ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-1025

Egalitarian Utilities Divide-and-Coordinate: Stop arguing about decisions, let's share rewards!

Meritxell Vinyals and **Juan Antonio Rodriguez-Aguilar** and **Jesus Cerquides**[1]

Abstract. In this paper we formulate a novel Divide-and-Coordinate (DaC) algorithm, the so-called Egalitarian Utilities Divide-and-Coordinate (EU-DaC) algorithm. The *Divide-and-Coordinate* (DaC) framework [3] is a family of bounded DCOP algorithms that solve DCOPs by exploiting the concept of agreement. The intuition behind EU-DaC is that agents get closer to an agreement, on the optimal solution, when they communicate the local max-marginals utilities for their assignments instead of only their preferred assignments. We provide empirical evidence supporting this hypothesis as well as illustrating the competitiveness of EU-DaC.

1 Introduction

A Distributed Constraint Optimization Problem (DCOP) [1] is a formal framework proposed to model cooperative networks where agents need to coordinate in a decentralized manner to find the joint actions that maximize their joint reward. In this work we focus on DCOP incomplete algorithms with quality guarantees that provide locally optimal solutions at low cost. By quality guarantees we mean that agents can bound the error over their solutions. Quality guarantees allow agents to be aware of the goodness of their decisions and detect when they eventually converge to poor solutions.

At this aim we focus in our previous work in the *Divide-and-Coordinate* (DaC) framework [3] which make headway in this direction. The DaC framework is a family of incomplete algorithms that allows to solve DCOPs by exploiting the concept of *agreement*. The DaC framework provides an upper bound on the quality of the optimal solution that agents can use to return per-instance quality guarantees. The Divide-and-Coordinate Sugradient Algorithm (DaCSA) [3] is a DaC algorithm in which agents coordinate by exchanging their preferred variable assignments.

Our main contribution in this paper is a new DaC algorithm, the Egalitarian Utilities Divide and Coordinate algorithm (EU-DaC), a bounded anytime DCOP algorithms in which agents coordinate using the individual max-marginal utilities to get closer to an agreement. Concretely, we advance the state-of-the-art by : (1) formulating a new DaC algorithm, EU-DaC, where agents coordinate over the information of their max-marginals utilities instead of their individual preferences; and (2) empirically showing that EU-DaC usually outperforms DaCSA and gets similar results than MGM-algorithms.

2 Divide-and-Coordinate: an overview

The key idea behind the DaC approach, introduced in [3], is the following: since solving a DCOP is NP-Hard, we can think of *dividing* an intractable problem into simpler subproblems that can be individually solved by each agent. Figure 1 shows an initial division of a 3-agent DCOP in which each agent uses its local relations to create its subproblem. For instance, the problem of agent a_1 is composed of its local relation r_1 over its variable x_1 and all binary relations shared with its neighbours (r_{12},r_{23}) (the table on the left shows the rewards for binary relations). When solving individual subproblems, agents may assign different values to their shared variables, thus causing conflicts between assignments. For instance, as shown by the local assignments in figure 1, agent a_1 disagrees with a_2 and a_3 on x_1. Thereafter, each agent proceeds to coordinate, during the so-called *coordinate* stage, by exchanging information about its disagreements with its neighbours. Agents subsequently employ information on disagreements to jointly update their subproblems, during the *divide* stage, to move closer to an agreement. In figure 1, a_1 will exchange information about its conflict over x_1 with a_2 and a_3, and will use such information to update its subproblem.

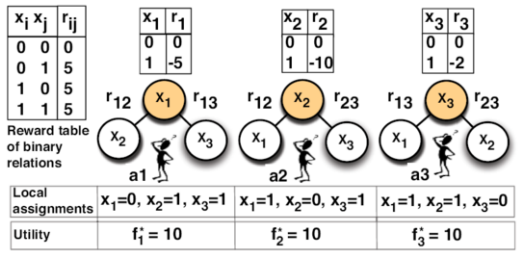

Figure 1. Example of an initial division in a 3 agent DCOP.

Then, the DaC framework is found on the following propositions:

(**Proposition 1**) *The sum of the solutions of individual subproblems is an upper bound on the quality of the optimal solution.*

(**Proposition 2**) *If agents reach an agreement on a joint solution when optimizing subproblems, such solution is the optimal.*

Although proposition 2 tells us that agreements stand for optimal solutions, when agents do not agree on their assignments, they can still provide bounded anytime solutions. At this aim, after exchanging information on disagreements with its neighbours, agents distributedly generate a *candidate* solution: an assignment for their variables as close to the agreement as they can. The quality of this *candidate* solution is bounded by the upper bound of proposition 1.

Observe that the DaC framework does not make explicit: (1) *what information* agents exchange about the disagreement during the coordinate stage; and (2) *how to modify subproblems* in the divide stage based on the coordination information to get closer to an agreement. Hence, particular implementations of these operations lead to different DaC algorithms. Algorithm 1 presents the pseudocode for a general DaC algorithm, whose operation is described below:

Initialization (lines 1-2). Each agent a_i starts by creating its local problem $\bar{\Phi}_i^0$ using its local relations.

Divide (lines 4-6). Each agent updates its local problem with coor-

[1] Artificial Intelligence Research Institute (IIIA), Spanish Scientific Research Council (CSIC), Campus UAB, Bellaterra, Spain

Algorithm 1 DaC(Φ)

Each agent a_i runs:

1: $bound \leftarrow \infty$; $solution, C_i \leftarrow \emptyset$; $bestValue \leftarrow -\infty$;
2: $\overline{\Phi}_i^0 \leftarrow$ createSubproblem($\langle \mathcal{X}^i, \mathcal{D}^i, \mathcal{R}^i \rangle$);
3: **repeat**
4: **/* Divide stage */**
5: $\overline{\Phi}_i^t \leftarrow$ *updateSubproblem*($\overline{\Phi}_i^{t-1}, \{\Psi_i\}$);
6: $(d_i^{*,t}, f_i^{*,t}) \leftarrow$ solveSubproblem($\overline{\Phi}_i^t$);
7: **/* Coordinate stage */**
8: **for** $x_v \in Neighbours(x_i)$ **do**
9: $\Psi_i^v \leftarrow$ *wrapCoordinationInfo*($d_i^{*,t}, f_i^{*,t}, C_i^{t-1}, \{\Psi\}$);
10: $\Psi_v^i \leftarrow$ exchangeCoordinationInfo(Ψ_i^v);
11: **end for**
12: **/*Update bounded anytime solution*/**
13: $C_i^t \leftarrow$ generateCandidateSolutions(x_i, C_i^{t-1});
14: **if** betterBoundOrSolutionAvailable($\{\Psi\}$) **then**
15: Update $bound, solution, bestValue$ if applies
16: **end if**
17: **until** termination condition satisfied
18: **return** $\langle solution, bestValue, bound \rangle$

dination information and subsequently solves it to obtain its preferred assignment (d_i^*) along with its utility (f_i^*). Since the DaC framework does not specify how each agent uses the information gathered from its neighbours ($\{\Psi_i\}$) to update its subproblem, the *updateSubproblem* function (line 5) is an abstract method.

Coordinate (lines 7-11). Each DaC algorithm must implement the *wrapCoordinationInfo* function (line 9) that assesses the coordination information (Ψ_i^v) that a_i exchanges with its neighbour a_v.

Bounded anytime solutions (lines 12-16). Each agent a_i generates a *candidate* solution for its variable x_i ($\mathcal{C}_i^t$ at line 13) by, for example, selecting the assignment in which most agents agree on. Agents distributedly assess candidate solution and bound's values of each pair of *divide* and *coordinate* stages and use them to update the bounded anytime solution (lines 14-16).

3 Egalitarian Utilities Divide-and-Coordinate

We propose the Egalitarian Utilities Divide and Coordinate algorithm (EU-DaC), a novel DaC algorithm that has agents coordinate by: (1) using their max-marginal utilities as information about their disagreement; and (2) updating their subproblems to get max-marginals closer to those of their neighbours. The rationale behind EU-DaC is that if agents agree on the max-marginal utilities, then they agree on their local assignments. According to DaC [3], this implies that agents have found a DCOP solution. Next we describe how EU-DaC implements the *coordinate* and *divide* stages of the DaC framework.

Coordinate. Each agent communicates to each one of its neighbours the maximum utility it can obtain for each one of the possible assignments of their shared variables, the so-called *max-marginal utility*. Consider the example in figure 1, the max-marginal utilities of a_1 for its variable x_1 are: $\mathcal{U}_1^1(0) = \max_{d \in \mathcal{D}_{23}} r_1(0) + r_{12}(0,d) + r_{13}(0,d) = 10$ and $\mathcal{U}_1^1(1) = \max_{d \in \mathcal{D}_{23}} r_1(1) + r_{12}(1,d) + r_{13}(1,d) = 5$. Hence, a_1 reports a max-marginal utility of 10 when setting x_1 to 0 and of 5 when setting it to 1. Thereafter each agent is ready to assess which utility changes apply to its subproblem to get closer to its neighbours. We refer to these utility changes as *coordination relations*.

On the one hand, each agent a_s assesses, for each variable of its neighbours $x_i \in N(x_s)$, a coordination relation ($\triangle_i^{s,t}$) that quantifies how much a_s must change its utility for x_i to agree with a_i:

$$\triangle_i^{s,t}(d) = \mathcal{U}_i^{i,t}(d) - \mathcal{U}_i^{s,t}(d) \quad (1)$$

On the other hand, a_s also assesses a coordination relation for its own variable x_s ($\bigtriangledown_s^{s,t}$) to counterbalance the utility updates of its neighbours:

$$\bigtriangledown_s^{s,t}(d) = - \sum_{x_j \in N(a_s)} \triangle_j^{s,t}(d) \quad (2)$$

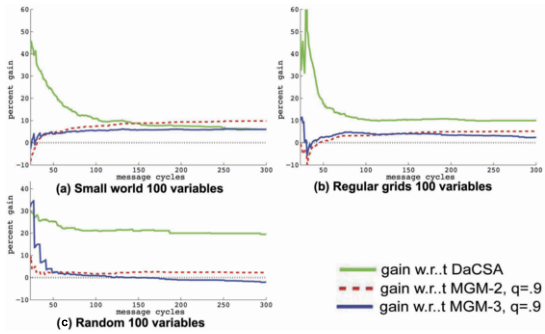

Figure 2. Percent gain of EU-DaC with respect to DaCSA and MGM-{2,3}

Divide. Each agent uses its coordination relations to update its local subproblem. The goal of an EU-DaC agent when updating its subproblem is to converge on the max-marginal utilities of its neighbours. Therefore, at each iteration t, each agent a_s updates its subproblem f_s^t by adding the coordination relation for each variable:

$$f_s^t(d) = f_s^{t-1}(d) + \gamma \cdot [\bigtriangledown_s^{s,t} + \sum_{x_j \in N_s} \triangle_j^{s,t}] \quad (3)$$

where $\gamma \in (0, 1]$ is a damping parameter that weighs the change.

To summarise, EU-DaC provides particular implementations of the *divide* and *coordinate* stages as follows. On the one hand, during each *divide* stage, each agent: (1) updates its subproblem by averaging its local utilities with those of its neighbours according to equation 3; and (2) solves the updated subproblem. On the other hand, during the *coordinate* stage, each agent: (1) exchanges its local max-marginal utilities over single variables shared with its neighbours; and (2) assesses its coordination relations using equations 1 and 2.

4 Experiments

We benchmarked EU-DaC against MGM{2,3} algorithms [2] and DaCSA [3]. We compared these algorithms based on the solution obtained at a number of message cycles (mcs) and by plotting the percent gain of EU-DaC with respect to each algorithm. Figures 2 (a)-(c) show the results for 100 agents on small-world, regular grid and random topologies (we use the same experimental scenarios than in [3]). Each graph shows the mean over 25 instances of the percent gain of EU-DaC respect to DaCSA and MGM-{2,3}.

First, observe that in all experimented topologies EU-DaC outperforms DaCSA. Thus, our results show that when agents explicitly communicate their max-marginal utilities instead of their preferred variable assignments they obtain results closer to an agreement. Observe that these gains are higher on structured topologies (small-world and regular grids) than on random ones. Moreover, EU-DaC usually outperforms MGM-2 in all scenarios and the same applies to MGM-3 on structured topologies, though these gains are lower than with respect to DaCSA.

ACKNOWLEDGEMENTS. Work funded by EVE (TIN2009-14702-C02-01), Agreement Technologies (CONSOLIDER CSD2007-0022, INGENIO 2010) and Generalitat de Catalunya (2009-SGR-1434). Meritxell Vinyals is supported by the Ministry of Education of Spain (FPU grant AP2006-04636).

References

[1] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo, 'Adopt: asynchronous distributed constraint optimization with quality guarantees', *Artif. Intell.*, **161**(1-2), 149–180, (2005).

[2] J. P. Pearce, *Local Optimization in Cooperative Agent Networks*, Ph.D. dissertation, Dissertation for Doctor of Philosophy in Computer Science, University of Southern California, Los Angeles, 2007.

[3] Meritxell Vinyals, Marc Pujol, Juan A. Rodríguez-Aguilar, and Jesús Cerquides, 'Divide and Coordinate: solving DCOPs by agreement', in *AAMAS*, pp. 149–156, (2010).