RF-006

分散制約最適化問題の解法 DUCT における冗長なサンプリング結果の削減
Reducing Redundant Samples in Distributed UCT Algorithm for Distributed Constraint Optimization Problems

冨板雅大 † 松井俊浩 †
Masahiro Tomiita Toshihiro Matsui

1. はじめに

電力需要に対する最適な配電網の構築 [1, 2] や広域の観測情報の収集するための技術 [3, 4, 5] として，マルチエージェントシステムが研究されている．マルチエージェントシステムでは，複数の自律的なエージェントが，それぞれに分散している情報を協調して処理する．このような枠組みは，中央集中型のシステムと比較して，耐障害性やスケーラビリティの点において優れている．このマルチエージェントシステムの協調問題解決の基礎的なモデルは分散制約最適化問題として定式化される [6, 7]．分散制約最適化問題では，エージェントに分散して配置された，変数と関数を用いて，問題を局所的に記述し，エージェント間のメッセージ通信により必要な情報のみを交換し，最適化する．分散制約最適化問題の解法は，厳密解法と非厳密解法に分類される．厳密解法には動的計画法に基づく DPOP[6] や木探索に基づく ADOPT[7] がある．これらの解法は必ず最適解を得ることができる．しかし，すべての解空間を探査する必要があり，問題が大規模で複雑になると，網羅的な探査は現実的ではないため，適用が困難である．そのような問題に対して，これらの解法を適用可能な規模に問題を近似する手法 [8, 9] もあるが，解品質の低下は避けられない．非厳密解法には局所探索に基づく DSA[10] などがある．このような解法は必ず最適解が得られるとは限らないが，計算量，メッセージサイズが少なく，大規模で複雑な問題にも適用できる．しかし，各変数について，その変数と制約により直接関係する変数のみを考慮する，局所的な問題をそれぞれ最適化するため，大域的な情報は活用されない．

本研究では，分散制約最適化問題の非厳密解法である DUCT[11] に注目する．DUCT の特徴は分散協調型の木探索にサンプリングを導入した点である．この手法の利点として，全探索が困難な問題に対して，比較的精度が良い解を得ることが期待されることが挙げられる．しかし，DUCT には，サンプリングを反復する際に，問題のグラフから構成した擬似木 [6] の深さが深いエージェントほど，保存すべき情報の数が際限なく増加するという問題点がある．

そこで，この問題点を改善するために，冗長なサンプリング結果を削減する手法を提案する．この手法では，解法が用いるグラフ構造である擬似木上の各エージェントについて，祖先エージェントから受信した変数値の情報および自身の変数値が同じサンプリング結果のうち，そのエージェントのコストが，最小のもの以外を削減する．これにより，各エージェントが記憶するサンプリング結果の数を削減できる．

本論文は以下のように構成される．2 章では，本研究の背景として分散制約最適化問題とその解法について述べる．3 章から 5 章では，DUCT とその問題点の改善手法について述べる．3 章では，既存手法である DUCT について説明する．4 章では，本論文で提案する DUCT における冗長なサンプリング結果の削減，提案手法の実現方法および期待される効果と影響について述べる．5 章では，提案手法の有効性を実験により評価する．6 章に実験結果についての考察を示し，最後に，7 章においてこれらをまとめる．

†名古屋工業大学, Nagoya Institute of Technology

2. 分散制約最適化問題とその解法

2.1. 分散制約最適化問題

分散制約最適化問題 [6, 7] は，制約最適化問題の変数が複数のエージェントに分散された問題である．制約最適化問題とは，変数間の制約に対応するコスト関数値の総和を最小化するような変数値の割当てを求める問題である．

分散制約最適化問題は，以下に示す $\langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$ の五つ組により定義される．

- 変数の集合 $\mathcal{X} = \{x_1, \ldots, x_n\}$
- 値域の集合 $\mathcal{D} = \{D_1, \ldots, D_n\}$
- コスト関数の集合 $\mathcal{F} = \{f_1, \ldots, f_m\}$
- エージェントの集合 $\mathcal{A} = \{a_1, \ldots, a_p\}$
- 関数 α: $\mathcal{X} \to \mathcal{A}$

各変数 x_i は有限で離散的な値域 D_i を持つ．各コスト関数 f_j は変数間の制約の評価値を表すものであり，変数のある部分集合 $\mathcal{X}_j \subseteq \mathcal{X}$ の値の組に対するコストを返す実数値関数である．本研究では，全てのコスト関数は，二つの変数の値の組に関する非負の値を返す，二項関数であると仮定する．各エージェント a_i は自身の状態を表す変数を管理する．各変数を，ある一つのエージェントに対応させる関数 α によって，管理する変数を決定する．本研究では，各エージェントは単一の変数を持つと仮定し，エージェント a_i が持つ変数を x_i とする．また表記の簡単のために，必要に応じてエージェントと変数を同一視する．分散制約最適化問題の目的は，式 (2.1) により表されるすべてコスト関数の総和を最小にするような，すべての変数への変数値の割当てを求めることである．

$$\sum_{j=1}^{m} f_j(\mathcal{X}_j) \tag{2.1}$$

分散制約最適化問題のグラフ表現の一つに制約網がある．制約網は，変数 $x_i \in \mathcal{X}$ をノード，コスト関数 $f_j \in \mathcal{F}$ を辺として表現したグラフである．図 1 に，5 つの変数を表現したノード，6 つのコスト関数を表現した辺からなる制約網を示す．各ノードは自身の変数が関係す

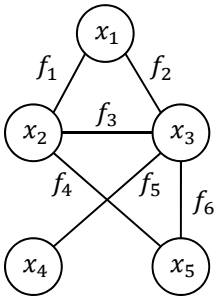

図 1: 制約網の例

るコスト関数の内容を知る．また，制約網において隣接しているノードとのみ，メッセージ通信をすることができる．各エージェントは，適切な分散協調アルゴリズムを用いて，自身の変数に対して最適な変数値の割当てを求める．

2.2. 分散制約最適化問題の解法

分散制約最適化問題の解法は，最適解が得られる厳密解法と，準最適解が得られる非厳密解法の 2 種類に分類される．

2.2.1. 厳密解法

代表的な厳密解法には，DPOP[6]，ADOPT[7] などがある．DPOP と ADOPT は，前処理として，制約網に対する深さ優先探索木などの生成木および，それに基づく擬似木 [6] を生成する．擬似木の辺に沿って，メッセージ交換する分散アルゴリズムにより最適解を求める．これらの解法は必ず最適解を得ることができる．しかし，最悪の場合にはすべての解空間を探査する必要がある．したがって，問題が大規模で複雑になると，網羅的な探査は現実的ではないため，適用が困難である．

2.2.2. 非厳密解法

代表的な非厳密解法には，DSA[10]，DUCT[11]，分散 Gibbs[12] などがある．DSA は局所探索を確率的に実行する．DUCT と分散 Gibbs は，制約網に基づく擬似木を用いて，確率的に解をサンプリングする．

DSA は，各エージェントが制約網において隣接するエージェントの状態を取得し，確率的に解を改善する．この手法では，エージェントの状態に関する情報のみをメッセージとして送受信するため，通信コストを低く抑えることができる．そのため比較的大規模な問題に適している．しかし，各エージェントはコスト関数で関係する近傍エージェントの状態のみに基づいて自身の状態を決定する．そのため，最適化において大域的な情報は活用されない．

DUCT は，各エージェントを擬似木によって同期し，変数値をサンプリングする．DUCT の特徴は，サンプリングが，ゲーム木探索などで用いられる UCT に基づくことである．その一方で，確率的な境界値を保持するために，各エージェントのメモリ使用量は比較的多い．

分散 Gibbs は，DUCT と同様に，各エージェントを擬似木によって同期し，変数値をサンプリングする．分散 Gibbs の特徴は，各エージェントのメモリ使用量が比較的少ないことである．その一方で，サンプリングは，自変数と現在の近傍の変数値のみに基づく．

表 1: 記号とその用途

記号	用途
t	サンプリングの回数
k	変数ノード x_k を管理するエージェント
$C(k)$	変数ノード x_k の子ノードの集合
コンテキスト a	根から葉へ伝播される部分解
コスト y_k^t	葉から根へ伝播されるコスト
バウンド B_k^t	変数ノード x_k の各変数値の評価値の最小値

本研究では，大域的に集計された確率的な境界値に基づくサンプリング手法である DUCT に注目し，そのメモリ使用量の削減について検討する．

3. 既存手法： DUCT

本章では，木探索にサンプリングを導入した分散制約最適化問題の非厳密解法である DUCT[11] について説明する．

3.1. 記号

DUCT において用いられる記号及びそれらの用途の一覧を表 1 に示す．コンテキスト a は，3.3 節で説明する擬似木上の木辺に沿って，コンテキストメッセージとして，根から葉へトップダウンに伝播される．また，コスト y_k^t は，擬似木上の木辺に沿って，コストメッセージとして，葉から根へボトムアップに伝播される．これらの詳細は，次節以降で説明する．

3.2. 全体の動作

DUCT 全体の動作は，以下に示す 3 つの手順からなる．

手順 1: 制約網に対応する擬似木の生成

手順 2: 根ノードから葉ノードへのトップダウンな，コンテキストの伝搬

手順 3: 葉ノードから根ノードへのボトムアップな，コストとバウンドの最小値の伝搬

まず，前処理として，手順 1 の制約網に対応する擬似木を生成する．その後，手順 2 の根ノードから葉ノードへのトップダウンな，コンテキストの伝搬と，手順 3 の葉ノードから根ノードへのボトムアップな，コストとバウンドの最小値の伝搬を繰り返す．上記の，手順 2 と 3 の一組を 1 回のサンプリングとする．このサンプリングを定められた回数だけ繰り返すことにより DUCT が実行される．

3.3. 制約網に対応する擬似木の生成

擬似木は，制約網上で深さ優先探索をすることにより生成される．まず，制約網の変数ノードを一つ選択して，これを根として深さ優先探索をする．この探索により通った辺を木辺，通らなかった辺を後退辺と呼ぶ．二つの変数ノードが後退辺でつながれている時，根に近い側を擬似親，葉に近い側を擬似子と呼ぶ．図 1 の制約網上の変数ノード x_1 を根として，制約網から擬似

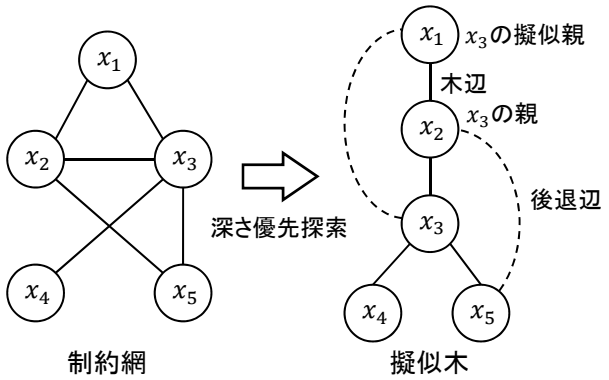

図 2: 制約網に対応する擬似木の生成

木を生成する例を図 2 に示す．

図 2 の擬似木において実線で表される辺が木辺，破線で表される辺が後退辺である．x_3 に着目すると，親は x_2，子は x_4 と x_5，擬似親は x_1 である．また，x_3 は x_1 の擬似子である．深さ優先探索により擬似木を生成した場合には，部分木間には辺がないため，問題が分割統治され，各部分木のコストなどの計算を並行できる．このように，前処理として擬似木を生成することは，分散制約最適化問題の解法において広く行われる．擬似木を用いる分散制約最適化問題の解法には，DUCT の他に DPOP[6] や ADOPT[7] などがある．

3.4. コンテキストの送信

擬似木を生成した後，根ノードの変数，つまりその変数を持つエージェントは，自身の変数値を決定する．この時，根ノードの変数は自身より下の部分木についての情報をまだ何も持たないため，変数値は根ノードの変数の値域から一様分布の乱数により，ランダムに選択される．根ノードの変数は，自身が選択した変数値を，擬似木上のすべての子ノードに送信する．このメッセージをコンテキストと呼び，a により表される．

続いて，コンテキストを受信した変数ノードも，自身の変数値を自身の変数の値域から一様分布の乱数により，ランダムに選択する．受信したコンテキストに自身の変数値の情報を加えた新たなコンテキストを，すべての子ノードに送信する．コンテキストが擬似木上のすべての葉ノードに到達するまで，トップダウンにこの操作を繰り返す．

擬似木上の葉ノードがコンテキストを受信した場合は，自身の変数について，親および擬似親の変数値に対するコスト関数の和が最小となる変数値を選択する．この時，根から葉までの経路上のすべての変数ノードに対して，一通りの割当てが決定したことになる．

3.5. コストとバウンドの送信

すべての変数ノードに対する割当てが決定した後で，この割当てのコストを計算する．コンテキストに対して変数値 d を決定した葉ノードは，自身の変数値と親および擬似親の変数値との間のコストを式 (3.1) により計算し，コストメッセージ y_k^t として親ノードへ送信する．

$$y_k^t = l(a,d) \tag{3.1}$$

ここで，$l(a,d)$ は，コンテキスト a を受信した変数ノードが変数値 d を選択した時の，自身の変数値と親および擬似親の変数値との間のコストの総和を意味する．すべての子ノードからコストを受信した変数ノードは，自身が根となる部分木のコストに自身の変数値と親および擬似親の変数値との間のコストの総和を加えたコスト y_k^t を式 (3.2) により計算し，コストメッセージとして親ノードへ送信する．

$$y_k^t = l(a,d) + \sum_{k' \in C(k)} y_{k'}^t \tag{3.2}$$

$\sum_{k' \in C(k)} y_{k'}^t$ は，変数ノード x_k のすべての子ノード $x_{k'}$ から受信したコスト $y_{k'}^t$ の総和を表す．コストが擬似木上の根ノードに到達するまで，ボトムアップにこの操作を繰り返す．このようにして，擬似木全体のコストを計算する．

すべての変数ノードはコストを計算し送信すると同時に，次回のサンプリングの確率を決定するためのバウンドと呼ばれる値を計算する．バウンドは各変数値それぞれの評価値であり，それらの最小値はコストメッセージにより親ノードへ送信される．バウンドは葉ノード以外のノードは式 (3.3) の上の場合により表される．また，葉ノードは下の場合により表される．

$$B_{a,d}^t = \begin{cases} l(a,d) + \max\left\{(\hat{\mu}_{a,d}^t - L_{a,d}^t), \sum_{k' \in C(k)} B_{k'}^t\right\} \\ l(a,d) \end{cases} \tag{3.3}$$

ここで，$B_{k'}^t$ は変数ノード x_k の子ノード $x_{k'}$ から送られてきたバウンドであり，式 (3.4) により表される．

$$B_{k'}^t = \min_{d' \in D_{k'}} B_{a',d'}^t \tag{3.4}$$

a' は，コンテキスト a を受信し変数値 d を選択した変数 x_k の子ノード $x_{k'}$ が受信したコンテキストを意味し，式 (3.5) のように表される．

$$a' = a \cup \{x_k = d\} \tag{3.5}$$

$\hat{\mu}_{a,d}^t$ は，コンテキスト a を受信した変数ノード x_k が変数値 d を選択した時に発見された最小のコストを意味し，式 (3.6) のように定義される．

$$\hat{\mu}_{a,d}^t = \min\{y_k^l \mid l \leq t : a_k^l = a, x_k^l = d\} \tag{3.6}$$

$L_{a,d}^t$ は，UCB スタイルのバウンド [13] を意味し，式 (3.7) のように定義される．

$$L_{a,d}^t = \sqrt{\frac{2\lambda_a \ln \tau_a^t}{\tau_{a,d}^t}} \tag{3.7}$$

λ_a は，最も深い葉ノードへのパスの長さを意味する．また，τ_a^t はコンテキスト a の受信した回数を意味し，$\tau_{a,d}^t$ はコンテキスト a を受信したとき，変数値 d を選択した回数を意味する．τ_a^t と $\tau_{a,d}^t$ はそれぞれ式 (3.8)，式 (3.9) のように定義される．

$$\tau_a^t = \sum_{l=1}^{t} \mathrm{II}\{a_k^l = a\} \tag{3.8}$$

$(x_1, x_2) = (0, 1)$ についての最小コスト

コンテキスト a	x_1	0	0	0	0
自身の変数値 d	x_2	0	1	1	1
コスト	y_2^t	7	5	2	3

図 3: x_2 を管理するエージェントのメモリ

$$\tau_{a,d}^t = \sum_{l=1}^{t} \mathrm{II}\{a_k^l = a \wedge x_k^l = d\} \tag{3.9}$$

ここで，$\mathrm{II}\{\cdot\}$ は，インジケータ関数であり，関数中の式が成立する時は 1，成立しない時は 0 となる関数である．

3.6. 二回目以降のサンプリング

DUCT では，変数値の決定とコストの計算を繰り返すことにより，解空間を探索する．2 回目以降の葉ノード以外の変数ノードの変数値の割当ては，一様分布の乱数によりランダムに値を選択されるのではなく，これまでの探索結果に基づく確率にしたがって選択される．変数ノード x_k^t に割当てる変数値は，式 (3.10) に従う．

$$x_k^t = \arg\min_{d \in S_a^t} B_{a,d}^t \tag{3.10}$$

ここで，S_k^t はサンプル可能な変数値の集合を意味し，式 (3.11) により定義される．

$$S_a^t = \{d \in D_k \mid B_{a,d}^t \neq l(a,d) + \hat{\mu}_{a,d}^t\} \tag{3.11}$$

3.7.DUCT の問題点

前節で説明した DUCT の動作において，各エージェントがサンプリングのために保存する情報は，受信したコンテキスト a と自身が選択した変数値 d，およびコスト y_k^t である．図 2 の変数 x_2 を管理するエージェントのメモリの内容の例を，図 3 に示す．図 3 のテーブルの縦方向は，コンテキスト a の規模が自身の先祖ノードの数により決定するため，先祖ノードの数に比例する．また，テーブルの横方向は，一回のサンプリングによって 1 列増加するため，サンプリングの回数に比例する．しかし，解空間が指数関数的であるため，格納しなければならないコンテキストの情報の最大の数は，先祖ノードの数に対して指数関数的である．したがって，多数のサンプリングをする時に，保存すべきコンテキストの情報の数が際限なく増加することが問題となる．この影響は，先祖ノードの数に従うため，擬似木上の下部に位置するノードほど大きくなる．

4. 提案手法： サンプリング結果の削減

本章では，前章において説明した DUCT の問題点を改善するために，冗長なサンプリング結果を削減する手法を提案する．

4.1. 冗長なサンプリング結果

DUCT におけるサンプリングの際に，各変数ノードのエージェントは，過去のサンプリング結果として以下の値を用いる．

$(x_1, x_2) = (0, 1)$ についての最小コスト

コンテキスト a	x_1	0	0	0	0
自身の変数値 d	x_2	0	1	1	1
コスト	y_2^t	7	5	2	3

削減できる

図 4: x_2 を管理するエージェントのメモリ中の削減できるサンプリング結果

- $B_{a,d}^t$: 式 (3.3) の変数値 d のバウンド
- $\hat{\mu}_{a,d}^t$: 式 (3.6) の部分木におけるコストの最小値
- $L_{a,d}^t$: 式 (3.7) の UCB スタイルのバウンド [13]
- τ_a^t: 式 (3.8) のコンテキスト a を受信した回数
- $\tau_{a,d}^t$: 式 (3.9) のコンテキスト a を受信し，変数値 d を選択した回数

ただし，$\hat{\mu}_{a,d}^t$ と $L_{a,d}^t$ は，$B_{a,d}^t$ の計算に影響する値である．また，τ_a^t と $\tau_{a,d}^t$ は，$L_{a,d}^t$ の計算に影響する値である．これらの値のうち $\hat{\mu}_{a,d}^t$ に注目する．前章の図 3 に示した x_2 を管理するエージェントのメモリを例に説明する．ここで，変数 x_1，x_2 は図 2 の擬似木における変数である．

ただし，図 3 に示したように，変数 x_2 を管理するエージェントが受け取るコンテキスト a はすべて $a = \{0\}$ すなわち同一のものとし，x_2 の値域は $D_2 = \{0, 1\}$ であると仮定する．変数 x_2 を管理するエージェントが，変数値 0 のバウンド $B_{a,0}^t$ を計算するために必要なサンプリング結果は，自身の値が 0 である 1 列のみである．また，変数値 1 のバウンド $B_{a,1}^t$ の計算において必要なサンプリング結果は，自身の値が 1 である 3 列である．しかし，変数値 1 のバウンド $B_{a,1}^t$ の計算において用いられる $\hat{\mu}_{a,1}^t$ はコンテキスト a を受信し，変数値 1 を選択した時に発見された最小のコストを意味するため，この場合に用いる値は，コストが 2 となっている列のみの値である．したがって，これらのサンプルのサンプリングの回数を無視すれば，図 4 のようにコストが 3 と 5 の列のサンプリング結果を削減できる．以上のように，各エージェントにおいて，受信したコンテキストと自身の選択した値が同じサンプリング結果のうち，コストが最小のもの以外は冗長であるといえる．

4.2. 提案手法の実現方法

提案手法は，アルゴリズム 1 に示す擬似コードのように実現される．この擬似コードは，基本的には 3.4 から 3.6 節に示したとおりの計算を表す．ただし，擬似コードの 35 行目において，各変数ノードは，過去のサンプリング結果を調査し，現在のサンプリング結果のコンテキストと自身の変数値が同じサンプリング結果がある場合，現在のサンプリング結果により得られたコストの値以上の値をコストとして持つサンプリング結果を把握する．その後，各変数ノードは，39 行目において，そのサンプリング結果を削減する．

Algorithm 1 提案手法における各変数ノード x_k の動作

```
1: t ⇐ サンプリング回数
2: 制約網に対応する擬似木の生成
3: for i = 1 to t do
4:     if x_k が根ノードである then
5:         if t==1 then
6:             変数値を D_k から一様分布の乱数によりランダムに選択
7:         else
8:             変数値を式 (3.10) に基づき選択
9:         end if
10:        選択した変数値をコンテキストとしてすべての子ノードに送信
11:        if すべての子ノードからコストとバウンドを受信 then
12:            擬似木全体のコストと各変数値のバウンドを計算
13:        end if
14:    else if x_k が葉ノードである then
15:        if 親ノードからコンテキストを受信 then
16:            近傍ノードの変数値に対するコスト関数の和が最小となる値を
               変数値に選択
17:            コストと各変数値のバウンドを計算
18:            コストと各変数値のバウンドの最小値を親ノードへ送信
19:        end if
20:    else
21:        if t==1 then
22:            変数値を D_k から一様分布の乱数によりランダムに選択
23:        else
24:            変数値を式 (3.10) に基づき選択
25:        end if
26:        if 親ノードからコンテキストを受信 then
27:            受信したコンテキストに自身の値を加えたものをすべての子
               ノードに送信
28:        end if
29:        if すべての子ノードからコストとバウンドを受信 then
30:            コストと各変数値のバウンドを計算
31:            コストと各変数値のバウンドの最小値を親ノードへ送信
32:        end if
33:    end if
34:    cost1 ⇐ サンプリングで得られたコスト
35:    コンテキストと自身の選択した値が同じ過去のサンプリング結果の有
       無の調査
36:    if コンテキストと自身の選択した値が同じ過去のサンプリング結果
       がある then
37:        cost2 ⇐ そのサンプリングで得られたコスト
38:        if cost2 ≥ cost1 then
39:            そのサンプリング結果の削減
40:        end if
41:    end if
42: end for
```

4.3. 予想される効果と影響

提案手法により削減されるサンプリング結果は，現在のサンプリング結果のコンテキストと自身の変数値が同じサンプリング結果である．したがって，サンプリングの回数に対して，解の収束が速い問題ほど，多くのサンプリング結果が削減できることが予想される．また，根エージェントに近いエージェントほど，多くのサンプリング結果が削減される．これは，先祖エージェントから受信するコンテキスト及び自身が選択する値の組み合わせの数が少ないためである．しかし，サンプリング結果の削減に伴い，$L^t_{a,d}$ の計算に影響する τ^t_a と $\tau^t_{a,d}$ の値が変化することにより，各変数における各変数値のバウンドが変化する可能性がある．このためにサンプリングが不十分となり，コスト値が大きくなることがあると予想される．特に，コストが小さい問題に関しては，バウンドの計算に影響する式 (3.3) 中の $\hat{\mu}^t_{a,d} - L^t_{a,d}$ の値が，コスト値が大きい問題と比較して，相対的に大きく変化することにより，この傾向がより顕著になることが予想される．

4.4. 提案手法により追加されるオーバヘッド

提案手法により，追加される処理によるオーバヘッドについて説明する．提案手法では，一回のサンプリングをする度に，各変数ノードにおいて，過去のすべてのサンプリング結果を調査し，現在のサンプリング結果と比較する．過去の一つのサンプリング結果と現在のサンプリング結果を比較するために必要な計算量は，コンテキスト a の規模に依存するため，先祖ノードの数に比例する．したがって，過去の一つのサンプリング結果との比較により，追加されるオーバヘッドは先祖エージェントの数を T とすると，$O(T)$ となる．これを現在のサンプリングの回数から 1 を引いた回数分だけ繰り返す．以上のような処理を全体のサンプリングの回数分繰り返すため，全体のサンプリングの回数を t とすると，各エージェントに追加される全体のオーバヘッドは式 (4.1) により表される．

$$O(T \cdot \sum_{k=1}^{t}(k-1)) = O(T \cdot \frac{t(t-1)}{2}) = O(T \cdot t^2) \quad (4.1)$$

これは，多項式時間であるため，各エージェントにおいて，提案手法によるオーバヘッドは，t が極端に大きな値でなければ，実際的な程度であるといえる．

5. 評価

本章では，既存手法の DUCT と提案手法を実験により比較する．

5.1. 問題の生成方法と評価方法

実験では，基本的な評価の結果を得ることを目的として，各変数ノードに対して制約辺をランダムに接続することにより作成したグラフ構造となる 2 種類の問題のクラスについて実験した．一方の問題のクラスでは，変数の数を $|\mathcal{X}| = 10, 20, \ldots, 50$ と変化させ，各変数の値域を $|D_i| = 20$，グラフ密度を $p = 0.3$ とした．他方の問題のクラスでは，各変数の値域を $|D_i| = 10, 20, \ldots, 50$ と変化させ，変数の数を $|\mathcal{X}| = 20$，グラフ密度を $p = 0.3$ とした．ここで，グラフ密度は，グラフ構造において実際の辺数を存在しうる最大辺数で割った値により表されるパラメータであり，1.0 の場合は完全グラフである．コスト関数 f_j の値はいずれも 0〜10 のランダムな値に設定した．これらの乱数値は一様分布に基づいて選択した．

また，上記と同様のグラフ構造であるが，系全体のコスト値の合計が小さい問題についても実験した．これらの問題では，変数の数を $|\mathcal{X}| = 10, 20, \ldots, 40$ と変化させ，各変数の値域を $|D_i| = 20$，グラフ密度を $p = 0.3$ とした．ただし，コスト関数 f_j の値は各エージェント数において，それぞれ表 2 に示す値の範囲におけるランダムな値に設定した．これらの乱数値も上記の場合と同様に一様分布に基づいて選択した．

表 2: 各エージェントにおけるコスト関数値の範囲

エージェント数 $\|\mathcal{X}\|$	コスト関数値の範囲
10	0〜10
20	0〜2
30	0,1
40	0,1

それぞれの問題について，既存手法と提案手法をサンプリング回数 5,000 回を 30 回ずつ実行し，各エージェントが記憶したサンプル数の総和とコスト値の平均を比較した．ただし，コスト値が小さい問題においては，

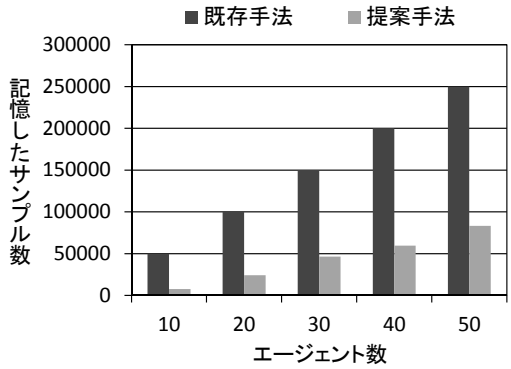

図 5: エージェント数を変化させた場合の記憶したサンプル数の比較

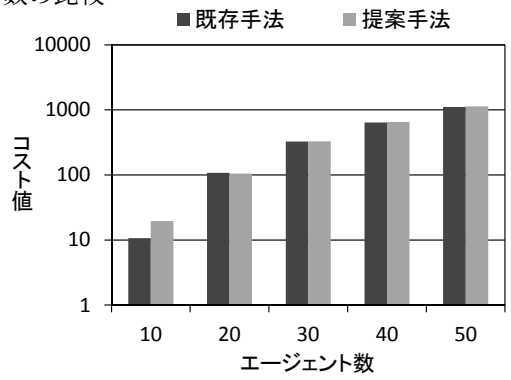

図 6: エージェント数を変化させた場合のコスト値の比較

既存手法と提案手法に加え，根エージェント以外のエージェントのみ提案手法を実行したものを比較した．この比較の詳細については，5.4 節において述べる．

記憶したサンプル数の総和は次のように評価した．既存手法ではサンプリング結果を削減しない．そのため，記憶したサンプル数の総和は常にエージェント数にサンプリング回数を掛けた値となる．その一方で，提案手法では，記憶するサンプル数の削減をしているが，各変数の初期値の組み合わせによって記憶するサンプル数の総和が異なる．そこで，最悪の場合すなわち，各エージェントが記憶したサンプル数の総和が最も大きい値を用いて既存手法と比較した．

5.2. エージェント数を変化させた場合の評価

エージェント数を変化させた場合の結果を図 5 と図 6 に示す．図 5 では，横軸をエージェント数，縦軸を記憶したサンプル数の総和とし，図 6 では，横軸をエージェント数，縦軸をコスト値とした．ただし，図 6 の縦軸の値は，対数目盛により表されている．

図 5 において，提案手法では既存手法と比較して，エージェント数 $|\mathcal{X}| = 10$ の場合では約 85%，$|\mathcal{X}| = 20$〜50 の場合では約 65〜75%のサンプル数が削減された．また，図 6 において，提案手法では既存手法と比較して，エージェント数 $|\mathcal{X}| = 10$ の場合を除き，削減されたサンプル数に対して，コスト値の大きな変化はないことが示された．

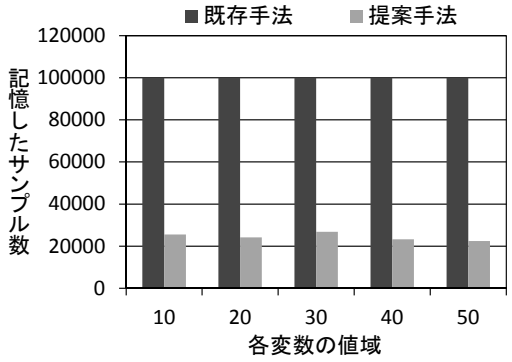

図 7: 各変数の値域を変化させた場合の記憶したサンプル数の比較

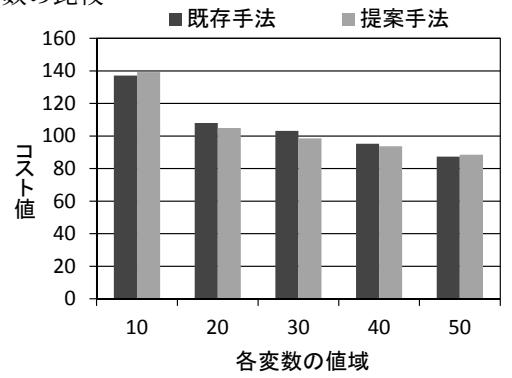

図 8: 各変数の値域を変化させた場合のコスト値の比較

5.3. 各変数の値域を変化させた場合の評価

各変数の値域を変化させた場合の結果を図 7 と図 8 に示す．図 7 では，横軸を各変数の値域，縦軸を記憶したサンプル数の総和とし，図 8 では，横軸を各変数の値域，縦軸をコスト値とした．

図 7 において，提案手法では既存手法と比較して，各変数の値域を変化させたどの場合においても約 75%のサンプルが削減された．また，提案手法では，いずれの値域の規模においても，記憶したサンプル数の総和に大きな変化はなかった．これは，値域の規模が増加しても，既存手法と提案手法のサンプル数は同一であり，削減される程度が一定であるためだと考えられる．図 8 において，提案手法では既存手法と比較して，削減されたサンプル数に対して，コスト値の大きな変化はないことが示された．しかし，$|D_i| = 20$〜40 において，既存手法よりも提案手法の方がコスト値が小さい．これは，値域の規模が増加するに伴い，各変数の初期値の組み合わせによる解品質の精度の違いが拡大したためであるだと考えられる．

5.4. コスト値が小さい問題の場合の評価

コスト値が小さい問題の場合の結果を図 9 と図 10 に示す．図 9 では，横軸をエージェント数，縦軸を記憶したサンプル数の総和とし，図 10 では，横軸をエージェント数，縦軸をコスト値とした．この比較では，既存手法と提案手法に加え，根エージェント以外のエージェントのみ提案手法を実行したものを比較した．提案手法では，各エージェントについて受信したコンテキスト

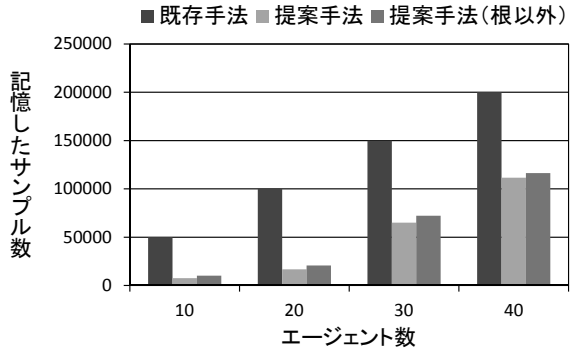

図 9: コスト値が小さい問題に対する記憶したサンプル数の比較

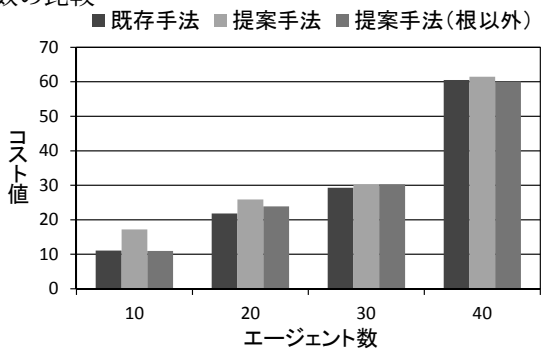

図 10: コスト値が小さい問題に対するコスト値の比較

と自身の値が同じサンプリング結果が複数ある場合に，部分木のコストが最小のサンプリング結果のみ記憶される．したがって，先祖エージェントを持たず，受信するコンテキストを持たない根エージェントは，他のエージェントと比較して多くのサンプリング結果が削減される．このことから，提案手法によりコスト値が大きく変化する問題がある場合は，これらの問題に対して，根エージェント以外のエージェントにおいて提案手法を実行することにより，記憶されるサンプル数が増加し，コスト値の変化を緩和できると期待される．

図 9 において，提案手法では既存手法と比較して，エージェント数 $|\mathcal{X}| = 10, 20$ の場合では約 85%，$|\mathcal{X}| = 30, 40$ の場合では約 45〜55%のサンプル数が削減された．また，根エージェント以外のエージェントにおいて提案手法を実行したものに関しては，根エージェントのみ既存手法を実行するため，提案手法と比べて，記憶したサンプル数の総和は約 5,000 程度増加している．図 10 において，提案手法では既存手法と比較して，エージェント数 $|\mathcal{X}| = 10, 20$ の場合を除き，削減されたサンプル数に対して，コスト値の大きな変化はないことが示された．また，$|\mathcal{X}| = 10, 20$ に関しても，根以外のエージェントにおいて提案手法を実行することにより，コスト値の変化を緩和できることが示された．

6. 考察

本章では，前章の評価により得られた結果から，既存手法と提案手法における $L^t_{a,d}$ の値の変化について考察する．提案手法では，各エージェントについて受信したコンテキストと自身の値が同じサンプリング結果が複数ある場合に，部分木のコストが最小のサンプリング結果のみ記憶される．したがって，各変数の値域の要素の値それぞれにおいて $\tau^t_{a,d}$ の値は 0 または 1 であり，また，τ^t_a の値は $0 \leq \tau^t_a \leq |D_i|$ である．以上により，$\tau^t_{a,d} = 1$ という前提ならば，$L^t_{a,d}$ の値は式 (6.1) のように表される．

$$L^t_{a,d} = \sqrt{2\lambda_a \ln \tau^t_a} \tag{6.1}$$

ここで，前章の評価において既存手法では，$L^t_{a,d}$ の値の理論上の最大値は，$\tau^t_{a,d} = 1$，$\tau^t_a = 5,000$ の時，$\sqrt{2\lambda_a \ln 5,000} \simeq 4.12\sqrt{\lambda_a}$ となる．また，前章の評価において，提案手法では，エージェント数 $|\mathcal{X}|$ を変化させた場合，各変数の値域の大きさは $|D_i| = 20$ である．したがって，$L^t_{a,d}$ の理論上の最大値は $L^t_{a,d} = \sqrt{2\lambda_a \ln 20} \simeq 2.44\sqrt{\lambda_a}$ である．その一方で，各変数の値域の大きさ $|D_i|$ を変化させた場合では，各変数の値域の大きさの最大値は $|D_i| = 50$ である．したがって，$L^t_{a,d}$ の理論上の最大値は $L^t_{a,d} = \sqrt{2\lambda_a \ln 50} \simeq 2.79\sqrt{\lambda_a}$ である．

しかし，コスト値の小さい問題を除いて，既存手法と提案手法においてコスト値に大きな変化がなかった問題に関しては，式 (3.3) のバウンド $B^t_{a,d}$ の計算に影響する $\hat{\mu}^t_{a,d}$ の値は，根に近い変数ノードにおいて 80 以上である．式 (3.3) よりこの値に $L^t_{a,d}$ の値を引いたとしても，λ_a の値が極端に大きくなければ，既存手法と提案手法において大きな差はない．したがって，既存手法と提案手法においてサンプリングに大きな影響はなく，コスト値にも大きな変化はなかったと考えられる．

コスト値が小さい問題に関しては，$\hat{\mu}^t_{a,d}$ の値は，根に近い変数ノードにおいても小さい値である．そのため，既存手法と提案手法において，式 (3.3) よりこの値に $L^t_{a,d}$ の値を引いた値は，コストが大きい問題と比較して，相対的に大きな差が出やすくなる．これにより，エージェント数 $|\mathcal{X}| = 10, 20$ の場合において，既存手法と提案手法を比較してコスト値に大きな変化があったと考えられる．その一方で，エージェント数 $|\mathcal{X}| = 30, 40$ の場合では，既存手法と提案手法においてコスト値に大きな差はない．これは，サンプリングの削減に伴い，$L^t_{a,d}$ の値が変化し，バウンド $B^t_{a,d}$ が変化する．これにより，各エージェントが選択する変数値が変化する可能性がある．しかし，これらの問題におけるコスト関数値の範囲は 0〜1 である．したがって，1 つのコスト関数あたりのコスト値の影響が小さいため，系全体のコスト値の合計への影響も小さくなったと考えられる．また，根エージェント以外のエージェントのみ提案手法を実行することにより，コスト値の変化が緩和された．これは，根エージェントのみ，十分なサンプリングをすることにより，根エージェントが関係するコスト関数のコスト値の合計が小さくなったためであると考えられる．

7. まとめ

本論文では，分散制約最適化問題の非厳密解法の DUCT の問題点を改善するために冗長なサンプリング結果を削減する手法を提案した．すなわち，各エー

ジェントにおいて，受信したコンテキストと自身の選択した変数値が同じサンプリング結果のうち，コストが最小のもの以外を削減することにより，各エージェントが記憶するサンプリング結果を削減した．実験により，エージェント数を変化させた場合，各変数の値域を変化させた場合及びコスト値が小さい問題において，記憶したサンプリング結果とコスト値について既存手法と提案手法を比較した．実験の結果から，コスト値が大きな問題に関しては，提案手法では既存手法と比較して，削減されたサンプル数に対して，コスト値の大きな変化はないことが示された．コスト値が小さい問題に関しては，根エージェント以外のエージェントに対して提案手法を実行することにより，コスト値の変化を緩和できることが示された．

今後の研究課題として，異なる問題に対して既存手法と提案手法を比較することが挙げられる．本論文では，ランダムなグラフ構造に対してのみ実験を行ったが，実ネットワークに近いスケールフリーグラフを含む，特定の構造を持つ問題について評価する必要がある．さらに，提案手法が有効な問題に対して，より多くのサンプル数を削減する手法の検討が挙げられる．また，提案手法による計算時間のオーバヘッドが，サンプリング回数の増加に対して，どの程度まで実際的であるかの調査が挙げられる．

謝辞

本研究の一部は，科研費基盤研究 (C)25330257 による．

参考文献

[1] 泉井良夫, 高野富裕, 小島康弘. スマートグリッドの基礎知識. 設備と管理, Vol. 44, No. 6, pp. 31–41, 2010.

[2] Sam Miller, Sarvapali D Ramchurn, and Alex Rogers. Optimal decentralised dispatch of embedded generation in the smart grid. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1, pp. 281–288. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[3] Norimichi Ukita. Real-time dense communication among agents for active tracking. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 1335–1336. ACM, 2005.

[4] 浮田宗伯. 能動視覚エージェント群の密な情報交換による多数対象の実時間協調追跡. 電子情報通信学会論文誌 D, Vol. 88, No. 9, pp. 1438–1447, 2005.

[5] 浮田宗伯, 松山隆司. 能動視覚エージェント群による複数対象の実時間協調追跡. 情報処理学会 CVIM 研究会論文誌, Vol. 43, No. 11, pp. 64–79, 2002.

[6] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. Technical report, 2005.

[7] Pragnesh J. Modi, W Shen, Milind Tambe, and Makoto Yokoo. ADOPT:Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal(AIJ)*, Vol. 161, pp. 149–180, 2005.

[8] Alex Rogers, Alessandro Farinelli, Ruben Stranders, and Nicholas R Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, Vol. 175, No. 2, pp. 730–759, 2011.

[9] 沖本天太, ジョヨンジュン, 岩崎敦, 横尾真. 擬似木に基づく分散制約最適化問題の精度保証付き非厳密解法の提案. 情報処理学会論文誌, Vol. 52, No. 12, pp. 3786–3795, 2011.

[10] Weixiong Zhang, Guandong Wang, and Lars Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *Proceedings of AAAI Workshop on Probabilistic Approaches in Search*, 2002.

[11] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. DUCT: An upper confidence bound approach to distributed constraint optimization problems. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 1, pp. 528–534, 2012.

[12] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 167–174. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[13] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, Vol. 47, No. 2-3, pp. 235–256, 2002.

[14] 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*, Vol. 2, pp. 639–646. International Foundation for Autonomous Agents and Multiagent Systems, 2008.