include communication-related information, specifically, the communication times for each constraint. Therefore, this new DCOP is defined by a tuple $\langle \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{C}, \alpha \rangle$, where $\mathbf{A} = \{a_i\}_{i=1}^{p}$ is a set of *agents*; $\mathbf{X} = \{x_i\}_{i=1}^{n}$ is a set of decision *variables*; $\mathbf{D} = \{D_x\}_{x \in \mathbf{X}}$ is a set of finite *domains*, each variable $x \in \mathbf{X}$ takes values from the set $D_x \in \mathbf{D}$; $\mathbf{F} = \{f_i\}_{i=1}^{m}$ is a set of *utility functions*, each defined over a mixed set of decision variables: $f_i : \times_{x \in \mathbf{x}^{f_i}} D_x \rightarrow \mathbb{R}^+ \cup \{\perp\}$, where $\mathbf{x}^{f_i} \subseteq \mathbf{X}$ is the *scope* of f_i and $\perp$ is a special element used to denote that a given combination of values for the variables in $\mathbf{x}^{f_i}$ is not allowed; $\alpha : \mathbf{X} \rightarrow \mathbf{A}$ is a function that associates each decision variable to one agent; and $\mathbf{C} = \{c_i\}_{i=1}^{m}$ is the set of communication times, where $c_i \in \mathbf{C}$ specifies the communication time for agents in the scope $\mathbf{x}^{f_i}$ of utility function $f_i \in \mathbf{F}$ to communicate with one another. The objective is to find an optimal complete solution that maximizes the sum over all utility functions.

		Root Variable		
		h_{mws}	h_{mwa}	h_{mus}
Non-Root Variables	h_{mws}	h_1	h_2	h_3
	h_{mwa}	h_4	h_5	h_6
	h_{mus}	h_7	h_8	h_9

Table 1: Pseudo-tree Generation Heuristics

Pseudo-tree Generation Heuristics

A *DFS pseudo-tree* arrangement for G is a *spanning tree* $T = \langle \mathbf{X}, E_T \rangle$ of G such that if $f_i \in \mathbf{F}$ and $\{x, y\} \subseteq \mathbf{x}^{f_i}$, then x and y appear in the same branch of T. Edges of G that are in E_T are called *tree edges* and edges that are not are called *backedges*. Tree edges connect a node with its parent and its children, while backedges connect a node with its *pseudo-parents* and its *pseudo-children*. We use $N(a_i) = \{a_j \in \mathcal{A} \mid \{x_i, x_j\} \in E\}$ to denote the neighbors of agent a_i; and $P(a_i)$, $C(a_i)$, $PP(a_i)$, and $PC(a_i)$ to denote the parent, the set of children, the set of pseudo-parent, and the set of pseudo-children of agent a_i in the pseudo-tree.

Since constructing optimal pseudo-trees is NP-hard, one typically uses greedy approaches like the Distributed DFS algorithm (Hamadi, Bessière, and Quinqueton 1998) to construct pseudo-trees. Additionally, since the edges of pseudo-trees are now weighted by the communication times for each edge, we define the *generalized depth* of pseudo-trees to take these communication times into account.

Definition 1 (Generalized Depth) *The generalized depth of a pseudo-tree is the largest sum of communication times $c_i \in \mathbf{C}$ across all constraints over all branches of the* *pseudo-tree. More specifically, the generalized depth $\hat{d}^* = \hat{d}_{root}$ and is defined recursively by:*

$$\hat{d}_{x_i} = \max_{f_k \in \mathbf{F}: \{x_i, x_j\} \in \mathbf{x}^{f_k} \wedge x_j \in C(x_i) \cup PC(x_i)} c_k + \hat{d}_{x_j} \quad (1)$$

where f_k is the constraint between x_i and its child or pseudo-child x_j, c_k is the communication time associated with that function, and $\hat{d}_{x_j}$ is the generalized depth of the sub-tree rooted at x_j.

It is straightforward to see that this generalized depth definition subsumes the previous depth definition for pseudo-trees with uniform communication times of 1.

Complete DCOP algorithms typically require that the variables in the problem be ordered according to some complete ordering in which case the variables are ordered into a pseudo-tree; a large number of complete algorithms, including ADOPT (Modi et al. 2005) and DPOP (Petcu and Faltings 2005), operate on pseudo-trees. The Distributed DFS algorithm assigns a score to each variable according to some heuristic and initiates a DFS-traversal of the constraint graph, greedily adding the neighboring variable with the largest score as the child of the current variable. The variables' scores can be chosen arbitrarily. A commonly used heuristic is the max-degree heuristic $h(x_i) = |N(x_i)|$. We thus introduce heuristics that can be used by Distributed DFS to create pseudo-trees with small generalized depths:

- The *max-weighted-sum* (mws) heuristic h_{mws}:

$$h_{mws}(x_i) = \sum_{f_k \in \mathbf{F}: \{x_i, x_j\} \in \mathbf{x}^{f_k} \wedge x_j \in N(x_i) \setminus [\{P(x_i)\} \cup PP(x_i)]} c_k \quad (2)$$

  It sums the communication times between variable x_i and all its neighbors x_j that are not yet part of the pseudo-tree. We thus ignore neighbors that are already part of the pseudo-tree in this heuristic and both heuristics below.

- The *max-weighted-average* (mwa) heuristic h_{mwa}:

$$h_{mwa}(x_i) = \frac{h_{mws}(x_i)}{|N(x_i) \setminus [\{P(x_i)\} \cup PP(x_i)]\,|} \quad (3)$$

  It is identical to the previous h_{mws} heuristic except that it averages the values over the number of neighboring variables that are not yet part of the pseudo-tree.

- The *max-unweighted-sum* (mus) heuristic h_{mus}:

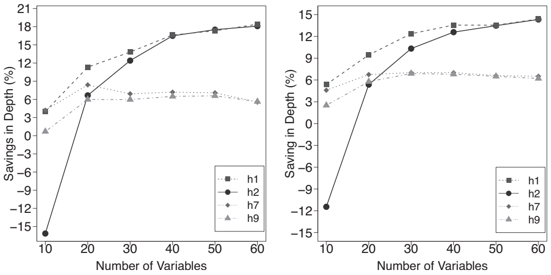
$$h_{mus}(x_i) = |N(x_i) \setminus [\{P(x_i)\} \cup PP(x_i)]\,| \quad (4)$$

  It is identical to the default max-degree heuristic except that it considers only neighboring variables that are not yet part of the pseudo-tree.

Table 1 tabulates nine combination of heuristics because the heuristic used to select the root of the pseudo-tree can differ from the heuristic used to select non-root variables.

Experimental Results

We empirically evaluate our 9 heuristics, tabulated in Table 1, against the default max-degree heuristic on random graphs. We vary the number of variables $|\mathbf{X}| = \{10, 20, 30, 40, 50, 60\}$ and set the constraint density $p_1 = 0.3$. For each configuration, we sample the physical distances d_i of the constraints from two possible truncated distributions – uniform and Gaussian $\mathcal{N}(50, 25)$ – from the range [1,100] and define the communication time $c_i = C \cdot d_i$ with these distances, where we set $C = 1$ millisecond per meter. We measure the generalized depths of pseudo-trees constructed by the heuristics and use them as proxies for runtimes of DCOP algorithms that operate on pseudo-trees.



(a) Uniform Distribution (b) Gaussian Distribution

Figure 1: Random Graphs

Figure 1 shows the results, where we plot the four best heuristics. Generally, the savings increase as the number of variables increases. The reason is because the number of possible pseudo-tree configurations increases as the number of variables increases. Thus, there is more room for improvement. The h_1 and h_2 heuristics converge to larger savings (≈18% with the uniform distribution and ≈15% with the Gaussian distribution) than the other two h_7 and h_9 heuristics (≈6% in both distributions), indicating that heuristics that take the communication times into account perform better than those that do not.

Conclusions

In this abstract, we incorporate non-uniform communication times in the default DCOP model in order to better reflect real-world applications and propose pseudo-tree construction heuristics that exploit these communication times to find pseudo-trees that are up to 20% shorter than pseudo-trees constructed by the max-degree heuristic. These heuristics can thus be used to speed up a large class of DCOP algorithms that operate on pseudo-trees.

References

Hamadi, Y.; Bessière, C.; and Quinqueton, J. 1998. Distributed intelligent backtracking. In *Proceedings of ECAI*, 219–223.

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

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *Proceedings of IJCAI*, 1413–1420.

Yeoh, W., and Yokoo, M. 2012. Distributed problem solving. *AI Magazine* 33(3):53–65.