DeQED: An Efficient Divide-and-Coordinate Algorithm for DCOP

Daisuke Hatano * † and **Katsutoshi Hirayama**
Kobe University
5-1-1 Fukaeminami-machi, Higashinada-ku, Kobe 658-0022, Japan
daisuke-hatano@stu.kobe-u.ac.jp, hirayama@maritime.kobe-u.ac.jp

Abstract

This paper presents a new DCOP algorithm called DeQED (Decomposition with Quadratic Encoding to Decentralize). DeQED is based on the Divide-and-Coordinate (DaC) framework, where the agents repeatedly solve their updated local sub-problems (the divide stage) and exchange coordination information that causes them to update their local sub-problems (the coordinate stage). Unlike other DaC-based DCOP algorithms, DeQED does not essentially increase the complexity of local sub-problems and allows agents to avoid exchanging (primal) variable values in the coordinate stage. Our experimental results show that DeQED significantly outperformed other incomplete DCOP algorithms for both random and structured instances.

1 Introduction

In many applications of distributed problem solving, the agents may want to optimize a global objective while preserving their privacy and security. This problem can be formalized as the Distributed Constraint Optimization Problem (DCOP). To solve DCOP, several complete algorithms have been presented in the literature [Modi *et al.*, 2005; Petcu and Faltings, 2005], but one recent trend may be incomplete algorithms [Farinelli *et al.*, 2008; Kiekintveld *et al.*, 2010; Vinyals *et al.*, 2010a; 2010b] due to the need to find high-quality solutions quickly for large-scale problem instances.

This paper presents a new DCOP algorithm called DeQED (Decomposition with Quadratic Encoding to Decentralize). DeQED is based on the Divide-and-Coordinate (DaC) framework, where the agents repeatedly solve their updated local sub-problems (the divide stage) and exchange coordination information that causes them to update their local sub-problems (the coordinate stage). Unlike other DaC-based DCOP algorithms [Vinyals *et al.*, 2010a; 2010b], DeQED does not essentially increase the complexity of local sub-problems and allows agents to avoid exchanging (primal) variable values in the coordinate stage.

*Current Affiliation: National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan

†JST, ERATO, Kawarabayashi Large Graph Project, Japan

Through comparison with MaxSum [Farinelli *et al.*, 2008], DALO [Kiekintveld *et al.*, 2010], and EU-DaC [Vinyals *et al.*, 2010b], we demonstrate that DeQED works very well both in terms of solution quality and efficiency.

The remainder of this paper is organized as follows. We first introduce DCOP and the DaC framework in Section 2, followed by the details of DeQED in Section 3. We experimentally compare DeQED with other incomplete DCOP algorithms in Section 4 and conclude this work in Section 5.

2 DCOP and DaC framework

COP is defined by a set X of *variables*, where each variable x_i has a finite *domain* D_i from which it takes its value, and a set F of *binary cost functions*, where each function $f_{i,j} : D_i \times D_j \rightarrow \Re^+$ returns a non-negative cost value for each binary relation between variable x_i's domain and variable x_j's domain.

DCOP is the COP where variables are controlled by a set A of agents. Each variable belongs to some agent who controls it. We denote the fact that variable x_i belongs to agent a by $\mathrm{belong}(x_i) = a$. The goal of COP and DCOP is to find a value assignment to X that minimizes the total sum of the values of cost functions.

In DCOP, the cost functions are divided into a set F_{inter} of *inter-agent* cost functions and a set F_{intra} of *intra-agent* cost functions. Formally, we define $F_{inter} \equiv \{f_{i,j} \,|\, \mathrm{belong}(x_i) \neq \mathrm{belong}(x_j)\}$ and $F_{intra} \equiv \{f_{i,j} \,|\, \mathrm{belong}(x_i) = \mathrm{belong}(x_j)\}$.

An agent in DCOP may have multiple variables in its control. These multiple variables of an agent can be divided into two sets. One is the set of variables involved in inter-agent cost functions, which we call *interface variables*. A set of interface variables of agent a is defined by $X^a_{inf} \equiv \{x_i \,|\, belong(x_i) = a, \exists x_j (f_{i,j} \in F_{inter}) \vee (f_{j,i} \in F_{inter})\}$. The other is the set of variables that is not involved in any inter-agent cost functions, which we call *hidden variables*. A set of hidden variables of agent a is defined by $X^a_{hid} \equiv \{x_i \,|\, belong(x_i) = a, x_i \notin X^a_{inf}\}$.

DaC (Divide-and-Coordinate) is the framework for solving DCOP, where the agents repeatedly solve their updated local sub-problems (the divide stage) and exchange coordination information that causes them to update their local sub-problems (the coordinate stage). The work in [Vinyals *et* *al.*, 2010a; 2010b] has instantiated this framework as follows. The problem is divided in such a way that the resulting sub-problems share some variables, each of which yields an inter-agent equality constraint among its copies. The value assignments on these copies (interface variables) may be in disagreement with each other when the agents solve their sub-problems independently. Therefore, the agents iterate the following two stages to reach the state where the assignments on every variable's copies are in agreement.

divide stage: Each agent updates its own sub-problem with information received from its neighbors and solves this updated sub-problem.

coordinate stage: Each agent sends information about disagreement on variables to its neighbors.

Both DaCSA [Vinyals *et al.*, 2010a] and EU-DaC [Vinyals *et al.*, 2010b] are DaC-based algorithms. Agents in DaCSA control *Lagrange multipliers*, each of which is defined for any shared variable, while agents in EU-DaC control *coordination parameters*, so that value assignments on every variable's copies are in agreement. It must be noted that, even when solving a DCOP instance with one variable per agent, each agent in DaCSA and EU-DaC has a tree-structured local sub-problem.

3 DeQED

As with DaCSA, DeQED exploits the Lagrangian decomposition technique, but the difference between DeQED and DaCSA is the way of encoding of the entire problem. In DeQED, we use *quadratic encoding*, in which an inter-agent cost function is encoded into the quadratic programming problem. Let us start this section by presenting the details of the quadratic encoding.

3.1 Quadratic encoding

Let us assume that every variable has the same domain, say D, without loss of generality. For cost function $f_{i,j} \in F$ between variable x_i and variable x_j, we introduce the following $|D| \times |D|$ cost matrix:

$$\boldsymbol{F}_{i,j} = \begin{pmatrix} c^{i,j}_{1,1} & c^{i,j}_{1,2} & \cdots & c^{i,j}_{1,|D|} \\ c^{i,j}_{2,1} & c^{i,j}_{2,2} & \cdots & c^{i,j}_{2,|D|} \\ \vdots & \vdots & \ddots & \vdots \\ c^{i,j}_{|D|,1} & c^{i,j}_{|D|,2} & \cdots & c^{i,j}_{|D|,|D|} \end{pmatrix},$$

whose element $c^{i,j}_{m,n}$ represents the cost when we assign variable x_i the mth value of domain and variable x_j the nth value of domain. Furthermore, for variables x_i and x_j, we introduce new variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ respectively, whose domains are the whole set of $|D|$-dimensional unit column vectors. Namely, we have $\boldsymbol{x}_i \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{|D|}\}$ and $\boldsymbol{x}_j \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{|D|}\}$, where $\boldsymbol{e}_1$ is $(1, 0, 0, \ldots, 0)^{\mathrm{T}}$, $\boldsymbol{e}_2$ is $(0, 1, 0, \ldots, 0)^{\mathrm{T}}$, and so on. Superscript T means the transpose of a vector.

Given this representation, the value of cost function $f_{i,j}$ can be computed by $(\boldsymbol{x}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{x}_j$. For example, when x_i takes the mth value and x_j takes the nth value, the cost is computed by $(\boldsymbol{e}_m)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{e}_n$, resulting in the (m, n)-element of $\boldsymbol{F}_{i,j}$.

Therefore, DCOP can be formulated as

$$\begin{aligned} &\mathcal{DCOP}\ : \\ &\min_{\boldsymbol{x}} \quad \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{x}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{x}_j + \sum_{f_{i,j} \in F_{intra}} (\boldsymbol{x}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{x}_j \\ &\text{s. t.} \quad \boldsymbol{x}_i, \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D|}\}, \quad \forall x_i \in X. \end{aligned}$$

Since the intra-agent cost functions are partitioned among the agents, we can put the intra-agent cost functions on agent a together to produce one black-box function φ^a that returns, given an assignment to the variables, the sum of the values of a's intra-agent cost functions. With this black-box function, we can reformulate DCOP as follows:

$$\begin{aligned} \mathcal{DCOP}\ : \min_{\boldsymbol{x}} \quad & \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{x}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{x}_j + \sum_{a \in A} \varphi^a(X) \\ \text{s. t.} \quad & \boldsymbol{x}_i, \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D|}\}, \quad \forall x_i \in X. \end{aligned}$$

We introduce two auxiliary variables $\boldsymbol{\alpha}^{i,j}_i$ and $\boldsymbol{\alpha}^{i,j}_j$ for each inter-agent cost function $f_{i,j}$. These auxiliary variables are supposed to be the copies of interface variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in terms of $f_{i,j}$, respectively. Hence, we have an equivalent description of the above DCOP as follows:

$$\begin{aligned} &\mathcal{DCOP}'\ : \\ &\min_{\boldsymbol{x},\boldsymbol{\alpha}} \quad \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{\alpha}^{i,j}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{\alpha}^{i,j}_j + \sum_{a \in A} \varphi^a(X) \\ &\text{s. t.} \quad \boldsymbol{x}_i = \boldsymbol{\alpha}^{i,j}_i,\ \boldsymbol{x}_j = \boldsymbol{\alpha}^{i,j}_j, \quad \forall f_{i,j} \in F_{inter}, \qquad (1) \\ &\qquad \boldsymbol{x}_i, \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D|}\}, \quad \forall x_i \in X, \\ &\qquad \boldsymbol{\alpha}^{i,j}_i, \boldsymbol{\alpha}^{i,j}_j \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D|}\}, \quad \forall f_{i,j} \in F_{inter}, \end{aligned}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{x}$ are decision variables. Due to space limitations, we omit the last two lines of the above formulation because they just describe the domain of these decision variables.

3.2 Lagrangian Decomposition

We decompose this problem into the sub-problems over the agents. First, we relax a set of copy constraints (1) to produce the *Lagrangian relaxation problem*:

$$\begin{aligned} \mathcal{L}\ :\ L(\boldsymbol{\mu}) \ \equiv\ \min_{\boldsymbol{x},\boldsymbol{\alpha}} & \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{\alpha}^{i,j}_i)^{\mathrm{T}} \cdot \boldsymbol{F}_{i,j} \cdot \boldsymbol{\alpha}^{i,j}_j + \sum_{a \in A} \varphi^a(X) \\ & + \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{\mu}^{i,j}_i)^{\mathrm{T}} (\boldsymbol{x}_i - \boldsymbol{\alpha}^{i,j}_i) \\ & + \sum_{f_{i,j} \in F_{inter}} (\boldsymbol{\mu}^{i,j}_j)^{\mathrm{T}} (\boldsymbol{x}_j - \boldsymbol{\alpha}^{i,j}_j), \end{aligned}$$

where both $\boldsymbol{\mu}^{i,j}_i$ and $\boldsymbol{\mu}^{i,j}_j$ are $|D|$-dimensional real-valued column vectors and called Lagrange multiplier vectors. For any fixed values for $\boldsymbol{\mu}$, the optimal value of $\mathcal{L}$, denoted by $L(\boldsymbol{\mu})$, provides a lower bound on the optimal value of $\mathcal{DCOP}'$.

Then, we decompose the objective function of $\mathcal{L}$ into the terms on the individual agents and the terms on auxiliary variables. As a result, we get the sub-problem on the agents:

$$
\begin{aligned}
&\mathcal{L}^{agn} \;:\; L^{agn}(\boldsymbol{\mu}) \\
&\equiv \min_{\boldsymbol{x}} \sum_{a\in A}\left\{\varphi^a(X) + \sum_{(x_i,x_j)\in P^a} (\boldsymbol{\mu}_i^{i,j})^{\mathrm{T}}\boldsymbol{x}_i + \sum_{(x_i,x_j)\in N^a} (\boldsymbol{\mu}_j^{i,j})^{\mathrm{T}}\boldsymbol{x}_j\right\}, \\
&= \sum_{a\in A} \min_{\boldsymbol{x}} \left\{\varphi^a(X) + \sum_{(x_i,x_j)\in P^a} (\boldsymbol{\mu}_i^{i,j})^{\mathrm{T}}\boldsymbol{x}_i + \sum_{(x_i,x_j)\in N^a} (\boldsymbol{\mu}_j^{i,j})^{\mathrm{T}}\boldsymbol{x}_j\right\}, \\
&\equiv \sum_{a\in A} L^a(\boldsymbol{\mu}),
\end{aligned}
$$

where $P^a \equiv \{(x_i,x_j) \,|\, f_{i,j} \in F_{inter}, \mathrm{belong}(x_i) = a\}$ and $N^a \equiv \{(x_i,x_j) \,|\, f_{i,j} \in F_{inter}, \mathrm{belong}(x_j) = a\}$. We also get the sub-problem on auxiliary variables:

$$
\begin{aligned}
&\mathcal{L}^{aux} \;:\; L^{aux}(\boldsymbol{\mu}) \\
&\equiv \min_{\boldsymbol{\alpha}} \sum_{f_{i,j}\in F_{inter}} \left\{(\boldsymbol{\alpha}_i^{i,j})^T \boldsymbol{F}_{i,j}\boldsymbol{\alpha}_j^{j,i} - (\boldsymbol{\mu}_i^{i,j})^{\mathrm{T}}\boldsymbol{\alpha}_i^{i,j} - (\boldsymbol{\mu}_j^{i,j})^{\mathrm{T}}\boldsymbol{\alpha}_j^{i,j}\right\}. \\
&= \sum_{f_{i,j}\in F_{inter}} \min_{\boldsymbol{\alpha}} \left\{(\boldsymbol{\alpha}_i^{i,j})^T \boldsymbol{F}_{i,j}\boldsymbol{\alpha}_j^{i,j} - (\boldsymbol{\mu}_i^{i,j})^{\mathrm{T}}\boldsymbol{\alpha}_i^{i,j} - (\boldsymbol{\mu}_j^{i,j})^{\mathrm{T}}\boldsymbol{\alpha}_j^{i,j}\right\}. \\
&\equiv \sum_{f_{i,j}\in F_{inter}} L_{i,j}^{aux}(\boldsymbol{\mu}).
\end{aligned}
$$

Note that, except for $\boldsymbol{\mu}$, these sub-problems do not share any decision variables; even the sub-problems on agents do not share their decision variables.

Given fixed values for $\boldsymbol{\mu}$, the sub-problem giving $L^a(\boldsymbol{\mu})$ of each agent a can be viewed as the Weighted Constraint Satisfaction Problem (WCSP). For example, the term of $(\boldsymbol{\mu}_i^{i,j})^{\mathrm{T}}\boldsymbol{x}_i$ is a unary soft constraint on variable x_i whose weights for domain values are $\boldsymbol{\mu}_i^{i,j}$. Furthermore, $\varphi^a(X)$ is actually a set of binary soft constraints. On the other hand, it is trivial to solve the sub-problem giving $L_{i,j}^{aux}(\boldsymbol{\mu})$ of each inter-agent cost function $f_{i,j}$ because we just select an optimal pair of values for the cost matrix that is modified with $\boldsymbol{\mu}$.

To summarize, we get the following Lagrangian decomposition:

$$
L(\boldsymbol{\mu}) = \sum_{a\in A} L^a(\boldsymbol{\mu}) + \sum_{f_{i,j}\in F_{inter}} L_{i,j}^{aux}(\boldsymbol{\mu}).
$$

3.3 Lagrangian Dual

The goal of the *Lagrangian dual problem* is to maximize the lower bound that is obtained by solving the Lagrangian relaxation problem by controlling values of $\boldsymbol{\mu}$. With the above decomposition, it is formally defined by

$$
\mathcal{D} \;:\; \max_{\boldsymbol{\mu}} \quad \sum_{a\in A} L^a(\boldsymbol{\mu}) + \sum_{f_{i,j}\in F_{inter}} L_{i,j}^{aux}(\boldsymbol{\mu}). \qquad (2)
$$

Clearly, the value of the objective function of $\mathcal{D}$ provides a lower bound on the optimal value of $\mathcal{DCOP}'$.

DeQED solves this decomposed Lagrangian dual problem by searching for the values of $\boldsymbol{\mu}$ that give the highest lower bound on the optimal value of $\mathcal{DCOP}'$.

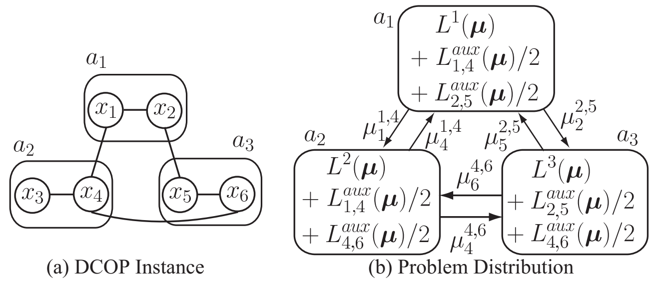

Figure 1: Example of problem distribution

3.4 Problem Distribution

We need to clarify which agent should compute which part of (2). Regarding the *primal phase*, where we solve the minimization problem over $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ with specific values on $\boldsymbol{\mu}$, we propose that

- $L^a(\boldsymbol{\mu})$ should be computed by agent a since it includes only agent a's variables;
- $L_{i,j}^{aux}(\boldsymbol{\mu})$ should be computed by either of the agents who control variables $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$ since it represents inter-agent cost function $f_{i,j}$ between these agents.

On the other hand, regarding the *dual phase*, where we solve the maximization problem over $\boldsymbol{\mu}$ with specific values on $\boldsymbol{x}$ and $\boldsymbol{\alpha}$, we propose that

- since Lagrange multiplier vectors $\boldsymbol{\mu}_i^{i,j}$ and $\boldsymbol{\mu}_j^{i,j}$ are related to inter-agent cost function $f_{i,j}$, both vectors should be controlled by the agents having variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively.

Let us present an example. In Figure 1 (a), nodes are variables, edges are binary cost functions, and boxes are agents. The Lagrangian dual problem of this example consists of the following:

- $X = \{x_1, x_2, \ldots, x_6\}$, $A = \{1, 2, 3\}$,
- $F_{inter} = \{f_{1,4}, f_{2,5}, f_{4,6}\}$, $F_{intra} = \{f_{1,2}, f_{3,4}, f_{5,6}\}$,
- $P^1 = \{(x_1,x_4), (x_2,x_5)\},\ N^1 = \{\}$,
- $P^2 = \{(x_4,x_6)\},\ N^2 = \{(x_1,x_4)\}$,
- $P^3 = \{\},\ N^3 = \{(x_2,x_5), (x_4,x_6)\}$,
- $\varphi^1(X)$ returns the objective of COP $: \{(x_1,x_2), (f_{1,2})\}$,
- $\varphi^2(X)$ returns the objective of COP $: \{(x_3,x_4), (f_{3,4})\}$,
- $\varphi^3(X)$ returns the objective of COP $: \{(x_5,x_6), (f_{5,6})\}$,
- $L^1(\boldsymbol{\mu}) = \min\,\{\varphi^1(X) + (\boldsymbol{\mu}_1^{1,4})^{\mathrm{T}}\boldsymbol{x}_1 + (\boldsymbol{\mu}_2^{2,5})^{\mathrm{T}}\boldsymbol{x}_2\}$,
- $L^2(\boldsymbol{\mu}) = \min\,\{\varphi^2(X) + (\boldsymbol{\mu}_4^{4,6})^{\mathrm{T}}\boldsymbol{x}_4 + (\boldsymbol{\mu}_4^{1,4})^{\mathrm{T}}\boldsymbol{x}_4\}$,
- $L^3(\boldsymbol{\mu}) = \min\,\{\varphi^3(X) + (\boldsymbol{\mu}_5^{2,5})^{\mathrm{T}}\boldsymbol{x}_5 + (\boldsymbol{\mu}_6^{4,6})^{\mathrm{T}}\boldsymbol{x}_6\}$,
- $L_{1,4}^{aux}(\boldsymbol{\mu}) = \min\,\{(\boldsymbol{\alpha}_1^{1,4})^T \cdot \boldsymbol{F}_{1,4} \cdot \boldsymbol{\alpha}_4^{1,4} - (\boldsymbol{\mu}_1^{1,4})^{\mathrm{T}}\boldsymbol{\alpha}_1^{1,4} - (\boldsymbol{\mu}_4^{1,4})^{\mathrm{T}}\boldsymbol{\alpha}_4^{1,4}\}$,

- $L_{2,5}^{aux}(\boldsymbol{\mu}) = \min\{(\boldsymbol{\alpha}_2^{2,5})^T \cdot \boldsymbol{F}_{2,5} \cdot \boldsymbol{\alpha}_5^{2,5} - (\boldsymbol{\mu}_2^{2,5})^{\mathrm{T}}\boldsymbol{\alpha}_2^{2,5} - (\boldsymbol{\mu}_5^{2,5})^{\mathrm{T}}\boldsymbol{\alpha}_5^{2,5}\}$,
- $L_{4,6}^{aux}(\boldsymbol{\mu}) = \min\{(\boldsymbol{\alpha}_4^{4,6})^T \cdot \boldsymbol{F}_{4,6} \cdot \boldsymbol{\alpha}_6^{4,6} - (\boldsymbol{\mu}_4^{4,6})^{\mathrm{T}}\boldsymbol{\alpha}_4^{4,6} - (\boldsymbol{\mu}_6^{4,6})^{\mathrm{T}}\boldsymbol{\alpha}_6^{4,6}\}$.

We propose to distribute these components over the agents as depicted in Figure 1 (b), where

- Agent 1 computes $L^1(\boldsymbol{\mu})$, $L_{1,4}^{aux}(\boldsymbol{\mu})/2$, and $L_{2,5}^{aux}(\boldsymbol{\mu})/2$ while controlling $\boldsymbol{\mu}_1^{1,4}$ and $\boldsymbol{\mu}_2^{2,5}$. Note that, in computing $L_{1,4}^{aux}(\boldsymbol{\mu})/2$ and $L_{2,5}^{aux}(\boldsymbol{\mu})/2$, it needs $\boldsymbol{\mu}_4^{1,4}$ and $\boldsymbol{\mu}_5^{2,5}$, which are controlled by the other agents.
- Agent 2 computes $L^2(\boldsymbol{\mu})$, $L_{1,4}^{aux}(\boldsymbol{\mu})/2$, and $L_{4,6}^{aux}(\boldsymbol{\mu})/2$ while controlling $\boldsymbol{\mu}_4^{1,4}$ and $\boldsymbol{\mu}_4^{4,6}$. Note that, in computing $L_{1,4}^{aux}(\boldsymbol{\mu})/2$ and $L_{4,6}^{aux}(\boldsymbol{\mu})/2$, it needs $\boldsymbol{\mu}_1^{1,4}$ and $\boldsymbol{\mu}_6^{4,6}$, which are controlled by the other agents.
- Agent 3 computes $L^3(\boldsymbol{\mu})$, $L_{2,5}^{aux}(\boldsymbol{\mu})/2$, and $L_{4,6}^{aux}(\boldsymbol{\mu})/2$ while controlling $\boldsymbol{\mu}_5^{2,5}$ and $\boldsymbol{\mu}_6^{4,6}$. Note that, in computing $L_{2,5}^{aux}(\boldsymbol{\mu})/2$ and $L_{4,6}^{aux}(\boldsymbol{\mu})/2$, it needs $\boldsymbol{\mu}_2^{2,5}$ and $\boldsymbol{\mu}_4^{4,6}$, which are controlled by the other agents.

We should emphasize that, given values of $\boldsymbol{\mu}$, $L^a(\boldsymbol{\mu})$ for agent a is a WCSP on variables $\boldsymbol{x}$, and $L_{i,j}^{aux}(\boldsymbol{\mu})$ for inter-agent cost function $f_{i,j}$ is a trivial problem on auxiliary variables $\boldsymbol{\alpha}$. In the above, each $L_{i,j}^{aux}(\boldsymbol{\mu})$ is divided by two and shared between a pair of agents. This is due to making the total sum of $L^a(\boldsymbol{\mu})$ over the agents and $L_{i,j}^{aux}(\boldsymbol{\mu})$ over the inter-agent cost functions equal to the objective of (2).

3.5 Minimal Procedure

Below is the minimal procedure of DeQED, where the agents try to find values for $\boldsymbol{\mu}$ that minimize the objective of (2).

Step 1: The agents initialize their $\boldsymbol{\mu}$ as $(0, \ldots, 0)^{\mathrm{T}}$.

Step 2: Every agent a sends, for each inter-agent cost function $f_{i,j}$ with $\mathrm{belong}(i) = a$, the value of $\boldsymbol{\mu}_i^{i,j}$ to the agent to which x_j belongs. Similarly, it sends, for each inter-agent cost function $f_{i,j}$ with $\mathrm{belong}(j) = a$, the value of $\boldsymbol{\mu}_j^{i,j}$ to the agent to which x_i belongs.

Step 3: After receiving all of the latest values for $\boldsymbol{\mu}$, every agent a solves $L^a(\boldsymbol{\mu})$ by an exact WCSP solver and $L_{i,j}^{aux}(\boldsymbol{\mu})$ by evaluating all possible pairs of the values for x_i and x_j.

Step 4: If *CanTerminate?* then the agents stop; otherwise they update $\boldsymbol{\mu}$ and go back to Step 2.

We refer to one iteration from Steps 2 to 4 as a *cycle*.

Next, we focus on Step 4, giving the details of what conditions cause the agents to stop and how to update Lagrange multiplier vector $\boldsymbol{\mu}$ in every cycle. Before doing so, however, it is worth mentioning how the agents can *explicitly* compute upper and lower bounds on the global optimal values of the original DCOP.

3.6 Computing Upper and Lower Bounds

DeQED can work in its minimal form. In that case, if it is successfully terminated, a final assignment for the variables is guaranteed to be globally optimal. However, if it terminates due to a time limit, agents in the minimal DeQED get only an assignment for their variables with no extra information. If the agents require a lower or upper bound on the global optimal, they need to explicitly communicate those bounds.

Remember that the value of the objective of (2) gives a lower bound on the global optimal. Thus, as with the method in [Hirayama *et al.*, 2009], agents can compute a lower bound by explicitly collecting all of the values of $L^a(\boldsymbol{\mu})$ and $L_{i,j}^{aux}(\boldsymbol{\mu})$ over the agents at a certain cycle by using a *spanning tree*. Note that, by this method, agents need to exchange the objective values of sub-problems but do not need to exchange an assignment on interface variables.

On the other hand, if we allow agents to exchange an assignment on interface variables, they can compute their pieces of the global objective. By collecting those pieces using a spanning tree, agents can compute an upper bound on the global optimal.

These upper and lower bounds can be computed on-line in every cycle by overlaying a distributed data-collection protocol on the minimal DeQED. Furthermore, as we shall see later, the agents in DeQED can exploit the best bounds, *BestUB* and *BestLB*, among those computed in deciding when to terminate the procedure and how to update the μ. We refer to this extended version of DeQED as DeQED$_a$ and to the minimal version of DeQED as DeQED$_m$.

It is noteworthy that the agents in DeQED$_m$ only exchange dual variables. Namely, they do not have to exchange values of interface variables. Considering that one major motivation of DCOP is privacy and security, this property of DeQED$_m$ should be important.

3.7 Termination

At Step 4 in the procedure, we have the following three criteria to make the agents terminate themselves.

satisfying relaxed constraints Since we have relaxed equality constraints (1) of $\mathcal{DCOP}'$, we can ensure that if the solutions to the sub-problems at Step 3 happen to satisfy all of the relaxed constraints, then these solutions constitute an optimal solution to DCOP. To detect the fact that all of the relaxed constraints are satisfied, the agents need to take a snapshot of the system, which can be done by using a spanning tree.

achieving a quality bound When the agents compute *BestUB* and *BestLB* as in Section 3.6, they can terminate themselves if $BestUB/BestLB$ becomes no more than a specified quality bound. The agents end with a global optimal solution if the quality bound is set to one.

exceeding a time limit Obviously, the agents terminate themselves whenever they use up the time allowed.

Clearly, DeQED$_a$ terminates with any of these three criteria while DeQED$_m$ does with the first or third one.

3.8 Updating Lagrange Multipliers

If none of the above criteria is met, the agents update their own $\boldsymbol{\mu}$ aiming at a tighter (higher) lower bound on the optimal value of DCOP. This involves a search algorithm for the

Lagrangian dual problem. We solve this problem with the *sub-gradient ascent* method [Bertsekas, 1999]. Here are the details of this method.

For each inter-agent cost function $f_{i,j}$,

1. the agents with x_i and x_j compute *sub-gradient* $\boldsymbol{G}_i^{i,j}$ and $\boldsymbol{G}_j^{i,j}$ as:

$$\boldsymbol{G}_i^{i,j} \equiv \boldsymbol{x}_i - \boldsymbol{\alpha}_i^{i,j}, \quad \boldsymbol{G}_j^{i,j} \equiv \boldsymbol{x}_j - \boldsymbol{\alpha}_j^{i,j},$$

which correspond to the coefficients of $\boldsymbol{\mu}_i^{i,j}$ and $\boldsymbol{\mu}_j^{i,j}$, respectively, in the objective function of $\mathcal{L}$.

2. they update $\boldsymbol{\mu}_i^{i,j}$ and $\boldsymbol{\mu}_j^{i,j}$ as

$$\boldsymbol{\mu}_i^{i,j} \leftarrow \boldsymbol{\mu}_i^{i,j} + D \cdot \boldsymbol{G}_i^{i,j}, \quad \boldsymbol{\mu}_j^{i,j} \leftarrow \boldsymbol{\mu}_j^{i,j} + D \cdot \boldsymbol{G}_j^{i,j},$$

where D is a scalar parameter, called *step length*, which is computed differently depending on the implementations of algorithm.

This method implies that an agent increases (decreases) the value of the Lagrange multiplier if its corresponding coefficient in the objective function of $\mathcal{L}$ is positive (negative), hoping that $L(\boldsymbol{\mu})$, a lower bound on the optimal value of DCOP, increases in the next cycle.

To compute step length, DeQED$_m$ follows a simple diminishing strategy where starting from a certain value of D (e.g. 10% of the possible maximum cost), we gradually reduce it by half after keeping its value for a fixed number of cycles (e.g. 10 cycles). Note that DeQED$_m$ should require some tuning process to find a valid schedule for diminishing step length.

On the other hand, DeQED$_a$ follows the conventional strategy, where we compute step length by

$$D \equiv \frac{\pi(BestUB - BestLB)}{\sum_{f_{i,j} \in F_{inter}} \{(\boldsymbol{G}_i^{i,j})^T \cdot \boldsymbol{G}_i^{i,j} + (\boldsymbol{G}_j^{i,j})^T \cdot \boldsymbol{G}_j^{i,j}\}}, \quad (3)$$

in which π is a scalar parameter that, starting from its initial value of two, is reduced by half when *BestLB* is not updated for a certain consecutive number of cycles (e.g. five cycles).

Although the sub-gradient ascent method is quite simple, it does not necessarily converge to an optimal solution to DCOP. Thus, both DeQED$_m$ and DeQED$_a$ are incomplete.

4 Experiments

We compared DeQED with DALO [Kiekintveld *et al.*, 2010], EU-DaC [Vinyals *et al.*, 2010b], and MaxSum [Farinelli *et al.*, 2008] on binary constraint networks with *random*, *regular grid*, and *scale-free* topologies. We did not adopt DaCSA since it was outperformed by EU-DaC [Vinyals *et al.*, 2010b].

Details on how to generate instances for each topology are as follows.

random : We created an n-node network whose density is the ratio of d, resulting in $\lfloor n(n-1) * d \rfloor$ edges in total.

regular grid : We created an n-node network arranged in a rectangular grid, where each node is connected to four neighboring nodes (except when it is located on the boundary).

scale-free : We created an n-node network based on the Barabasi-Albert (BA) model, where starting from two nodes with an edge, we added a node one-by-one while randomly connecting the added node to two existing nodes by new edges. Such two nodes are selected with probabilities that are proportional to the numbers of their connected edges. The total number of edges is $2(n-2)+1$.

We created 20 instances for each of these three topologies. For each instance of these networks, we ascertained its connectivity. Namely, there is no disconnected sub-network in every instance.

We created a COP instance for each instance of networks, where the domain size of all variables (nodes) is three and the cost value of binary cost functions (edges) is randomly selected from $\{1, 2, ..., 10^5\}$. Then, following the convention in the literature, we created a DCOP instance so that one agent has exactly one variable and its related cost functions.

Our experiments were conducted on a discrete event simulator that simulates concurrent activities of multiple agents using the cycle-base mechanism, where the agents repeat the cycle of receiving messages, perform local computations, and sending messages until a termination condition is met.

To evaluate the performance of each algorithm, we measured the number of *cycles*, *simulated runtime* [Sultanik *et al.*, 2007] and *quality upper bound* on this simulator. The number of cycles is one of the conventional measures to evaluate the performance of the DisCSP or DCOP algorithm, while the simulated runtime is a relatively new measure, which corresponds to the longest sequence of runtime on the agents. For each algorithm, the quality upper bound is computed by dividing the obtained global cost by BestLB that is computed by DeQED$_a$. Note that the obtained global cost is the one at a cutoff cycle for DeQED$_m$ while BestUB (the best upper bound found by a cutoff cycle) for other algorithms including DeQED$_a$. On the quality upper bound, a figure closer to one is better.

Since all of the algorithms are incomplete, our interest is on how quickly each of these algorithms finds a better solution. Therefore, in our experiments, we observed an average quality upper bound for each algorithm when cutting off a run at a certain cycle bound, which ranges from 50 to 500 cycles in steps of 50. Furthermore, since these algorithms clearly have different computational costs in one cycle, we also observed an average quality upper bound against simulated runtime at the above cut-off cycles.

These experiments were conducted on an Intel Core-i7 2600@3.4GHz with 4 Cores, 8 threads and 8 GB memory. The main codes of DeQED and EU-DaC were written in Java and compiled with JDK 1.6.0-20 on Ubuntu 11.10 (64 bit). We downloaded DALO from the USC DCOP Repository and used the DALO-t with $t = 1$. On the MaxSum algorithm, we used the code in the FRODO version 2.10.5 [Léauté *et al.*, 2009] with a default setting.

The results are shown in Figure 2, where the left part denoted by (a) shows the average quality upper bound against the number of cycles and the right part denoted by (b) shows the average quality upper bound against simulated runtime.

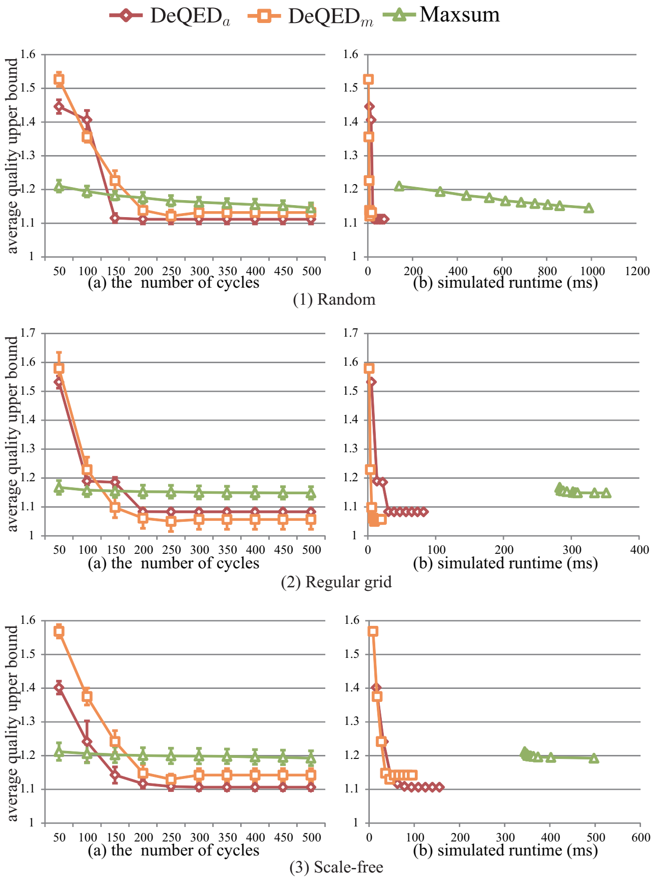

Figure 2: Average quality upper bounds for (1) 1000-node random networks with density 0.005, (2)1000-node regular grid networks and (3) 1000-node scale-free networks.

In these figures, we omit the results of DALO and EU-DaC because their quality upper bounds were more than 1.4, even in the best case. Moreover, EU-DaC spent more than 10,000 ms of the simulated runtime to finish 500 cycles. Thus, we plot the average quality upper bounds with error bars only for MaxSum, DeQED$_m$ and DeQED$_a$ for readability.

Figure 2 shows that DeQED clearly outperformed MaxSum for any topology of networks. In particular, DeQED converged quite efficiently in the simulated runtime. One reason for this efficiency is that the computational cost of each agent in DeQED increases only linearly with the number of its neighbors, while that in MaxSum increases exponentially [Farinelli *et al.*, 2008].

When comparing DeQED$_m$ and DeQED$_a$, we observed that DeQED$_a$ generally shows faster convergence than DeQED$_m$ with respect to the number of cycles, but DeQED$_m$ is slightly faster than DeQED$_a$ with respect to simulated runtime. The reason behind this difference is that the agents in DeQED$_a$ are required to collect and compute upper and lower bounds on-line, which imposes extra computation on the agents. On the other hand, the agents in DeQED$_m$ do not have to perform such extra computation.

We should also point out that EU-DaC, another algorithm

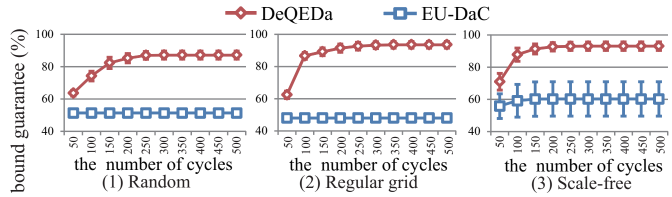

Figure 3: Bound guarantees of DeQED$_a$ and EU-DaC for (1) 100-node random networks with density 0.05, (2)100-node regular grid networks and (3) 100-node scale-free networks

based on the DaC framework, did not perform well especially in simulated runtime. An agent in EU-DaC solves the local problem that includes the copies of variables of neighboring agents and tries to make an agreement on the assignments between their copies and originals. Thus, even for the DCOP with one variable per agent, EU-DaC has to solve much larger local problems, which clearly increases the cost of local computations. On the other hand, in DeQED, the local problem of each agent is still trivial for the DCOP with one variable per agent.

Moreover, we also compared DeQED$_a$ and EU-DaC with respect to the tightness of their bounds as the cycle proceeds. Similar to the experiments in [Vinyals *et al.*, 2010b], we measured *bound guarantee* at a certain number c of cycles for each algorithm A, which is defined by

$$bg_A(c) \equiv BestLB_A(c)/BestUB_A(c) \times 100,$$

where $BestLB_A(c)$ is the highest lower bound that algorithm A has found by the cycle of c, and $BestUB_A(c)$ is the lowest upper bound that algorithm A has found by the cycle of c. The results are shown in Figure 3, where the x-axis indicates the number of cycles and the y-axis indicates the bound guarantees averaged over 20 instances. From these results, we can say that DeQED$_a$ can always obtain tighter bounds than EU-DaC.

5 Conclusions

We provided a new DCOP algorithm called DeQED (Decomposition with Quadratic Encoding to Decentralize). The contribution of DeQED is twofold.

First, DeQED does not essentially increase the complexity of local subproblems. The previous DaC-based algorithms, DaCSA and EU-DaC, solve the local problem that includes the copies of variables of neighboring agents, which clearly increases the cost of local computations. However, in DeQED, the local problem of each agent is trivial.

Second, it allows agents to avoid exchanging (primal) variable values in the coordinate stage. The agents in DaCSA and EU-DaC need to exchange (primal) variable values. However, the agents in DeQED can exchange only the values of Lagrange multipliers.

Furthermore, we experimentally confirmed that DeQED worked significantly better than other representative incomplete DCOP algorithms.

References

[Bertsekas, 1999] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[Farinelli *et al.*, 2008] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, *(AAMAS-2008)*, pages 639–646, 2008.

[Hirayama *et al.*, 2009] Katsutoshi Hirayama, Toshihiro Matsui, and Makoto Yokoo. Adaptive price update in distributed Lagrangian relaxation protocol. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, *(AAMAS-2009)*, pages 1033–1040, 2009.

[Kiekintveld *et al.*, 2010] Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, *(AAMAS-2010)*, pages 133–140, 2010.

[Léauté *et al.*, 2009] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the IJCAI-09 Distributed Constraint Reasoning Workshop*, pages 160–164, 2009. http://liawww.epfl.ch/frodo/.

[Modi *et al.*, 2005] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[Petcu and Faltings, 2005] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, *(IJCAI-2005)*, pages 266–271, 2005.

[Sultanik *et al.*, 2007] Evan A. Sultanik, Robert N. Lass, and William C. Regli. Dcopolis: A framework for simulating and deploying distributed constraint reasoning algorithms. In *Proceedings of the Ninth International Workshop on Distributed Constraint Reasoning*, pages 1667–1668, 2007.

[Vinyals *et al.*, 2010a] Meritxell Vinyals, Marc Pujol, Juan Antonio Rodriguez-Aguilar, and Jesus Cerquides. Divide-and-coordinate: DCOPs by agreement. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, *(AAMAS-2010)*, pages 149–156, 2010.

[Vinyals *et al.*, 2010b] Meritxell Vinyals, Juan Antonio Rodriguez-Aguilar, and Jesus Cerquides. Divide-and-coordinate by egalitarian utilities: Turning DCOPs into egalitarian worlds. In *Proceedings of the 3rd International Workshop on Optimization in Multi-Agent Systems*, 2010.