Generalizing DPOP: Action-GDL, a new complete algorithm for DCOPs

(Extended Abstract)

M. Vinyals
IIIA, Artificial Intelligence Research Institute
Spanish National Research Council
meritxell@iiia.csic.es

J.A. Rodriguez-Aguilar
IIIA, Artificial Intelligence Research Institute
Spanish National Research Council
jar@iiia.csic.es

J. Cerquides *
WAI, Dep. Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
cerquide@maia.ub.es

ABSTRACT

In this paper we make three main contributions (fully detailed in [5]). Firstly, we formulate a new algorithm, the so-called Action-GDL, which extends GDL [1] to apply it to Distributed Constraint Optimization Problems (DCOPs). Secondly, we show that Action-GDL generalizes DPOP[4], a low-complexity, state-of-the-art algorithm to solve DCOPs. Finally, we provide empirical evidence showing that Action-GDL can outperform DPOP in terms of the amount of computation, communication and parallelism.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms theory

Keywords

GDL, DCOP, Distributed Junction Tree, DPOP

Action-GDL is a novel, complete algorithm that extends GDL [1] to efficiently solve DCOPs. GDL is a general message-passing algorithm that exploits the way a global function factors into a combination of local functions generalizing a large family of well-known algorithms. In our case, the rationale to apply (and extend) GDL is that a DCOP requires the maximization of a global function resulting from the combination of local functions.

GDL is defined over two binary operations [1]. In our case over addition and maximization (the max-sum GDL) because we aim at maximizing some objective function. In order to ensure optimality and convergence, GDL arranges the objective function to assess in a *junction tree structure* (JT)[2].

*Partially funded by TIN2006-15662-C02-01. M.Vinyals is supported by the Ministry of Education of Spain (FPU grant AP2006-04636). JAR thanks JC2008-00337.

Cite as: Generalizing DPOP: Action-GDL, a new complete algorithm for DCOPs, (Extended Abstract), M. Vinyals, J.A. Rodriguez-Aguilar, J. Cerquides, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1239–1240

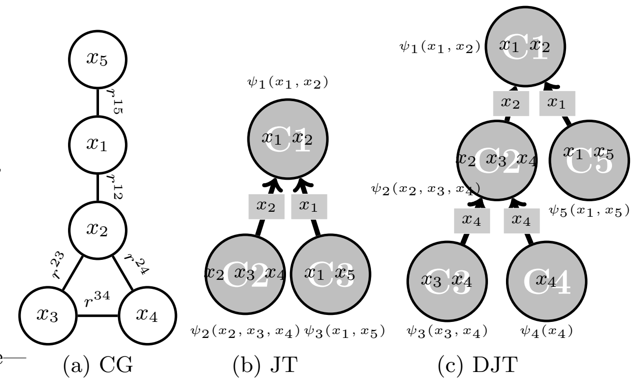

Figure 1: CG and JT/DJT arrangements

Fig. 1(a) shows an example of DCOP represented by its constraint graph (CG). We use the definition and nomenclature for the DCOP as formulated in [5]. Fig.1(b) shows one of the possible JTs for this DCOP, a tree of three cliques, where each clique is a subset of variables of the DCOP. Nodes in the figure stand for cliques and edges for separators. Thus, for example, C_1 contains variables x_1, x_2; C_3 contains variables x_1, x_5; and their separator is composed of their intersection x_1. Each clique C_i is associated with a potential ψ_i, a function whose domain is a subset of C_i. Moreover, by making $\Psi = \{\psi_1 = r^{12}, \psi_2 = r^{23} + r^{34} + r^{24}, \psi_3 = r^{15}\}$ the function encoded is the same as the one in the CG.

Message/local knowledge ($\widehat{\mathcal{K}}$)
1. $\mu_{21} = \max_{x_3,x_4} \psi_2(x_2,x_3,x_4)$
2. $\mu_{31} = \max_{x_5} \psi_3(x_1,x_5)$
3. $\widehat{\mathcal{K}}_1 = \psi_1(x_1,x_2) + \mu_{21}(x_2) + \mu_{31}(x_1)$
4. $\mu_{12} = \max_{x_1} \psi_1(x_1,x_2) + \mu_{31}(x_1)$
5. $\mu_{13} = \max_{x_1} \psi_1(x_1,x_2) + \mu_{21}(x_2)$
6. $\widehat{\mathcal{K}}_2 = \psi_2(x_2,x_3,x_4) + \mu_{12}(x_2)$
7. $\widehat{\mathcal{K}}_3 = \psi_3(x_1,x_5) + \mu_{13}(x_1)$

Messages/local knowledge $\widehat{\mathcal{K}}$
1. $\mu_{21} = \max_{x_3,x_4} \psi_2(x_2,x_3,x_4)$
2. $\mu_{31} = \max_{x_5} \psi_3(x_1,x_5)$
3. $\widehat{\mathcal{K}}_1 = \psi_1(x_1,x_2) + \mu_{21}(x_2) + \mu_{31}(x_1)$
4. $\{c_1^*, c_2^*\} = arg\max_{x_1,x_2} \widehat{\mathcal{K}}_1(x_1,x_2)$
5. $\sigma_{12} = c_2^*$, $\sigma_{13} = c_1^*$
6. $\widehat{\mathcal{K}}_2 = \psi_2(\sigma_{12},x_3,x_4)$
7. $(c_3^*, c_4^*) = arg\max_{x_3,x_4} \widehat{\mathcal{K}}_2(x_3,x_4)$
8. $\widehat{\mathcal{K}}_3 = \psi_3(\sigma_{13},x_5)$
9. $c_5^* = arg\max_{x_5} \widehat{\mathcal{K}}_3(x_5)$

Table 1: Traces of GDL (left) and Action-GDL (right)

GDL defines a message-passing phase for cliques to exchange information about their variables. Once a clique has received messages from all its clique neighbors it has all information related to its variables. Table 1 (left) displays a trace of GDL over the JT in figure 1(b). At step 1, clique $\mathcal{C}_2 = \{x_2, x_3, x_4\}$ sends a message μ_{21} to clique $\mathcal{C}_1 = \{x_1, x_2\}$ with the values of its local function, ψ_2, after 'filtering out' dependence on all variables but those common to $\mathcal{C}_2$ and $\mathcal{C}_1$ (namely variables which are not in their separator). An equivalent process is executed by clique C_3 to send a message to C_1 (step 2). At step 3, after clique $\mathcal{C}_1$ receives the values of its children's local functions for its variables x_1, x_2, it combines them with its potential into its local knowledge $\widehat{\mathcal{K}}_1$. At that point, since $\mathcal{C}_1$ has received messages from all its neighbors, $\widehat{\mathcal{K}}_1$ contains all the information related to its variables x_1, x_2. At steps 4 and 5, clique $\mathcal{C}_1$ sends messages to its children that contain the combination of its local function, ψ_1, with other children messages filtering out all variables in the separator. Thus, $\mathcal{C}_2$ receives a message from $\mathcal{C}_1$ that contains the potential ψ_1 combined with μ_{31} and filtered out over x_2. Then, it can compute $\widehat{\mathcal{K}}_2$ (step 6).

However, the capability of computing any objective function, as provided by GDL, is not enough when solving DCOPs. We need to go one step beyond GDL to allow a group of agents make a joint decision (regarding their variables' values) that maximizes any objective function. For this purpose, Action-GDL extends GDL by: (1) supporting the distribution of the problem; and (2) inferring decision variables.

Supporting the distribution of the problem. GDL runs over a JT in which all cliques are considered to be located in a single agent, which is in charge of running GDL. Action-GDL solves a DCOP where variables and relations are distributed over agents that cooperatively solve the problem. Therefore, Action-GDL extends GDL to deal with cliques that are distributed to different agents and control that agents have knowledge about the local information (potential) related to its cliques. This is accomplished by running Action-GDL over a distributed junction tree (DJT), where each clique is assigned to an agent. Fig.1(c) shows a DJT for the DCOP of figure 1(a). This DJT has 5 cliques, one for each agent of the DCOP (clique C_i is assigned to agent a_i). The set of potentials contains the set of relations of the DCOP distributed as follows: $\psi_1 = r^{12}, \psi_2 = r^{23} + r^{24}, \psi_3 = r^{34}, \psi_4 = \{\}, \psi_5 = r^{15}$. Notice that this DJT has the property that agents are assigned a clique whose potential contains relations that this agent knows (in DCOP relations that contains some agent's variable). Thus, agent 2 is assigned clique 2 whose potential contains relations that include variable x_2, namely r^{23}, r^{24}. That is not true in the JT of figure 1(b) since in that case there is not a single agent who knows all relations assigned to potential ψ_2, namely r^{23}, r^{34}, r^{24}.

To compile such DJT, we propose to use the method introduced in [3] during the Action-GDL pre-processing phase, which allows agents to distributedly compile a DJT to feed Action-GDL. The advantatge of this method is that it captures how relations aredistributed among agents.

Inferring decision variables In a DCOP, clique variables are decision variables. Computing a clique's objective function stands for assigning values to these decisions. As explained above, when a clique in GDL has received messages from all its neighbors, it has all information related to its variables and it can infer their values. Therefore, when a clique infers their state solving a DCOP, there is no need to propagate more information related to its variables down the tree, and instead it can directly propagate its decisions. Hence, once all cliques have received messages from all their children (messages sent up the tree), the second message-passing phase of GDL (messages sent down the tree to children) is no longer necessary. Instead, this phase is replaced by a message-passing phase for cliques to exchange *decisions* with their children (down the tree), which is precisely the extension that Action-GDL introduces. Henceforth, we shall refer to the first message-passing phase as *utility propagation*, and to the second one as *value propagation*. Very importantly, notice that the value propagation phase ensures that whenever multiple optimal joint decisions are feasible, cliques converge to the very same joint decision, namely to the very same solution of a DCOP.

To illustrate that change, table 1 (right) displays a trace of Action-GDL over the JT in figure 1(b). Steps 1-3 are equivalent to steps 1-3 in GDL, since they correspond to message sent up to the tree (messages sent during the utility propagation phase). However, at step 4 the root clique C_1 has received messages from all their children and can assess the optimal value for x_1, x_2, namely c_1^*, c_2^*. At that point, it starts the value propagation phase, and C_1 propagates the optimal value for x_1, x_2 down the tree to $\mathcal{C}_2$ and $\mathcal{C}_3$ through value messages σ_{12} σ_{13} respectively (step 5). At steps 6-7, $\mathcal{C}_2$ assesses the values of x_2, x_4 using its parent inferred value for x_2, namely c_2^*. Same process is repeated in steps 8-9 for C_3 using its parent inferred value for x_1.

Finally, we claim that DPOP executions are equivalent to the execution of Action-GDL under certain DJTs. To prove that, in [5] we: (i) define a mapping from pseudotrees to a subclass of DJTs; and (ii) prove that, given any pseudotree, the execution of DPOP over the pseudotree is equivalent to the execution of Action-GDL over the DJT produced by our mapping for the pseudotree. Since given a pseudotree there is a DJT such that Action-GDL execution is equal to DPOP execution, Action-GDL can be at least as efficient as DPOP (by mimicking its behavior) when solving DCOPs. Moreover, Action-GDL can yield better algorithmic performance than DPOP. Action-GDL can achieve such improvement because: (i) DJTs allow to explore problem arrangements that can- not be represented via pseudotrees; and (ii) it can assess multiple variables' values at once. Hence, our early empirical results, included in [5], indicate that alternative DJT arrangements can lead to significant savings in communication and computation costs (which increase as the number of variables grow) and to reductions of the maximum degree of parallelism (from 25% to 40% of reduction).

1. REFERENCES

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[2] F. V. Jensen and F. Jensen. Optimal junction trees. In *UAI*, pages 360–366, 1994.

[3] M. A. Paskin, C. Guestrin, and J. McFadden. A robust architecture for distributed inference in sensor networks. In *IPSN*, pages 55–62, 2005.

[4] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.

[5] M. Vinyals, J.A.Rodriguez-Aguilar, and J. Cerquides. Proving the equivalence of action-gdl and dpop. Technical report, IIIA-CSIC, 2008. Available at `http://www2.iiia.csic.es/~meritxell/publications/TRR200804.pdf`.