1E3-OS-4-8

DeQED: 双対変数の値を交換する分散制約最適化アルゴリズム

DeQED: DCOP Solving by Exchanging Dual Variables

波多野大督　　平山勝敏
Daisuke HATANO　Katsutoshi HIRAYAMA

神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University

DCOP-CLP is a Distributed Constraint Optimization Problem with complex local problems, which is an expansion of the Distributed Constraint Optimization Problem (DCOPs). In this problem, each agent is faced with complex decision-making, like solving a Constraint Optimization Problem (COP), and must make decisions to optimize global objective. For DCOP-CLP, we provide an algorithm, called DeQED (Decomposition with Quadratic Encoding to Decentralize), based on a Lagrangian decomposition technique that divides the entire problem into sub-problems, each of which can be completed using an exact weighted constraint satisfaction problem solver. One notable feature of this algorithm is that it produces both upper and lower bounds on the global optimal without exchanging such agent primal information as values and constraints. Moreover, our experimental results show that our solver outperforms other algorithms for both DCOPs and DCOP-CLP and provides empirically good solutions for large-scale problems.

1. はじめに

センサーネットワークや　会議日程調整問題のように，多くのマルチエージェントシステムでは各々のエージェントがある制限下において協調的に目的を達成するように意思決定を行う．分散制約最適化問題 (Distributed Constraint Optimization Problem, DCOP) はこのようなマルチエージェントシステムに対する基本的な枠組みである．近年の DCOP アルゴリズムは大規模な問題に対して，良質な解を高速に求めることを目的とする傾向がある．Maxsum や DALO，DaCSA などが代表的な発見的解法である．これらの解法は基本的にエージェントが 1 つの変数を持つ問題に対して研究されている．しかし，例えば，組織内のあるメンバ間で会議の日程調整をする必要がある場合，もしメンバ間にプライバシーが存在しなければ，メンバ間の日程は 1 つのグループとして調整することができる．組織間での会議日程の調整などのように，エージェントを 1 つのまとまりとして扱える場合は容易に考えられる．このようなエージェントが 1 つの局所問題を持つ DCOP を複雑な局所問題を伴う DCOP (DCOP with Complex Local Problems, DCOP-CLP) と呼ぶ．既存の DCOP ソルバーの中では DaCSA が DCOP-CLP を解くことができる．DaCSA[Vinyals 10] は初めてのラグランジュ分解法に基づいたソルバーで，上界と下界を求められる点が 1 つの特徴である．DaCSA の分解過程では，全体の問題を整数計画問題に変換し，緩和することで緩和問題を生成しているが，整数計画問題への変換時に余分な変数を生成してしまい，大規模な問題を解くことが困難になる可能性がある．

本提案ソルバーである DeQED (Decomposition with Quadratic Encoding to Decentralize) は DaCSA と同様，ラグランジュ分解法に基づいたソルバーで，全体の問題を部分問題に分解する．DaCSA との違いは変換方法である．DeQED では，ラグランジュ分解を使用するために全体の問題を 2 次計画問題に変換する．この変換方法では，制約にかかるコストを行列，変数が取りうる値をベクトルとして表現する．特徴は，エージェントと主要な情報を通信することなく最適解に対する上界と下界を求めることができる点である．

連絡先: 波多野大督，daisuke-hatano@stu.kobe-u.ac.jp,
神戸大学大学院海事科学研究科海事科学専攻,
〒 658-0022　神戸市東灘区深江南町 5 － 1 － 1

2. 分散制約最適化問題

DCOP は変数集合 $X = \{x_1, x_2, ..., x_n\}$ と値域集合 $D = \{D_1, D_2, ..., D_n\}$，コスト関数集合 $F = \{f_1, f_2, ..., f_m\}$ で定義される．X のある要素 x_s はエージェント s が持つ変数を表し，その取りうる値は値域 D の要素 D_s から選択される．F の要素である $f_s : D_{s1} \times D_{s2} \rightarrow \mathbb{R}^+$ はエージェント $s1$ と $s2$ の間のコスト関数で，非負実数値を返す．分散制約最適化の目的は全体のコストが最小となる割当を探索することであるが，各エージェントは自身の変数のみにしか値を割り当てられないため，協調的に割当を決定する必要がある．

一方，DCOP-CLP は以下のように定義される．

定義 1 (複雑な局所問題を伴う分散制約最適化問題) *DCOP-CLP* は *COP* の集合 $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$ とエージェント間コスト関数の集合 F で定義される．集合 Φ のある要素 ϕ_s は 3 つ組 $\{X^i, D^i, F^i\}$ で定義される．集合 F のある要素 $f^{i,j}_{k,l} : D^i_k \times D^j_l \rightarrow \mathbb{R}^+$ はエージェント i の変数 x^i_k とエージェント j の変数 x^j_l に値を割り当てたときに非負実数値を返すコスト関数である．さらに，エージェント間制約に関わる変数を接点変数と呼ぶ．

DCOP との違いは 2 点である．1 つはエージェント s が COP^s を管理すること，つまりエージェント s は複数変数複数コスト関数を持つ問題を集中的に解くことである．もう 1 つはエージェント内とエージェント間の 2 種類のコスト関数があることである．エージェント内コスト関数はあるエージェントが持つ 2 つの変数の間のコスト関数を表す．一方，エージェント間コスト関数は異なるエージェントが持つ 2 つの接点変数間のコスト関数を表す．全体のコストは 2 種類のコスト関数の全コストを合計した値であり，このコストの合計値を最小化するような解を探すことが目的となる．

3. 解法

DCOP-CLP を解くために，ラグランジュ分解法に基づいた解法を提案する．この解法は最適解に対する上界と下界を求めることができる．

3.1 2 次計画問題への変換

まず，全体の問題を 2 次計画問題に変換する．コスト関数を変換するためにエージェント i の変数 x_k^i とエージェント j の変数 x_l^j の間のエージェント間コスト関数を以下のように $|D_k^i| \times |D_l^j|$ のコスト関数に変換する．

$$\boldsymbol{F}_{k,l}^{i,j} = \begin{pmatrix} c_{1,1}^{i,j} & c_{1,2}^{i,j} & \cdots & c_{1,|D_j|}^{i,j} \\ c_{2,1}^{i,j} & c_{2,2}^{i,j} & \cdots & c_{2,|D_j|}^{i,j} \\ \vdots & \vdots & \ddots & \vdots \\ c_{|D_i|,1}^{i,j} & c_{|D_i|,2}^{i,j} & \cdots & c_{|D_i|,|D_j|}^{i,j} \end{pmatrix},$$

ここで，コスト行列の n 行 m 列目の要素 $c_{n,m}^{i,j}$ は変数 x_k と x_l にそれぞれ D_k^i の n 番目の値と D_l^j の m 番目の値割り当てたときのコストを表す．エージェント間コスト関数 $f_{k,l}^{i,j}$ のコストは以下の 2 次計画問題を解くことで求められる．

$$\begin{aligned} \min. \quad & \boldsymbol{\alpha}_{k,l}^{i,jT} \cdot \boldsymbol{F}_{k,l}^{i,j} \cdot \boldsymbol{\alpha}_{l,k}^{j,i} \\ \text{s. t.} \quad & \boldsymbol{x}_k^i - \boldsymbol{\alpha}_{k,l}^{i,j} = 0,\ \boldsymbol{x}_l^j - \boldsymbol{\alpha}_{l,k}^{j,i} = 0, \\ & \boldsymbol{x}_k^i, \boldsymbol{\alpha}_{k,l}^{i,j} \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D_k^i|}\}, \\ & \boldsymbol{x}_l^j, \boldsymbol{\alpha}_{l,k}^{j,i} \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|D_l^j|}\}, \end{aligned} \tag{1}$$

$\boldsymbol{e}_m$ は m 番目の要素が 1 でそれ以外の要素が 0 の単位列ベクトルである．ここで，変数 $\boldsymbol{x}_k^i$ に単位列ベクトル $\boldsymbol{e}_m$ を割り当てることは変数 $\boldsymbol{x}_k^i$ に D_k^i の m 番目の値を割り当てることを意味する．変数 $\boldsymbol{x}_k^i$ と $\boldsymbol{x}_l^j$ はこれらの単位列ベクトルを値に取る接点変数である．さらに，$\boldsymbol{\alpha}_{k,l}^{i,j}$ と $\boldsymbol{\alpha}_{l,k}^{j,i}$ は補助変数で，単位列ベクトルを値として取る．また，$\boldsymbol{\alpha}_{k,l}^{i,j}$ の右肩の T は転置を表す．注意として，$\boldsymbol{\alpha}_{k,l}^{i,j}$ と $\boldsymbol{\alpha}_{l,k}^{j,i}$ は (i,j,k,l) の各組み合わせ毎に導入される．これは 1 つの接点変数に対して，その接点変数が関わるエージェント間コスト関数の数と同じ数の制約を加えることを表している．この 2 次計画問題では，変数 x と補助変数 α が決定変数である．以下では，これらの決定変数に関する値域制約を省略する．さらに，各 COPϕ^s を 2 次計画問題に加えると以下のようになる．

$$\begin{aligned} \mathcal{P} \ : \min. \quad & \sum_{\boldsymbol{F}_{k,l}^{i,j} \in F} \boldsymbol{\alpha}_{k,l}^{i,jT} \cdot \boldsymbol{F}_{k,l}^{i,j} \cdot \boldsymbol{\alpha}_{l,k}^{j,i} + \sum_{s=1}^{n} \varphi^s(\boldsymbol{x}^{\boldsymbol{s}}) \\ \text{s. t.} \quad & \boldsymbol{x}_k^i - \boldsymbol{\alpha}_{k,l}^{i,j} = 0, \\ & \forall \boldsymbol{F}_{k,l}^{i,j} \in F, \end{aligned} \tag{2}$$

$$\begin{aligned} & \boldsymbol{x}_l^j - \boldsymbol{\alpha}_{l,k}^{j,i} = 0, \\ & \forall \boldsymbol{F}_{k,l}^{i,j} \in F, \end{aligned} \tag{3}$$

ここで，$\varphi^s(\boldsymbol{x}^{\boldsymbol{s}})$ は関数 $\varphi^s : D_1^s \times D_2^s \times \cdots \times D_{|D^s|}^s \to \mathbb{R}^+$ であり，$\boldsymbol{x}^{\boldsymbol{s}}$ はエージェント s が持つ変数の集合である．この関数は，$\boldsymbol{x}^{\boldsymbol{s}}$ の値を D^s 上の値に解釈し，その値の時のエージェント内コスト関数のコストの総和を返す．問題 $\mathcal{P}$ はすべてのコスト関数を含んでいるため，この問題を解くことで全体の問題に対する最適解を得ることができる．

3.2 分解

ここでは $\mathcal{P}$ を各エージェントに分配するために，$\mathcal{P}$ を部分問題に分解する．まず，(2) と (3) の制約を緩和して，ラグランジュ緩和問題を生成する．

$$\begin{aligned} \mathcal{L} \ : \ L(\boldsymbol{\mu}) \ = \ \min. \quad & \sum_{\boldsymbol{F}_{k,l}^{i,j} \in F} \boldsymbol{\alpha}_{k,l}^{i,jT} \cdot \boldsymbol{F}_{k,l}^{i,j} \cdot \boldsymbol{\alpha}_{l,k}^{j,i} + \sum_{s=1}^{n} \varphi^s(\boldsymbol{x}^{\boldsymbol{s}}) \\ & + \sum_{\boldsymbol{F}_{k,l}^{i,j} \in F} \boldsymbol{\mu}_{k,l}^{i,j}(\boldsymbol{x}_k^i - \boldsymbol{\alpha}_{k,l}^{i,j}) \\ & + \sum_{\boldsymbol{F}_{k,l}^{i,j} \in F} \boldsymbol{\mu}_{l,k}^{j,i}(\boldsymbol{x}_l^j - \boldsymbol{\alpha}_{l,k}^{j,i}), \end{aligned}$$

ここで，$\boldsymbol{\mu}_{k,l}^{i,j}$ と $\boldsymbol{\mu}_{l,k}^{j,i}$ はラグランジュ乗数と呼ばれ，それぞれ $|D_k^i|$ 次元と $|D_l^j|$ 次元の実数値を取る単位ベクトルである．$\boldsymbol{\mu}$ を全ての $\boldsymbol{\mu}_{k,l}^{i,j}$ と $\boldsymbol{\mu}_{l,k}^{j,i}$ の集合とすると，問題 $\mathcal{L}$ はいかなる $\boldsymbol{\mu}$ の値の集合に対して，原問題 $\mathcal{P}$ の下界を与える．

問題 $\mathcal{L}$ は，各エージェント間コスト関数 (i,j,k,l) の組み合わせに対して，

$$\begin{aligned} L_{i,j,k,l}^{aux}(\boldsymbol{\mu}) \ = \ \min. \quad & (\boldsymbol{\alpha}_{k,l}^{i,j})^T \cdot \boldsymbol{F}_{k,l}^{i,j} \cdot \boldsymbol{\alpha}_{l,k}^{j,i} \\ & - \boldsymbol{\mu}_{k,l}^{i,j} \boldsymbol{\alpha}_{k,l}^{i,j} - \boldsymbol{\mu}_{l,k}^{j,i} \boldsymbol{\alpha}_{l,k}^{j,i}, \end{aligned} \tag{4}$$

のように分解でき，さらに各エージェント s に対して，

$$L^s(\boldsymbol{\mu}) \ = \ \min. \sum_{\boldsymbol{F}_{k,l}^{s,j} \in F} \boldsymbol{\mu}_{k,l}^{s,j} \boldsymbol{x}_k^s + \sum_{\boldsymbol{F}_{k,l}^{i,s} \in F} \boldsymbol{\mu}_{k,l}^{i,s} \boldsymbol{x}_l^s + \varphi^s(\boldsymbol{x}^{\boldsymbol{s}}), \tag{5}$$

という部分問題に分解できる．ある $\boldsymbol{\mu}$ の値の集合が与えられたとき，部分問題 (5) は重み付き制約充足問題 (WCSP) に変換できる．例えば，目的関数の 1 つ目の項である $\boldsymbol{\mu}_{k,l}^{i,j} \boldsymbol{x}_k^i$ はソフト制約に変換できる．この制約では変数 $\boldsymbol{x}_k^i$ の値を変化させると，対応する $\boldsymbol{\mu}_{k,l}^{i,j}$ の値が重みとしてかかる．全ての x の項に対してこの変換は適用できる．さらに，COPϕ^s ($\varphi^s(\boldsymbol{x}^{\boldsymbol{s}})$) は WCSP の 1 種であるため，WCSP 形式に変換できる．したがって，部分問題 (5) は WCSP として解くことができる．対照的に，部分問題 (4) に関しては，補助変数の全ての値の組み合わせに関して目的関数値を計算する必要がある．

一方，ラグランジュ双対問題は形式的に以下のように定義される

$$\mathcal{D} \ : \ \max. \quad L(\boldsymbol{\mu}) \quad \text{s. t.} \quad \boldsymbol{\mu} \in \Re,$$

$L(\boldsymbol{\mu})$ は値 $\boldsymbol{\mu}$ における $\mathcal{L}$ の最適値である．これは明らかにラグランジュ乗数に関する無制約最大化問題であり，その目的関数値は $\mathcal{P}$ の最適値に対する下界である．$\mathcal{L}$ の分解の結果から $\mathcal{D}$ は以下のように分解できる．

$$\mathcal{D} \ : \ \max. \quad \sum_{\boldsymbol{F}_{k,l}^{i,j} \in F} L_{i,j,k,l}^{aux}(\boldsymbol{\mu}) + \sum_{s=1}^{n} L^s(\boldsymbol{\mu}) \quad \text{s. t.} \quad \boldsymbol{\mu} \in \Re.$$

提案方法では，最も高い目的関数値，つまり $\mathcal{P}$ の最適値に対する最も高い下界を与える $\boldsymbol{\mu}$ を探すためにラグランジュ双対問題を解く．

3.3 部分問題の分配

部分問題を以下のように各エージェントに分配する．エージェント s に関する部分問題 (5) はエージェント s が持つ決定変数 x^s のみで構成されているため，$L^s(\boldsymbol{\mu})$ は明らかにエージェント s が解く問題である．部分問題 (4) には，エージェント s と j の決定変数 $\alpha_{k,l}^{s,j}, \alpha_{l,k}^{j,s}$ があるので，$L_{s,j,k,l}^{aux}(\boldsymbol{\mu})$ は両エージェントが解く問題とする．このことはエージェント s と j が別々に $L_{s,j,k,l}^{aux}(\boldsymbol{\mu})$ を解くことを意味する．同様に，$L_{i,s,k,l}^{aux}(\boldsymbol{\mu})$ はエージェント i と s が解く．結果として，エージェント s は以下の部分問題を解く必要がある．

$$L_{s,j,k,l}^{aux}(\boldsymbol{\mu})/2, \quad \forall \boldsymbol{F}_{k,l}^{s,j} \in F, \tag{6}$$

$$L_{i,s,k,l}^{aux}(\boldsymbol{\mu})/2, \quad \forall \boldsymbol{F}_{k,l}^{i,s} \in F, \tag{7}$$

$$L^s(\boldsymbol{\mu}), \tag{8}$$

ここで，(6) と (7) は 2 つのエージェントに共有されているため，目的関数値を 2 で割る必要がある．この分配方法では共有された部分問題 (6) と (7) に対して，他のエージェントと同じ解が求められるように計算プロトコルを統一する必要がある．この分配方法は補助変数の通信を必要としない点，つまり，$\boldsymbol{\mu}$ を通信するだけで部分問題 (6) から (8) の解が得られる点が特徴といえる．

3.4 分散解法の概要

ここでは，ラグランジュ分解問題 $\mathcal{D}$ の最適値が最大となるような $\boldsymbol{\mu}$ を分散環境化において求める手順を示す．

Step 1: 全ての $\boldsymbol{\mu}$ の要素に 0 を代入する．
Step 2: 各エージェントは部分問題 (6) と (7) を解くために必要な $\boldsymbol{\mu}$ の値を交換する．
Step 3: 各エージェントは部分問題 (6) から (8) を解く
Step 4: 各エージェントは全域木上の隣接するエージェントと情報を交換し，最も高い下界 *BestLB* と最も低い上界 *BestUB* を計算する．
Step 5: もし終了条件を満たしているならば，*BestLB*，*BestUB* とその割り当てを返す．もし満たしていないならば，Step 6 で $\boldsymbol{\mu}$ を更新する．
Step 6: もし更新条件を満たしているならば，*BestLB* と *BestUB* を用いて更新幅を決定し，$\boldsymbol{\mu}$ を更新する．もし満たしていないならば，固定値を用いて更新し，Step 2 へ戻る．

この手順は Step 1 から始まり，終了条件を満たすまで Step 2 から Step 6 を繰り返す．この Step 2 から Step 6 までの操作を 1 サイクルとする．次に Step 4 と 5，6 について詳細述べる．

3.5 上界と下界の計算

$\boldsymbol{\mu}$ の更新に使用する上界と下界を計算するために，各エージェントの部分的な上界 (UB^s) と下界 (LB^s) を集める必要がある．LB^s はエージェント s の部分問題 (6) から (8) の目的関数値の合計である．また，UB^s はエージェント s が持つコスト関数に実行可能解を割り当てることにより得られる．この実行可能解は補助変数の値を (2) と (3) の制約式を満たすように x の値に写像することで得られる．つまり，UB^s を計算するためにはエージェントは，同じ接点変数を持つエージェントと接点変数の値を交換する必要がある．

分散環境下で情報を集めるためには，分散プロトコルが必要である．ここでは，[Hirayama 09] で提案されている分散プロトコルを用いる．このプロトコルで集める情報は UB^s と LB^s，G^s である．$G_{(t)}^s$ は $\boldsymbol{\mu}$ を更新するときに使用する劣勾配の値の 2 乗和の合計値のことである．集めた *LB* と *UB* に対して，それまでに得られた値よりも良い値であればそれぞれ *BestLB*, *BestUB* として値を保持する．*LB* と *UB* を集め終えたエージェントは *LB* と *UB* を用いて $\boldsymbol{\mu}$ を更新できる．

3.6 終了条件

もしエージェントがあるサイクルで *BestUB* と *BestLB* を得た場合，エージェントは *BestLB* を用いて *BestUB* が最適解かどうか判断できる．*BestUB* と *BestLB* が一致した場合，この *BestUB* を与える実行可能解が最適解であるとわかるため，操作を終了することができる．一方，Step2 から Step6 の操作は少なくとも 1 サイクル処理をしていれば，いつでも操作を終了することができる．この場合，それまでの *BestUB* と *BestLB* が得られる．

3.7 ラグランジュ乗数の更新

もし Step5 で終了条件を満たさない場合，より厳密な下界を得るためにエージェントは μ を更新する．これはラグランジュ双対問題に対する探索問題に相当し，劣勾配法を用いて探索する [Bertsekas 99]．劣勾配法に用いる更新幅は *BestLB* と *BestUB* が集められているかどうかで異なる．もしエージェント s が集め終えている場合，*BestLB*，*BestUB* を用いる．そうでない場合は，妥当な固定値 (例えばコスト関数の最大値の 10 分の 1 など) を用いる．劣勾配法は非常に簡単な方法であるが，この方法は必ずしも最適解に収束するとは限らない．

Step6 において注目すべき点は，もしエージェントが固定値のみを用いる場合，エージェントは $\boldsymbol{\mu}$ のみを交換するだけで Step2 から Step6 の操作うぃ繰り返すことができる点である．

4. 実験

DeQED の性能比較を行うために以下の 3 つの既存の DCOP ソルバーを使用した．

- MaxSum: メッセージ伝達方法である Generalized Distributive Law に基づいた発見的解法である．[Farinelli 08].
- DALO: t-optimality に基づいた発見的解法で，実行せずに解の質を保証することができる [Kiekintveld 10].
- DaCSA:ラグランジュ分解に基づく解法で，各部分問題を Toulbar2 を用いて解く．実行可能解は各サイクル毎に多数決を行い，緩和した制約を満たすように決定される．
- DeQED_A: 同じくラグランジュ分解に基づく解法で，各部分問題を Toulbar2 を用いて解く．実行可能解は各サイクル毎に補助変数の値を部分問題の最適解に合わせることで得られる．ラグランジュ乗数の更新に上界と下界の値を用いる．
- DeQED_F:DeQED_A との違いは更新幅にコスト関数の最大値の 10 分の 1 の値を用いる点である．この値は 10 サイクル毎に半分にする．

実験ではランダムグラフを 20 問生成し，使用した．このグラフはノードの数が 1,000 で各ノードが平均 3 つのエッジを持つ．また，各ノードのドメインサイズは 3 で，各エッジ上のコスト関数は $\{1, 2, ..., 10^5\}$ からランダムに選択されたコストをもつ．

ソルバーの性能を比較するために，以下では，DCOP と DCOP-CLP の 2 種類の実験設定を用意した．DCOP の実験では，DeQED に対して，MaxSum，DALO，DaCSA の比較をした．DCOP-CLP の実験では，エージェントの数を 100 にした場合，つまり各エージェントが局所問題を持つ場合に得られ

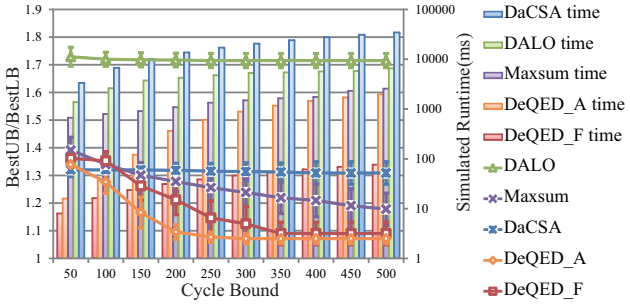

図 1: 1000 エージェントを持つランダムグラフに対する実行可能解の質の平均対制限サイクル数

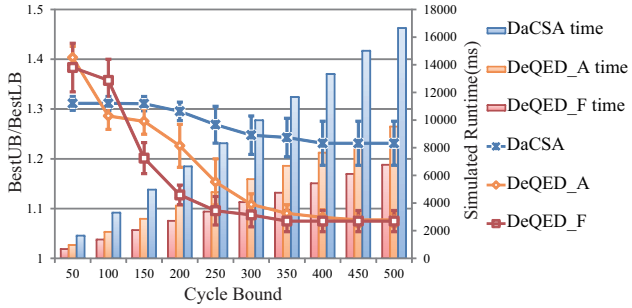

図 2: 100 エージェントを持つランダムグラフに対する実行可能解の質の平均対制限サイクル数

る実行可能解の質を比較する．対象とするソルバーは DaCSA と DeQED である．これらの実験では，サイクル数を 50 から 500 まで 50 刻みに設定し，各サイクルでの実行可能解の質を測った．サイクル数は DCOP の最も基本的な評価尺度であるが，1 サイクルにかかる計算コストはソルバーにより異なるため，ここでは評価尺度として simulated runtime[Sultanik 07] を用いた．実験では，あるサイクル内に発見した実行可能解の解の質 *BestUB*/*BestLB* を測る．ここで，*BestLB* は DaCSA もしくは DeQED が発見した最も高い下界を用いる．明らかに，実行可能解の質は 1 に近いほど良い．

DaCSA と DeQED は Java で書き，JDK1.6.0_20 でコンパイルした．DaCSA と DeQED の内部ソルバーとして Toulbar2 (version 0.9.4.0) を著者のページからダウンロードし，使用した．DALO も著者のページ [*1] からダウンロードし，DALO-t=1 のアルゴリズムで実験した．MaxSum については， FRODO (version 2.10.4) に実装されているものを初期設定のまま使用した．これらのソルバーの実験は Core-i7@3.4GHz，4 コア，8 スレッド，8GB メモリを積んだ Ubuntu11.10(64bit) において実験をした．

図 1 は DeQED が他のソルバーよりも良い解の質を与えていることを示している．また，DeQED_F は DeQED_A と比較しそれほど差がないという結果が得られた．しかし DeQED_A のほうが比較的少ないサイクル数で収束していることがわかる．simulated runtime では DeQED_F が DeQED_A と比較して高速であることがわかる．その理由としては，メッセージ通信量が影響していると考えられる．上界，下界を集める場合，メッセージの情報量は増加していくため，その情報の処理にかかる時間も増えてしまう．それに対し，DeQED_F の通信はラグランジュ乗数の値だけであるため，どのサイクルにおいても情報量はつねに一定である．

DeQED が同じラグランジュ分解を用いた DaCSA よりも実行時間が短い理由は，DaCSA は局所問題を解く必要があるのに対して，DeQED は部分問題 (8) において局所問題を解く必要がないためである．つまり，部分問題 (8) は係数の符号を見るだけで値が決定できる簡単な問題となる．しかし，simulated runtime は実装環境に依存するため，必ずこの順列になるとは言えない．現に DaCSA は局所問題を解くために Toulbar2 を外部コマンドとして呼び出している上に Toulbar2 に解かせる局所問題をファイルとして記述している．より正確な比較を行うためには実装面における工夫が必要になると考えられる．なお，DCOP-CLP においては，DeQED も同様の方法で局所問題を解くため，公平な比較ができると考えられる．図 2 は Agent 数を変化させても DeQED が比較的良い解の質を与えていることを示している．simulated runtime に関してはどの実験においても DeQED の方が短い時間で実行できている．その理由としては，DaCSA は部分問題を作る際にエージェント間コスト関数の数と同数の余分な変数を生成する必要があることが考えられる．一方，DeQED もエージェント間コスト関数と同数の余分な変数を生成するが，これらの変数は全て補助変数であり，比較的簡単な部分問題 (6) と (7) を解くことで解が得られる．

∗1 http://teamcore.usc.edu/dcop/

5. おわりに

DCOP の解法である DeQED を提案した．DeQED は 2 次計画変換を用いるラグランジュ分解に基づいた解法である．特筆すべき特徴は，DeQED_F のようにエージェントが双対変数を交換するだけで上界と下界を得ることができるという点である．実験では，DeQED_F は適切な更新幅を選択していればDeQED_A と同等程度の解が得られることが分かった．

参考文献

[Bertsekas 99] Bertsekas, D. P. and Bertsekas, D. P.: *Nonlinear Programming*, Athena Scientific, 2nd edition (1999)

[Farinelli 08] Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm, AAMAS '08, pp. 639–646 (2008)

[Hirayama 09] Hirayama, K., Matsui, T., and Yokoo, M.: Adaptive price update in distributed Lagrangian relaxation protocol, AAMAS '09, pp. 1033–1040 (2009)

[Kiekintveld 10] Kiekintveld, C., Yin, Z., Kumar, A., and Tambe, M.: Asynchronous algorithms for approximate distributed constraint optimization with quality bounds, AAMAS '10, pp. 133–140 (2010)

[Sultanik 07] Sultanik, E. A., Lass, R. N., and Regli, W. C.: DCOPolis: a framework for simulating and deploying distributed constraint reasoning algorithms, AAMAS '08, pp. 1667–1668 (2007)

[Vinyals 10] Vinyals, M., Pujol, M., Rodriguez-Aguilar, J. A., and Cerquides, J.: Divide-and-coordinate: DCOPs by agreement, AAMAS '10, pp. 149–156 (2010)