Multi-Variable Agents Decomposition for DCOPs to Exploit Multi-Level Parallelism*

(Extended Abstract)

Ferdinando Fioretto
Computer Science Department
New Mexico State University
Las Cruces, NM 88003, USA
ffiorett@cs.nmsu.edu

William Yeoh
Computer Science Department
New Mexico State University
Las Cruces, NM 88003, USA
wyeoh@cs.nmsu.edu

Enrico Pontelli
Computer Science Department
New Mexico State University
Las Cruces, NM 88003, USA
epontell@cs.nmsu.edu

ABSTRACT

Current DCOP algorithms suffer from a major limiting assumption—each agent can handle only a single variable of the problem—which limits their scalability. This paper proposes a novel *Multi-Variable Agent* (MVA) DCOP decomposition, which: *(i)* Exploits co-locality of an agent's variables, allowing us to adopt efficient centralized techniques; *(ii)* Enables the use of hierarchical parallel models, such us those based on GPGPUs; and *(iii)* Empirically reduces the amount of communication required in several classes of DCOP algorithms. Experimental results show that our MVA decomposition outperforms non-decomposed DCOP algorithms, in terms of network load and scalability.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

Keywords

Distributed Constraint Optimization; DCOP; GPGPU

1. INTRODUCTION

Distributed Constraint Optimization (DCOP) [6] is an elegant formalism to model cooperative multi-agent problems and has been applied to solve coordination and resource allocation problems. However, the applicability of DCOPs in large problems faces a main limitation due to the common assumption that each agent controls exclusively one variable. To cope with such restriction, reformulation techniques are used to transform a general DCOP into one where each agent controls exclusively one variable. There are two commonly used reformulation techniques [1, 7]: *(i) Compilation*, where each agent creates a new *pseudo-variable*, whose domain is the Cartesian product of the domains of all variables of the agent; and *(ii) Decomposition*, where each agent creates a *pseudo-agent* for each of its variables. While both techniques are relatively straightforward, they can be inefficient. In compilation, the memory requirement of each agent grows exponentially with the number of variables that it controls. In decomposition, the DCOP algorithms will treat two pseudo-agents as independent entities, resulting in unnecessary computation costs. This paper aims at overcoming these limitations by proposing a new *Multi-Variable Agent* (MVA) DCOP decomposition, which enables a separation between the agents *local subproblems*, which can be solved independently using centralized solvers and GPGPUs, and the DCOP *global problem*, which requires coordination of the agents.

*This research is partially supported by NSF grant HRD-1345232. 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, or the U.S. government.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

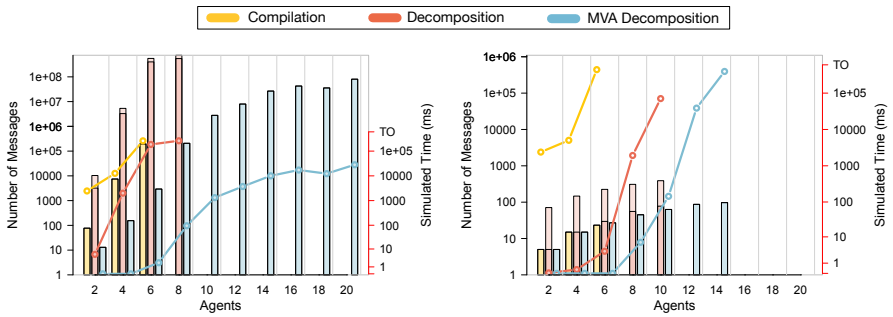
2. BACKGROUND

DCOP: A DCOP is defined by $\langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$, where: $\mathcal{X} = \{x_1, \dots, x_n\}$ is a set of *variables*, $\mathcal{D} = \{D_1, \dots, D_n\}$ is a set of finite *domains*, where D_i is the domain of variable x_i, $\mathcal{F} = \{f_1, \dots, f_m\}$ is a set of *cost functions* (also called *constraints*), where each k-ary cost function $f_i : D_{i_1} \times D_{i_2} \times \dots \times D_{i_k} \mapsto \mathbb{N} \cup \{0, \infty\}$ specifies the cost of each combination of values of variables in its *scope* $\mathbf{x}^i = \langle x_{i_1}, \dots, x_{i_k} \rangle$, $\mathcal{A} = \{a_1, \dots, a_p\}$ is a set of *agents*; and $\alpha : \mathcal{X} \to \mathcal{A}$ maps each variable to one agent. Given a DCOP P, $G_P = (\mathcal{X}, E_{\mathcal{F}})$ is the *constraint graph* of P, where $\{x, y\} \in E_{\mathcal{F}}$ iff $\exists f_i \in \mathcal{F}$ s.t. $\{x, y\} \subseteq \mathbf{x}^i$. A *solution* σ is a value assignment for a set $X_\sigma \subseteq \mathcal{X}$ of variables that is consistent with their respective domains. The utility $F(\sigma) = \sum_{f_i \in \mathcal{F}, \mathbf{x}^i \subseteq X_\sigma} f(\sigma)$ is the sum of the utilities across all the applicable utility functions in σ. A solution σ is *complete* if $X_\sigma = \mathcal{X}$. The goal is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} F(\mathbf{x})$.

GPGPU: *General Purpose Graphics Processing Units* (GPGPUs) are multiprocessor devices, offering hundreds of computing cores and a rich memory hierarchy. A parallel computation is executed by several *threads* and can be substantially faster than its sequential (i.e., ran on CPU) counterpart. GPGPUs support *Single-Instruction Multiple-Thread* (SIMT) processing. In SIMT the same instruction is executed by different threads, while handling different data.

3. MVA DECOMPOSITION

For each agent $a_i \in \mathcal{A}$, we define the set of its *local* variables: $L_i = \{x_j \in \mathcal{X} \mid \alpha(x_j) = a_i\}$; the set of its *boundary* variables: $B_i = \{x_j \in L_i \mid \exists x_k \in \mathcal{X} \wedge \exists f_s \in \mathcal{F} : \alpha(x_k) \neq a_i \wedge \{x_j, x_k\} \subseteq \mathbf{x}^s\}$; and its *local constraint graph*: $G_i = (L_i, E_{\mathcal{F}_i})$, where $\mathcal{F}_i = \{f_j \in \mathcal{F} \mid \mathbf{x}^j \subseteq L_i\}$. In the MVA decomposition, a DCOP problem P is decomposed into $|\mathcal{A}|$ sub-problems $P_i = (L_i, B_i, \mathcal{F}_i)$. In addition to the decomposed problem P_i, each agent receives the *global* DCOP algorithm P_G, which is common to all agents in the problem and defines the agent's coordination protocol and the behavior associated to the receipt of a message, and the *local* algorithm P_L, which can differ between agents and is used to solve the agent's



		DPOP-DFBnB		DPOP-Gibbs	
		CPU	*GPGPU*	*CPU*	*GPGPU*
$\|L_i\|$	6	17000	2000	45	3
	12	timeout	$2.28 \cdot 10^7$	211	4
	18	timeout	timeout	447	6
	24	timeout	timeout	1271	17
	30	timeout	timeout	2275	36
$\|D_i\|$	6	3400	450	44	8
	12	61000	5000	315	22
	18	129000	30000	836	80
	24	437000	48000	1946	204
	30	988000	94500	5853	432

simulated time (ms)

Figure 1: Random Graph Instances with CPUs: AFB-DFBnB (left), DPOP-DFBnB (middle), and CPU vs. GPGPUs (right)

subproblem. The execution of the MVA framework for each agent a_i has the following conceptual phases:

1. *Wait*: The agent waits for incoming messages. If the received message results in a new value assignment $\sigma(x_r, k)$ for a boundary variable x_r of B_i, then the agent will proceed to Phase 2. If not, it will proceed to Phase 4.

2. *Check*: The agent checks if it has performed a complete new assignment for all its boundary variables, indexed by the number k, which establishes an enumeration for the boundary variables' assignments. If it has, then the agent will proceed to Phase 3, otherwise it will return to Phase 1.

3. *Local Optimization*: When a complete assignment is given, the agent passes the control to a local solver, which solves the following problem:

$$\min \sum_{f_j \in \mathcal{F}_i} f_j(\mathbf{x}^j) \quad \text{subject to: } x_r = \sigma(x_r, k) \quad \forall x_r \in B_i \qquad (1)$$

Solving this problem results in finding the best assignment for the agent's local variables given the particular assignment for its boundary variables. Notice that the local solver P_L is independent from the DCOP structure and it can be customized based on the agent's local requirements. Once the agent solves its subproblem, it proceeds to Phase 4.

4. *Global Optimization*: The agent processes the new assignment as established by the DCOP algorithm P_G, executes the necessary communications, and returns to Phase 1.

The agents can execute these phases independently of one another because they exploit the co-locality of their local variables without any additional privacy loss. In addition, the local optimization process can operate on $m > 1$ combinations of value assignments of the boundary variables, before passing control to the next phase. This is the case when the agent explores m different assignments for its boundary variables in Phases 2 and 3. These operations are performed by storing the best local solution and their corresponding costs in a cost table of size m, which we call MVA_TABLE. Using the MVA decomposition, each agent computes only the necessary rows of the table on demand.

For the local optimization process within each agent we use DFBnB and Gibbs sampling [2] as representative complete and incomplete algorithms. The use of hierarchical parallel solutions is motivated by the observation that the search for the best local solution for each row of the MVA_TABLE is independent of the search for another row and, as such, they can be performed in parallel. This observation finds a natural fit for SIMT processing and, therefore, in addition to the CPU versions of DFBnB and Gibbs, we provide their GPGPU counterparts. The use of GPGPUs allows us to speed up the local optimization process and, consequently, reduces the overall DCOP solving time.

4. EXPERIMENTAL RESULTS

We evaluate our MVA decomposition with two global DCOP algorithms (AFB [3] and DPOP [4]) and two local centralized solvers (DFBnB and Gibbs) implemented on CPUs and GPGPUs, and compare it against the Compilation and the Decomposition preprocessing techniques on random graphs. We report the simulated time [5] as well as the network load, averaged over 50 instances and impose a timeout of 600s. Figure 1 shows the results, where AFB and DPOP uses DFBnB as local solver. Dark (light) bars indicate the number of external (internal) agent-to-agent messages, and lines indicate runtime, all in logarithmic scale (the smaller, the better). We vary the number of agents and fix the number of local variables, their domain size, the density of the local and global constraint graphs, and the constraint tightness to $6, 4, 0.6, 0.4$, and 0.4, respectively. Figure 1 (right) illustrates the speedup obtained by parallelizing DFBnB and Gibbs on GPGPUs, using DPOP as the global algorithm. We fix $|\mathcal{A}| = 4$ and vary $|L_i|$ and $|D_i|$. Unlike Decomposition, MVA and Compilation do not need internal agent communication since agent subproblems are solved locally within each agent. The number of external messages required by each framework is similar for DPOP because they are linear in the number of agents, and in turn independent of the number of local variables. AFB on MVA requires up to one order of magnitude fewer messages compared to Compilation, and several orders of magnitude fewer compared to Decomposition. The reason is that AFB agents uses messages to request for cost estimates and announce complete solutions. These requests occur more regularly with Decomposition and Compilation than with the MVA decomposition. All the algorithms are fastest on the MVA framework followed by with Decomposition and Compilation. Finally, the speedup obtained by parallelizing the MVA algorithms on GPGPUs is at least one order of magnitude in most configurations.

REFERENCES

[1] D. Burke and K. Brown. Efficiently handling complex local problems in distributed constraint optimisation. In *ECAI*, pages 701–702, 2006.

[2] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE TPAMI*, 6(6):721–741, 1984.

[3] A. Gershman, A. Meisels, and R. Zivan. Asynchronous Forward-Bounding for distributed COPs. *JAIR*, 34:61–88, 2009.

[4] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 1413–1420, 2005.

[5] E. Sultanik, P. J. Modi, and W. C. Regli. On modeling multiagent task scheduling as a distributed constraint optimization problem. In *IJCAI*, pages 1531–1536, 2007.

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

[7] M. Yokoo, editor. *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*. Springer, 2001.