平成24年度 修士論文

分散制約最適化問題の非厳密解法の改良に関する研究

指導教官
松尾 啓志 教授
松井 俊浩 准教授

名古屋工業大学大学院工学研究科

創成シミュレーション工学専攻
平成23年度入学　23413574番

柳田 大輝

平成25年2月5日

目 次

第 1 章　はじめに **1**

第 2 章　分散制約最適化問題とその解法 **4**

2.1　分散制約最適化問題 (DCOP) 4

2.2　分散制約最適化問題の解法 5

2.2.1　厳密解法 5

2.2.2　非厳密解法 6

第 3 章　従来手法: KOPT アルゴリズム **7**

3.1　k-Optimal 7

3.2　KOPT アルゴリズムの動作 7

3.3　KOPT アルゴリズムの問題点 12

第 4 章　KOPT アルゴリズムを改良する提案手法: 探索の多重化 **14**

4.1　多重化の方法と同期周期 14

4.2　割当ての同期のための通信 15

4.3　通信のオーバーラップ 16

第 5 章　評価: KOPT 探索の多重化の効果 **18**

5.1　問題の生成方法 18

5.2　KOPT アルゴリズムとの比較 19

5.3　割当ての同期時に選択される探索のパラメータ 20

5.4　同一パラメータによる多重化との比較 21

5.5　割当ての同期に必要な通信時間の隠蔽の効果 21

第 6 章　従来手法: DUCT アルゴリズム **25**
6.1　DUCT アルゴリズムの動作 25
6.2　DUCT の保存するコンテキスト 28

第 7 章　DUCT を改良する提案手法 1: コンテキスト中の必要な変数値の保存 **30**
7.1　基本的なアイデア 30
7.1.1　コストの計算に必要な変数値 31
7.1.2　効果が高い場合と低い場合 33
7.1.3　解に与える影響 34

第 8 章　DUCT を改良する提案手法 2: 保存するコンテキスト量の制限 **35**
8.1　追い出すコンテキストの選択基準 35
8.2　解の悪化 36

第 9 章　評価: DUCT のコンテキスト中の必要な変数値の保存による効果 **37**
9.1　問題の生成方法 37
9.2　グラフの構造による削減効果の違い 38
9.3　DUCT アルゴリズムとの比較 38

第 10 章 評価: DUCT の保存するコンテキスト量の制限 **41**

第 11 章 おわりに **45**

参考文献 **48**

第1章

はじめに

近年，複数の自律的なエージェントが協調して動作する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は，集中して行われる処理に比較して，耐障害性やスケーラビリティの点において優れている．このようなマルチエージェントシステムにおける協調問題は，分散制約最適化問題として定式化できる．[1, 2]

分散制約最適化問題を解くアルゴリズムは，厳密解法と非厳密解法に分類される．厳密解法は，必ず最適解を得ることができるが，エージェント数や制約数が増えて問題の規模と複雑さが大きくなると，計算量やメッセージサイズなどが指数関数的に増大するという問題がある．それに対して非厳密解法は，最適解を必ず発見するとは限らないが，計算量やメッセージサイズなどを比較的小さく抑えることができる．そのため，規模が大きく複雑な問題に対しては非厳密解法が適用される．厳密解法には，ADOPT[1] や DPOP[3] などがある．非厳密解法には，DSA[4]，Max-Sum[2],KOPT[5],DUCT[6] などがある．

本研究では，第一に，KOPT アルゴリズムに注目する．これまで，非厳密解法によって得られる解の品質については，あまり議論がされてこなかった．例えば，DSA では，得られる解の品質は保証されない．近年，k-Optimality という指標が提案された [7]．k-Optimal な解とは，k 個以下のエージェントが持つ変数値を変更しても改善しない解である．任意の k についての k-Optimal な解を導くことができる近似解法として KOPT アルゴリズムが提案された．

従来の KOPT アルゴリズムでは，パラメータ k が大きいほど広範囲のエージェント

と通信を行うので，より長い探索時間を必要とする．つまり，得られる解の品質と探索時間についてのトレードオフが存在する．本研究では，KOPT アルゴリズムに対して複数の異なるパラメータ k を設定し，探索を多重化することにより，解品質と探索時間抑制を両立する手法を提案する．計算機シミュレーションによる評価に基づいて，提案手法の有効性を確認した．

本研究では，第二に，分散制約最適化問題の非厳密解法の１つである DUCT アルゴリズムに注目する．DUCT アルゴリズムの特徴は，木探索にサンプリングを導入したことである．解の候補が多く全探索が不可能な問題に対して，サンプリングを行うことにより，精度の良い解が得られることが期待される．しかし，DUCT には，問題のグラフから構成した擬似木の深さが深くなると，各エージェントが使用するメモリ量が大きくなる問題がある．

この問題を解決する方法として，2 つの手法を提案する． 1 つ目は，エージェントが変数割り当てに対するコストの計算に冗長な値の保存をやめることで，DUCT のメモリ使用量を削減する手法である．2 つ目は，エージェントがサンプリングのために保存する変数のうち，今後使われる見込みの薄いものの保存をやめることで，DUCT のメモリ使用量を一定量に制限する手法である．1 つ目の手法は，解コストを悪化させることなくメモリ使用量を削減することができるが，削減できる条件に当てはまらないエージェントの場合は効果がない．2 つ目の手法は，全てのエージェントに対してメモリ量を制限できるが，制限したメモリ量が小さいと解コストが大きく悪化する可能性がある．

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，KOPT アルゴリズムとその改良手法について述べる．3 章では，従来手法である KOPT アルゴリズムについて説明する．4 章では，本研究で提案する KOPT アルゴリズムの探索を多重化する手法について述べる．5 章では，提案手法について計算機シミュレーションを行い，その結果を示す．

6 章から 9 章では，DUCT アルゴリズムとその改良手法について述べる．6 章では従来手法である DUCT アルゴリズムとその問題点について述べる．7 章および 8 章では，DUCT を改良する本研究で 提案する 2 つの手法について述べる．9 章および 10 章で

はDUCTアルゴリズムを改良する提案手法について計算機シミュレーションにより評価を行いその結果を示す．最後に，11章においてこれらをまとめる．

第2章

分散制約最適化問題とその解法

本章では，分散制約最適化問題 (Distributed Constraint Optimization Problems) の定式化と，現在研究されている分散制約最適化問題の解法について説明する．

2.1 分散制約最適化問題 (DCOP)

分散制約最適化問題は，制約最適化問題の変数をエージェントに分散して配置したものである．制約最適化問題は，与えられた制約を出来る限り満たすような変数値の割り当てを求める問題である．

分散制約最適化問題は，n 個の変数 $x_1, \cdots, x_n$ と制約の集合 C から構成される．各変数 x_i は有限で離散的な領域 D_i に含まれる値を取る．変数は複数のエージェントに分散して配置される．本研究では，1 つのエージェントに配置される変数は 1 つのみとする．エージェント A_i に対して割り当てられた変数を x_i と記述する．同じ変数が複数の異なるエージェントに配置されることは無いものとする．

各エージェントは，自身が持つ変数の値のみを直接知ることができる．また，各エージェントは自身が持つ変数の値のみを変更できる．各エージェントは，他のエージェントの持つ変数の値をメッセージ交換により取得する．エージェント A_i の管理する変数 x_i に割り当てられている変数値を a_i と表し，変数の組 $[x_1, \cdots, x_n]$ に対する割り当てを $a = [a_1, \cdots, a_n]$ と表す．本研究では，問題は二項制約のみを含むものとし，変数 x_i, x_j に関する制約を $c_{i,j}$ と表す．また，各制約 $c_{i,j}$ に対応する評価関数を $f_{i,j}$ と表す．$f_{i,j}$ は，変数 x_i, x_j の割当てに応じて評価値を返す．

本研究でエージェント同士がメッセージ交換に用いる通信路は，制約で関係する変数を持つ 2 つのエージェントの間にあるものとする．あるエージェントに関して通信路で結ばれたエージェントを,「隣接するエージェント」と呼ぶ．分散制約最適化問題の目的は，すべての制約についての評価値の和 (式 2.1) が最大となるような，大域的に最適な変数値の割当て a を求めることである．

$$R(a) = \sum_{f_{i,j} \text{ for all constraints}} f_{i,j}(a_i, a_j) \tag{2.1}$$

また，問題によっては，全ての制約についてのコスト値の和 (式 2.2) を最小とするような，変数値の割り当て a を求めることを目的とする．

$$Cost(a) = \sum_{f_{i,j} \text{ for all constraints}} f_{i,j}(a_i, a_j) \tag{2.2}$$

本研究では，KOPT アルゴリズムについて議論する場合は最大化問題である式 2.1 を用い，DUCT アルゴリズムについて議論する場合は最小化問題である式 2.2 を用いる．各エージェントは他のエージェントとメッセージ通信により情報を交換しつつ自身の変数値を変更し，探索処理を実行する．

2.2 分散制約最適化問題の解法

現在研究されている分散制約最適化問題について説明する．分散制約最適化問題を解くアルゴリズムは，得られる解の性質から，必ず最適解が得られる厳密解法と，最適解が得られるとは限らない非厳密解法の 2 つに分類される．

2.2.1 厳密解法

分散制約最適化問題の代表的な厳密解法には，ADOPT(Asynchronous Distributed Constraint Optimization)[1] や DPOP(Dynamic Programming Optimisation Protocol)[3] などがある．ADOPT や DPOP は，前処理として，半順序関係のある擬似木を構築する．続いて，擬似木の辺に沿って，メッセージ交換を行う分散アルゴリズムによって解空間を探索する．これらのアルゴリズムにより発見される解は，必ず最適解である

が，変数の数，値域，制約辺などで定義される問題の規模が大きくなると，計算量やメッセージサイズなどが指数関数的に増大し，解くことができなくなるという問題がある．

2.2.2 非厳密解法

分散制約最適化問題の非厳密解法には，DSA(Distributed Stochastic Algorithm)[4]，Max-Sum アルゴリズム [2]，KOPT アルゴリズム [5]，DUCT アルゴリズム [6] が挙げられる．

DSA アルゴリズムは，それぞれのエージェントが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．各エージェントは，隣接するエージェントの状態のみを考慮して自身の状態を決定するため，局所的な最適解に収束しやすい．また，複雑なグラフでは解の精度が低い．改良手法として，タブーリストを用いることで局所最適解からの脱出を行う DSTS[8] がある．

Max-Sum アルゴリズムは，factor グラフと呼ばれるグラフ上でメッセージを伝搬し，周囲の制約を考慮し状態を選択する．

KOPT アルゴリズムは，k-Optimal な解を得るため，k 個のエージェントから成るグループ内で状態について合意を形成して，解の改善を繰り返す．アルゴリズムの詳細については，第 3 章で述べる．

DUCT アルゴリズムは，擬似木を構成し，サンプリングによる探索を行う．アルゴリズムの詳細については，第 6 章で述べる．

第3章

従来手法: KOPTアルゴリズム

近年，非厳密解法によって得られる解の品質の指標として，k-optimality が提案された [7]．本節では，最適性の指標である k-optimal について説明する．次に，既存手法である KOPT アルゴリズム [5] の特徴と動作について述べる．

3.1 k-Optimal

k-Optimal とは，分散制約最適化問題における解の最適性の指標である．k-Optimal な解とは，k 個以下のエージェントが持つ変数値を変更しても，評価値 $R(a)$ をそれ以上向上させることができない解である．

変数の割当て a と $\tilde{a}$ の間で，異なる変数値を取る変数の数を $d(a, \tilde{a})$ と表現する．k-Optiml な割当て a は，式 3.1 のように定義される．

$$R(a) - R(\tilde{a}) \geq 0 \forall \tilde{a} \ such \ that \ d(a, \tilde{a}) \leq k \tag{3.1}$$

この指標を用いると，n 個の変数を持つ制約最適化問題を厳密解法により解いた場合の解は n-Optimal であるといえる．

3.2 KOPTアルゴリズムの動作

KOPT アルゴリズムは，分散制約最適化問題の非厳密解法であり，1 以上 n 以下の任意の k についての k-Optimal な解を導くアルゴリズムである．また，評価値がメッ

セージ通信回数に伴い，単調に増加する．そのため，任意時間アルゴリズム (anytime algorithm) の一種であると言える．

KOPT アルゴリズムのメッセージ通信は，全てのエージェントが大域的な同期の下で動作する．また，各エージェントには，変数名などの順序関係のある ID があらかじめ与えられているとする．

KOPT アルゴリズムは，3 つのフェーズからなる計算を反復処理の単位として，繰り返し処理を実行する．各フェーズでの処理内容は，次のようになる．

- フェーズ 1 周囲のエージェントと情報の交換
- フェーズ 2 エージェント同士でグループを構成し最適な割当てを計算
- フェーズ 3 グループ内のエージェントで合意を形成し変数値を更新

各フェーズの動作の詳細を順に説明する．

フェーズ 1 – 情報の交換

はじめに，各エージェントは $\lfloor (k+2)/2 \rfloor$ ホップ先までのエージェントの持つ変数値とそれに関する制約の情報を取得する．そのために，各エージェントは自身の変数値および関連する制約の情報を隣接するエージェントに送信する．同時に，周囲のエージェントから受信した情報を中継する．$\lfloor (k+2)/2 \rfloor$ ホップ先まで情報を送るので，このフェーズには，$\lfloor (k+2)/2 \rfloor$ ステップを必要とする．

KOPT の実行例を 3.1 に示す．ここで $k = 2$ であるとする．変数 x_2 を管理するエージェント A_2 に注目すると，A_2 が入手する情報は変数 $x_1, x_3, x_4, x_5, x_6, x_8$ の値である．この情報は，次のように A_2 に伝えられる．まず 1 ステップ目に A_2 は，隣接するエージェントである A_1, A_3, A_5 と通信を行い，その変数 x_1, x_3, x_5 を取得する．また，このとき A_1, A_2, A_5 も，それぞれ隣接するエージェントの情報を入手する．2 ステップ目には，A_2 は再び隣接するエージェント A_1, A_3, A_5 と通信することで，それらのエージェントが 1 ステップ目に入手した変数 x_4, x_6, x_8 を入手する．

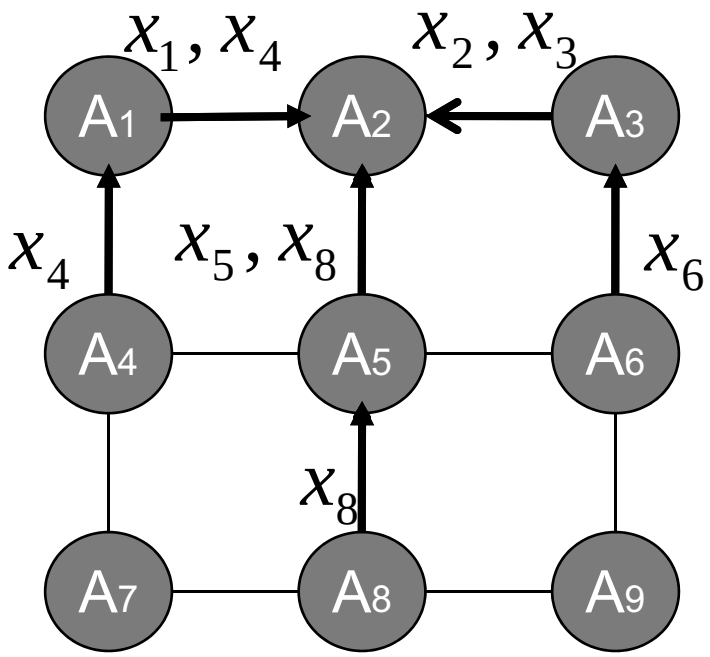

図 3.1: エージェント A_1 が受信する変数値（$k = 2$ のとき）

フェーズ 2 – グループの構成と割当ての計算

各エージェントはフェーズ 1 で情報を入手できた変数を持つエージェント，すなわち自身を中心に $\lfloor (k+2)/2 \rfloor$ ホップ中から，$k-1$ 個のエージェントを選択する．選択の方法は，自身からグラフ上の距離が短いエージェントから順番に $k-1$ 個をランダムに選択する．このとき選択されたエージェントと自身のエージェントを，動的エージェント (active agent) と呼ぶ．そして，この動的エージェントに隣接している動的エージェントでないエージェントを静的エージェント (static agent) と呼ぶ．動的エージェントと静的エージェントを合わせてグループと呼ぶ．グループに所属するエージェントを，そのグループのメンバーと呼ぶ．つまりこの時，各エージェントは自身を中心とするグループを持っている．中心となるエージェントを，そのグループのメディエータ (mediator) と呼ぶ．すべてのエージェントが，自身がメディエータを務めるグ

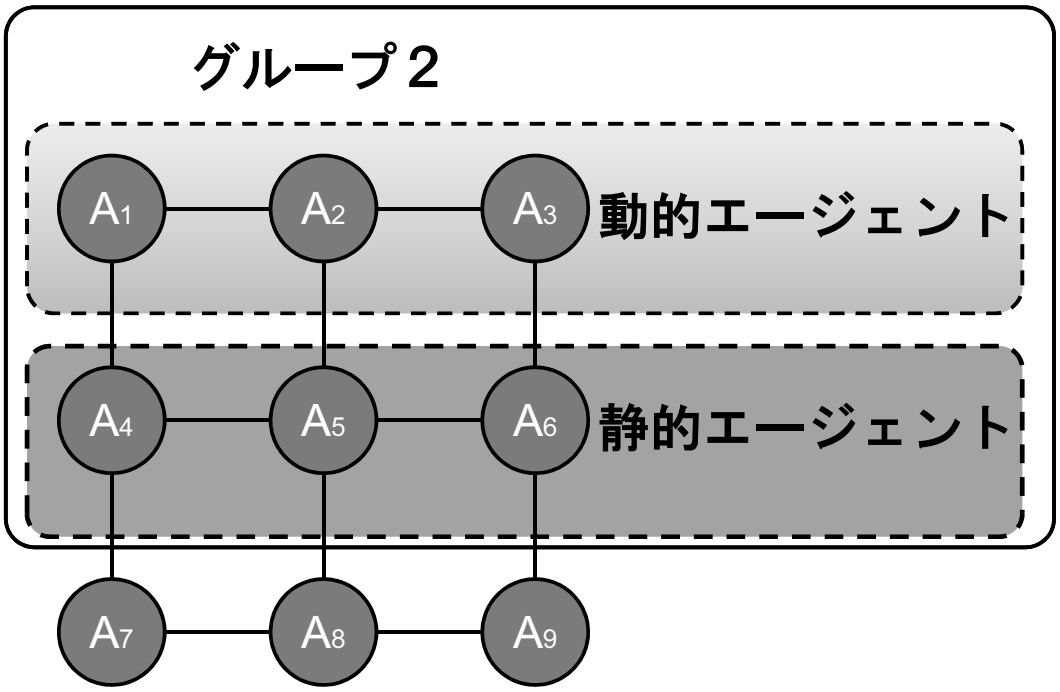

図 3.2: A_2 を中心とするグループ２の一例 (k=3 のとき)

ループを構成するので，グループはエージェントの数だけ存在することになる．また，エージェントはメディエータであると同時に，他のエージェントがメディエータを務めるグループのメンバーでもある．

3.2 を例に，$k = 3$ である時の，グループ，動的エージェント，静的エージェントの関係を説明する．エージェント A_2 に注目すると，A_2 を中心に $k-1$ 個のエージェントを選ぶ場合，$[A_1, A_3], [A_1, A_5], [A_3, A_5]$ の 3 通りがある．ここでは，$[A_1, A_3]$ が選択されたとする．この A_1, A_3 と自身の A_2 を，グループ 2 の動的エージェントと呼ぶ．また，中心となった A_2 をグループ 2 のメディエータと呼ぶ．そして，これらの動的エージェントに接している A_4, A_5, A_6 をグループ 2 の静的エージェントと呼ぶ．$A_1, A_2, A_3, A_4, A_5, A_6$ をグループ 2 のメンバーと呼ぶ．まとめると，3.1 のようになる．

表 3.1: グループ 2 の例 (k=3 のとき)

役割	エージェント
メディエータ	A_2
動的エージェント	A_1, A_2, A_3
静的エージェント	A_4, A_5, A_6
メンバー	$A_1, A_2, A_3, A_4, A_5, A_6$

メディエータは，フェーズ 1 で取得した情報を基に，グループ内の変数に対して最適な割当てを求める．このとき，動的エージェントの持つ変数値のみを変更可能，静的エージェントの持つ変数値は変更不可能と仮定して最適な割当てを計算する．この計算には，制約最適化問題を解く任意の厳密解法を用いることができる．ここでは，グループ内の変数に対する最適な割当てを計算するにとどめ，実際には変数値を変更しない．

続いて，この最適な割当てに変更した場合にグループ全体として増加する評価値と，グループ内の各エージェントが持つ変数に対する割当てをグループ内のエージェントに対して送信する．この情報もフェーズ 1 と同様に，エージェントが中継して伝えるので，$\lfloor (k+2)/2 \rfloor$ ステップを要する．

フェーズ 3 – 合意の形成と変数値の更新

フェーズ 3 では最初に，各エージェントがフェーズ 2 で受信した情報から，最も評価値の増加が大きくなるグループを選択する．このとき，複数のグループが選択される場合は，メディエータとなるエージェントの ID が最も小さいグループを選択する．

続いて，選択したグループが提示した自身の変数に対する割当てを，次に取る予定の変数値とする．この次に取る予定の変数値を選択したグループのメンバーのうちの動的エージェントに送信する．この送信には，$k-1$ ステップを要する．

最後に，自身が選択したグループの提示する割当てを，そのグループのメンバーのうち全ての動的エージェントが次にとる予定の変数値としていた場合，実際に自身の変数値を次にとる予定の変数値に変更する．そうではなかった場合は，変数値の変更は行わない．つまり，グループ内の動的エージェントと次にとる変数値についての合

表 3.2: 各フェーズに必要な通信ステップ数

フェーズ	ステップ数
1	$\lfloor \frac{k+2}{2} \rfloor$
2	$\lfloor \frac{k+2}{2} \rfloor$
3	$k-1$

意がとれた場合にのみ変数値を更新する．この 3 つのフェーズを 1 回の反復処理単位として繰り返すことで，k-Optimal な解に近づく．1 回の反復処理に必要な通信ステップ数は k に依存し,（式 3.2）のように表される．これは，フェーズ 1 からフェーズ 3 で必要な通信ステップ数 3.2 の合計である．

$$2 \times \lfloor \frac{k}{2} \rfloor + k + 1 \tag{3.2}$$

3.3 KOPT アルゴリズムの問題点

KOPT アルゴリズムに存在するパラメータ k は，k を n に近づけるほど最適解に近い解が得られる．しかし，k を大きくすると，1 回の反復処理に必要なメッセージ通信ステップ数が，k に関する式 3.2 により増大する．変数値の更新は 1 回の反復処理の最後に行われるフェーズ 3 で実行されるので，k を大きくする程，変数の割当てを更新する周期が長くなることになる．k に対するアルゴリズムの 1 回の反復処理に必要な通信ステップ数を 3.3 に示す．

また，パラメータ k の値を大きくするほどフェーズ 2 で 1 つのグループに含まれるエージェントの数が増えることから，合意を形成することが難しくなる．この 2 つの理由により，パラメータ k の値を大きくすると，解の更新頻度が下がり，評価値 $R(a)$ の向上が遅くなると考えられる．

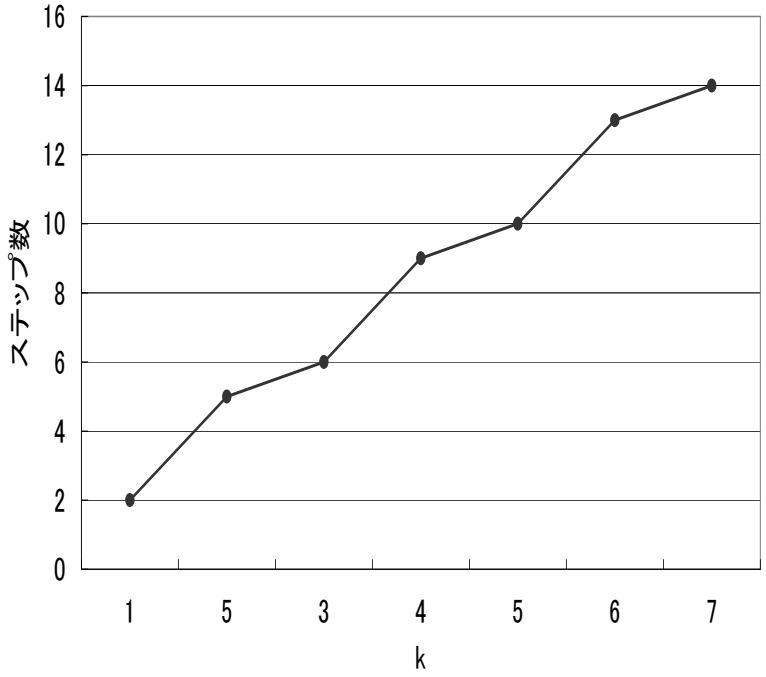

図 3.3: k に対する 1 回の反復処理に必要な通信ステップ数

第4章

KOPTアルゴリズムを改良する提案手法: 探索の多重化

本研究では，KOPT アルゴリズムを拡張し，パラメータ k の異なる複数の探索が平行するような探索の多重化を提案する．これにより，前節で述べた既存の KOPT アルゴリズムの問題点を解消し，解品質と探索時間抑制を両立させる．

アルゴリズムに対して個別にパラメータを設定した探索を多重に動作させ，もっともよい解を採用する手法は，アルゴリズムポートフォリオ (Algorithm Portfolio) として研究されている [9, 8].

本研究では，KOPT アルゴリズムのパラメータ k と，得られる評価値の関係に注目して，1 種類のアルゴリズムに異なるパラメータを設定する探索の多重化を行った.

4.1 多重化の方法と同期周期

KOPT アルゴリズムに対してパラメータ k の異なる探索を多重化する方法について説明する．既存の KOPT アルゴリズムは，パラメータ k を 1 つだけ選択して実行する．これに対して，提案する多重化を行う手法では，複数のパラメータを選択し並列に実行する．そして，一定のステップ数毎に，それぞれのパラメータで実行した探索結果のうち，最も高い評価値 $R(a)$ を獲得した変数の割当て a を選択し，他のパラメータで実行した探索へこの割当て a をコピーし探索を再開する．この操作を「割当ての同期をとる」と呼ぶ.

この操作を一定周期毎にフェーズ 3 の後に行う．これは，各エージェントにおいてフェーズ 3 の最後に変数値の更新が行われるためである．また，割当ての同期をとる周期は，事前に選択したパラメータ k それぞれの 1 回の反復処理に必要なステップ数の公倍数で行う．これは，割当ての同期をとるためのフェーズ 3 の終わりを一致させるためである．例として，多重化して実行する探索のパラメータ k として 1,2,5 を選択した場合の割当ての同期は，10 ステップの整数倍毎に行う．これは，1 回の反復処理には，式 3.2 より，$k = 1$ の場合に 2 ステップ，$k = 2$ の場合に 5 ステップ，$k = 5$ の場合に 10 ステップであり，その最小公倍数が 10 ステップの通信を必要とするからである．この場合，1 回の同期までに行われる解の最大更新回数は，$k = 1$ の場合に 5 回，$k = 2$ の場合に 2 回，$k = 5$ の場合に 1 回である．

4.2 割当ての同期のための通信

割当ての同期をとるためには，多重化して実行した各探索についてそれぞれの評価値 $R(a)$ を，各エージェントが把握する必要がある．しかし，分散制約最適化問題では，各エージェントは直接他のエージェントの状態を知ることができないので，直接的に $R(a)$ を求めることができない．そのため，各エージェントは通信により他のすべてのエージェントの状態を取得し，$R(a)$ を求める．そこで，全エージェントの変数の割当て a を取得するための通信を行う．全てのエージェントが，他の全てのエージェントの持つ変数の状態を取得するために必要な通信ステップ数は，問題の制約グラフの直径 d に等しい．この理由は，最も多くの通信ステップ数を必要とするのが，制約グラフ上で最大頂点間距離を持つエージェントの組が互いの状態を取得する場合であるためだ．また，初回の割当ての同期のための通信に限り，関係する制約関数 $f_{i,j}$ の情報も，変数の状態に合わせて送信する．この割当ての同期のための通信に必要なステップ数を加味した場合の，1 回の反復処理に必要な通信回数は式 4.1 に示すとおりである．k_{max} は，多重に実行する探索のパラメータ k の最大値である．このとき，全てのエージェントは，制約グラフの直径 d を事前に与えられているとする．

$$2 \times \lfloor \frac{k_{max}}{2} \rfloor + k_{max} + 1 + d \tag{4.1}$$

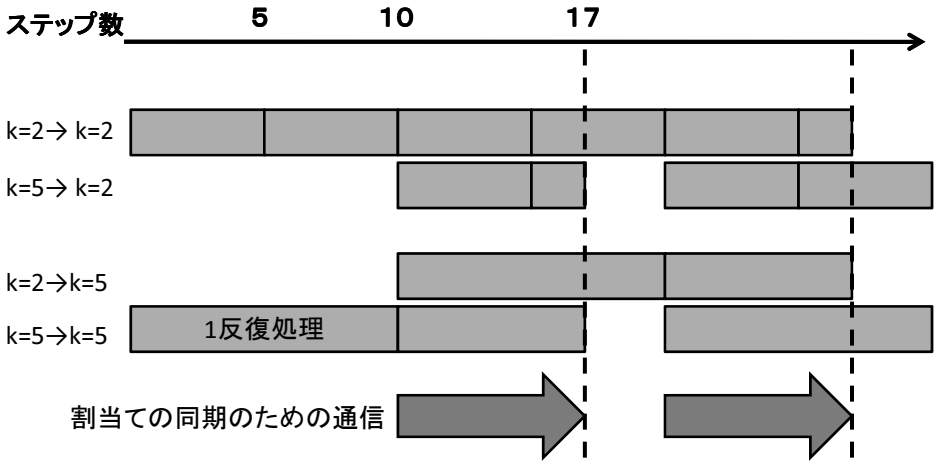

図 4.1: 通信をオーバラップさせた場合の動作概要

4.3 通信のオーバーラップ

この方法では，既存の KOPT アルゴリズムに比べて解の更新が，1 反復処理毎に d ステップ遅れることになる．そこで，割当ての同期をとるための通信を探索とオーバーラップさせることで，同期をとるための通信に必要なステップ数を隠蔽する手法を導入する．この手法では 1 反復処理に必要なステップ数を，割当ての同期をとるタイミングで通信を行わずに $R(a)$ を求めることができるとした場合と同じステップ数（式 3.2）に抑えることができる．

アルゴリズムの動作を，制約グラフの直径が 7 である問題に対して，$k = 2$ と $k = 5$ を設定し探索を多重に実行する場合を例に説明する．この時，$k = 2$ を設定した探索の 1 反復処理に必要なステップ数は，式 4.1 より 5 ステップ，同様に，$k = 5$ を設定した場合は 10 ステップである．よって，割当ての同期をとる周期は，5 と 10 の最小公倍数である 10 ステップとする．まず，$k = 2$ と $k = 5$ による探索を開始し多重に実行する．$k = 2$ を設定した探索が反復処理を 2 回，$k = 5$ を設定した探索が 1 回終えたとき，通信ステップ数は開始から 10 ステップである．そこで，割当ての同期のための通信を

開始する．同時に次の反復処理を開始する．このとき，多重に実行する探索は，$k = 2$ を設定した探索により発見された割当てを採用する場合と，$k = 5$ による割当てを採用する場合を仮定し，4 重に実行する．つまり，$k = 2$ による探索が発見した割当てを引き継いで $k = 2$ で行う探索と $k = 5$ で行う探索，$k = 5$ による探索が発見した割当てを引き継いで $k = 2$ で行う探索と $k = 5$ で行う探索の 4 種類である．続いて，制約グラフの直径が 7 であるので，開始から 17 ステップ目には全エージェントが割当ての同期に必要な他のエージェントの持つ変数値の状態 a を取得している．ここで，各エージェントは 2 重に実行を開始した探索それぞれについての評価値 $R(a)$ を計算する．この $R(a)$ の計算結果は，10 ステップ時点での割当て a に対するものである．最も高い評価値 $R(a)$ を導いた探索の割当てを引き継いだ探索を残して，他の探索の実行を破棄する．$R(a)$ を計算した結果，10 ステップ時点で最も高い評価値 $R(a)$ を導く割当てを発見した探索が $k = 2$ による探索であったとすると，この割当てを引き継いで実行中の $k = 2$ による探索と $k = 5$ による探索は引き続き実行を行い，$k = 5$ による割当てを引き継いだ探索は実行を中止する．以降も同様の方法で割当ての同期を行う．動作の様子を 4.1 に示す．

ただし，この方法で同期を行う場合は，割当ての同期を行う周期より少ない通信ステップ数で割当ての同期のための通信を終える必要がある．そのため，制約グラフの直径 d は式 4.2 を満たさなければならない．

$$d \leq 2 \times \lfloor \frac{k_{max}}{2} \rfloor + k_{max} + 1 \tag{4.2}$$

第5章

評価: KOPT探索の多重化の効果

本章では，提案する探索の多重化による効果を計算機シミュレーションにより検証する．既存アルゴリズムとの比較は，通信ステップ数に対する評価値 $R(a)$ の推移と最終的な評価値 $R(a)$ の観点から行った．分散アルゴリズムの性能の指標としては，メッセージ交換の通信回数が用いられており [1, 8]，本論文における評価でもこの指標を採用した．これは，各エージェントの計算時間が 1 回のメッセージ通信時間に比べて，十分に短いとの仮定に基づく．

5.1 問題の生成方法

今回の実験では，不規則な制約の構造を持つ問題として，頂点をランダムに選択して辺でつないだグラフを用いた．頂点がエージェントに，辺が制約に相当する．

問題の生成方法は，まずエージェント数 n と制約数 m を設定し，辺が m 本に達するまで，ランダムに 2 頂点 (v_i, V_j) を選択し辺 $e_{i,j}$ でつなぐ．この時，すでに辺でつながれている頂点の組は選択しないものとする．辺 $e_{i,j}$ に対応する評価関数 $f_{i,j}$ は，頂点 v_i, v_j に対応する変数 x_i, x_j の組み合わせに対して，1 から 100 の整数値から同じ値を選ばないようにランダムに評価値を割り当てる．評価関数は，制約辺ごとに生成した．また，変数の値域は 3 とした．この問題の生成方法は，[5] を参考にした．

例として，5.1 に示すような評価関数が生成される．この評価関数 $f_{i,j}$ は，$x_i = 1$ か

表 5.1: 評価関数の例

$x_i \setminus x_j$	0	1	2
0	33	69	39
1	99	55	43
2	66	62	27

表 5.2: 生成した問題

エージェント数	1000
制約数	3000
直径	7(80%),8(20%)

つ $x_j = 0$ のとき，最大の評価値 99 を返す.

5.2 KOPT アルゴリズムとの比較

提案する多重化した KOPT アルゴリズムと既存の KOPT アルゴリズムとの比較を行った．対象とする問題は，エージェント数 $n = 1000$，制約数はエージェント数の 3 倍の $m = 3000$ として，前節の方法で 10 個生成した．生成した 10 個の問題のうち，直径が 7 のグラフが 8 個，8 のグラフが 2 個であった．以降の比較も同じ条件で行う.

提案手法では，3 個のパラメータ k に 1,2,5 を設定し，10 サイクル毎に割当ての同期を行う事とした．割当ての同期を行う周期は，制約グラフのうち最大の直径よりも長くする必要があることと，多重に実行する探索のパラメータ k の 1 回の反復実行に必要なステップ数の公倍数である必要があることから 10 サイクルを選択した（パラメータ $k = 1, k = 2, k = 3$ の 1 反復実行に必要なステップ数は，式 3.2 よりそれぞれ 2,5,10 ステップである）．各アルゴリズムは，1 つの問題インスタンスにつき 100 回ずつ実行し，合計 1000 回の実行結果のステップ毎の評価値の平均を集計した．また，KOPT アルゴリズムは停止機構を持たないので，シミュレータにより一定ステップ数で停止させることとした．停止させるステップ数は，評価値が十分に収束すると考えられる 500 ステップとした.

ステップ数に対する評価値の推移を 5.1 に示す．グラフの横軸がステップ数，縦軸が評価値を表し，評価値の値が大きいほどよい解であることを表す．パラメータ $k = 1$ を設定した既存の KOPT は，早くから評価値が上昇するが，40 ステップ目付近で評価値の上昇がほぼ止まってしまう．$k = 2$ を設定した探索の場合は，$k = 1$ を設定した場合に比べて，評価値の上昇が遅いが，最終的には $k = 1$ の場合に比べて高い評価値が得られる．同様に，$k = 5$ の場合は，$k = 1$ や $k = 2$ の場合に比べて評価値の上昇は遅いが，最終的に得られる評価値はいずれの場合よりも高い．

これに対し，提案した多重化を行う方法では，既存の KOPT アルゴリズムに $k = 1$ を設定した場合と同様に，探索の早い時期から評価値が大きく上昇する．そして評価値は単調に上昇する．最終的に得られた解の評価値を 5.2 に示す．最終的に得られる解は既存の KOPT アルゴリズムに $k = 5$ を設定した場合とほぼ同等である．

$k = 1$ や $k = 2$ により求めた割当てに同期をとることで，局所最適に陥ってしまい最終的に得られる評価値が，$k = 5$ のみで探索を行った場合よりも低くなる可能性を心配したが，そのような事は起こらなかった．これは，KOPT アルゴリズムの性質から，より大きい k を用いて探索を実行することで，より広い範囲のエージェント間で調整を行うようになるので，局所解から脱出できるためであると考えられる．

5.3 割当ての同期時に選択される探索のパラメータ

割当ての同期を行い割当てが更新される場合に，どのパラメータ k を指定した探索の割当てが選択されているかを調査した．結果を 5.3 に示す．横軸がステップ数，縦軸が割当ての同期をとる際に選択された探索のパラメータ k の割合を示す．探索の開始直後は小さな $k = 1$ を設定した探索の割当てが採用される割合が大きいが，探索が進むに連れて大きな k である $k = 5$ を設定した探索により得られた割当てが採用される割合が大きくなる．その後，徐々に $k = 1$ による割当てが採用される割合が上昇し，$k = 5$ による割当てが採用される割合と拮抗する．

5.4 同一パラメータによる多重化との比較

本論文で提案した手法は，複数の異なるパラメータ k を指定して探索を多重に行うものであるが，同一のパラメータ k を指定して探索を多重に行った場合との比較を行った．結果を 5.4 に示す．同一パラメータで多重化を行った場合は，既存の KOPT アルゴリズムに比べて最終的に得られる解の評価値に向上がみられるものの，ステップ数に対して評価値が向上する割合はあまり変わらなかった．よって，同一パラメータによる探索を多重化するよりも，異なるパラメータによる探索を多重化することが有効であるといえる．

5.5 割当ての同期に必要な通信時間の隠蔽の効果

4.3 で提案した割当ての同期に必要な通信時間を隠蔽する方法について，その有効性を確認するために，同期と探索のオーバーラップを行わない方法と提案手法の比較を行った．その結果を 5.5 に示す．オーバラップを行う場合は，行わない場合に比べて，評価値が早く上昇していることが確認できる．オーバーラップを行う場合は評価値が上昇するタイミングが一定周期（10 ステップ）で発生するのに対して，行わない場合が一定周期でないのは，評価に用いた制約グラフの直径が一定でないためである．これは，オーバーラップを行う場合は，割当ての同期を行う周期（式 4.1）が制約グラフの直径 d に依存し，評価に用いた問題にはグラフの直径が 7 のものと 8 のものの 2 種類が存在したためである．

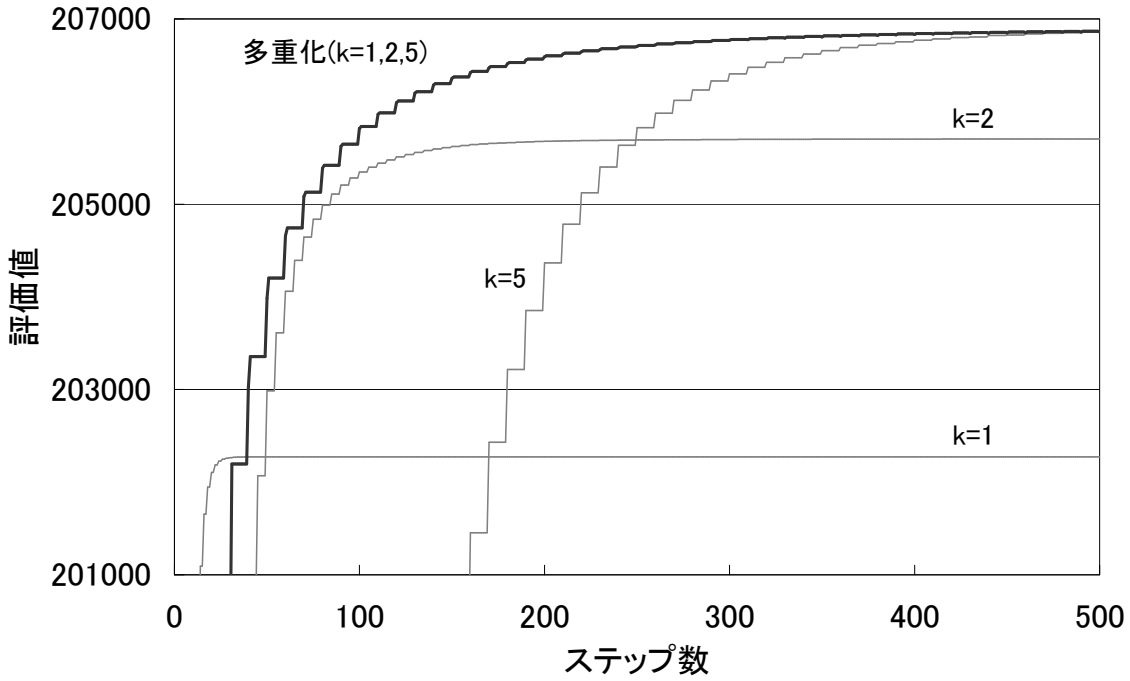

図 5.1: 既存の KOPT アルゴリズムとの比較：ステップ数に対する評価値の推移

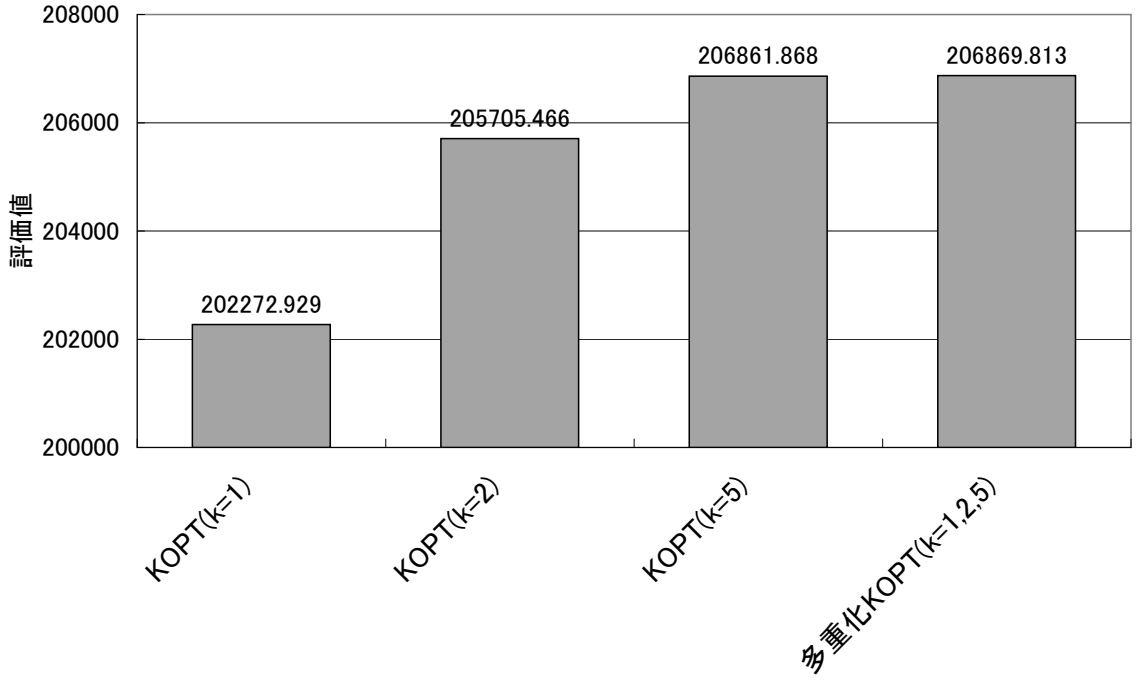

図 5.2: 既存の KOPT アルゴリズムとの比較：停止時点での評価値

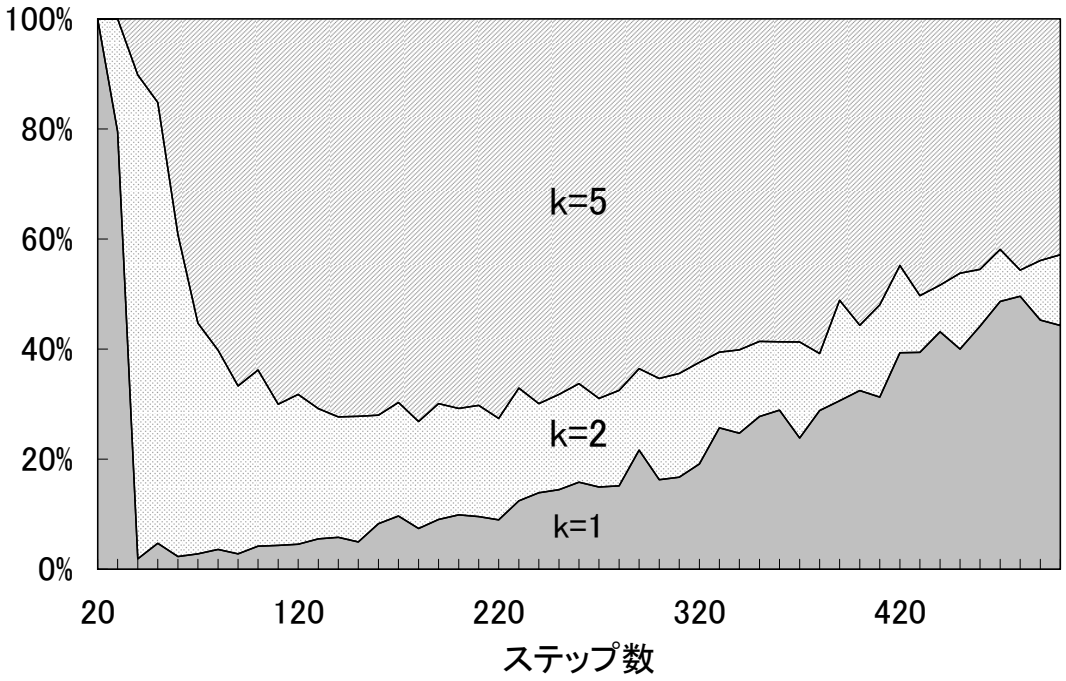

図 5.3: 同期の際に選択されるパラメータの割合

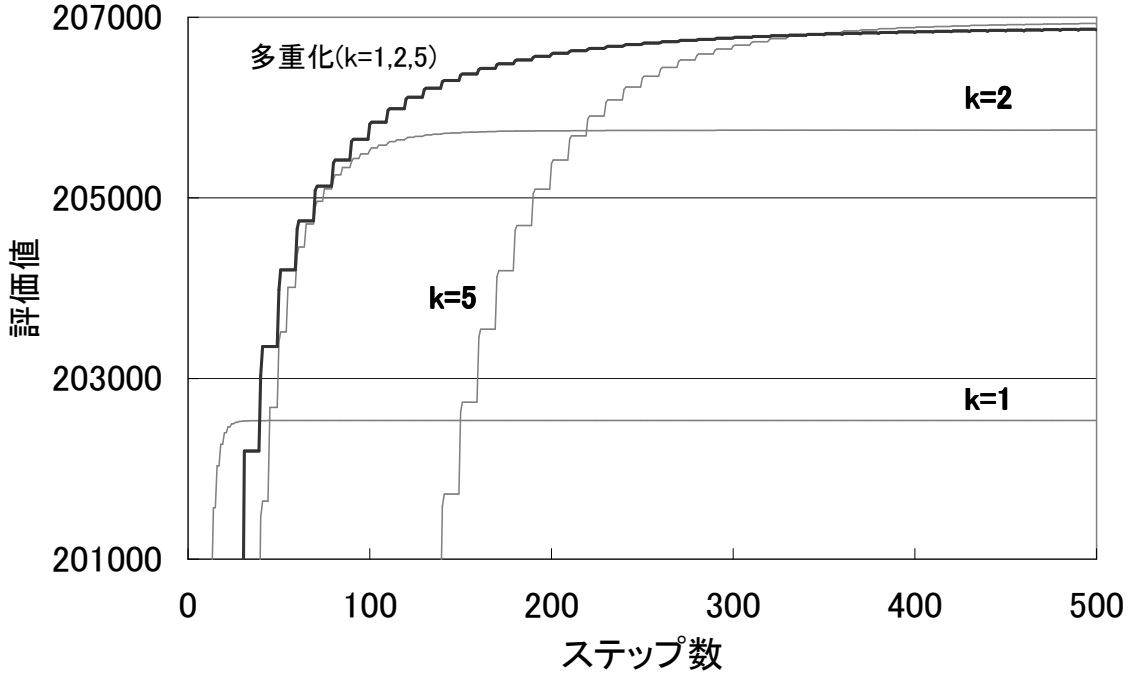

図 5.4: 同一パラメータで多重化した場合との比較：ステップ数に対する評価値の推移

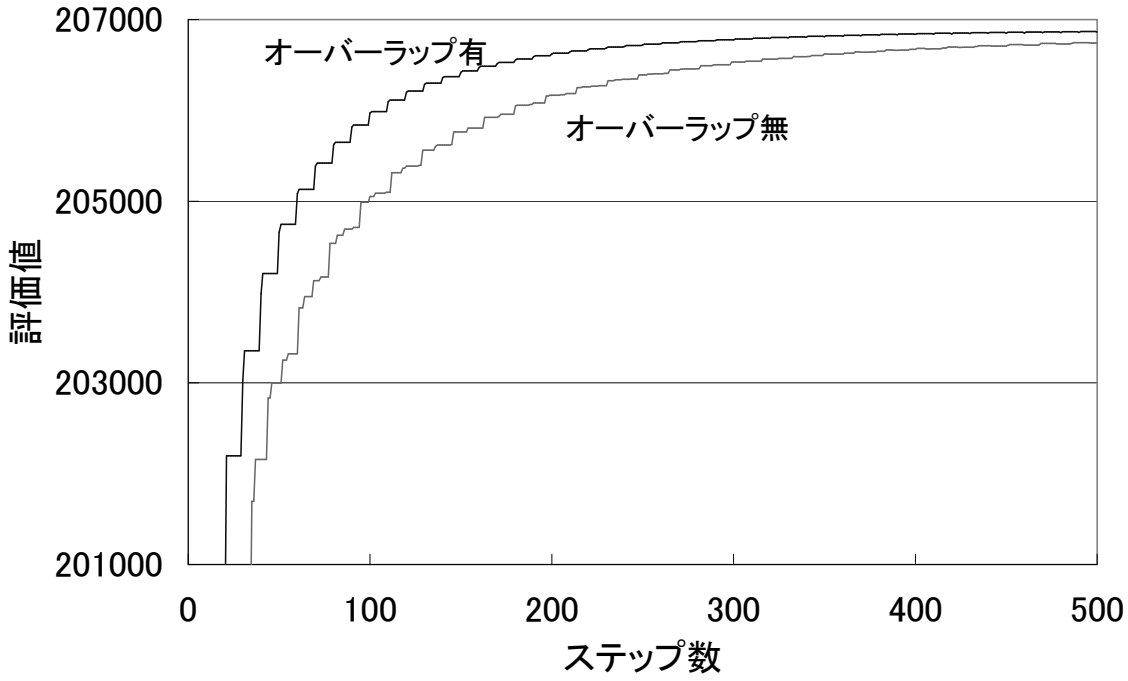

図 5.5: オーバーラップを行わない場合との比較：ステップ数に対する評価値の推移

第6章

従来手法: DUCTアルゴリズム

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解法であるDUCTアルゴリズムについて述べる．

6.1 DUCTアルゴリズムの動作

DUCTアルゴリズムでは，まず，前処理として制約網から擬似木を構成する．擬似木は，例えば深さ優先探索により構成できる．制約網からエージェントを１つ選び深さ優先探索を行う．この探索で通った辺を擬似木の木辺とし，通らなかった辺を擬似木の後退辺とする．後退辺でつながったエージェントのうち，根に近い側を擬似親，葉に近い側を擬似子と呼ぶ．擬似木を構成する一連の操作は，分散アルゴリズムで実行される．制約ネットワークから擬似木を構成する例を，図6.1に示す．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 に注目すると，親は x_2，子は x_4, 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．擬似木を構成する方法は，深さ優先探索に限らないが，部分木をまたぐ後退辺が存在してはならない．深さ優先探索で擬似木を構成した場合は，後退辺が部分木をまたぐことはない．このように，前処理として擬似木を構成することは，分散制約最適化問題の解法では広く行われている．[10] 擬似木を用いる分散制約最適化問題の解法は，DUCTの他にDPOP[3]などがある．

DUCTでは，擬似木を構成した後，初めに根に位置するエージェントが自身の管理する変数値 d を決める．この時，根に位置するエージェント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ので，変数値 d は値域 D_k からランダムに選択され

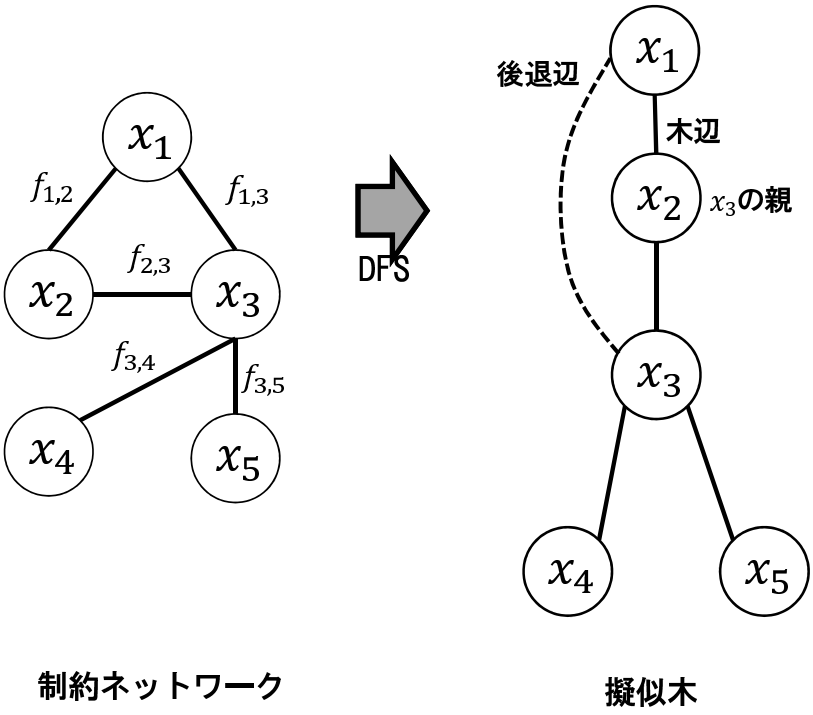

図 6.1: 制約ネットワークからの擬似木の構成

る．D_k は，エージェント k の変数の値域を表す．エージェントは，自身の変数に対して変数値 d を選んだことを，擬似木上の子エージェントに対して送信する．このメッセージをコンテキストメッセージと呼ぶ．

続いて，コンテキストメッセージを受信したエージェントも，自身の変数値をランダムに決定する．そして，受信したコンテキストメッセージにこの変数値の情報を加えて，子エージェントに送信する．コンテキストメッセージが擬似木上の葉に位置するエージェントに届くまで，この操作を繰り返す．コンテキストメッセージを受信したエージェントが擬似木上の葉であった場合は，自身の変数について，親および擬似親の変数値に対してコスト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全てのエージェントが持つ変数に対して，１通りの割り当てが決まった

ことになる．

変数に対する割り当てが決まると，続いて，この割り当て a のコストを計算する．コンテキストメッセージに対して変数値を決めた葉エージェントは，自身の変数値と親および擬似親の変数値の間のコスト (式 6.1) を計算し，コストメッセージとして親へ送信する．

$$y_k^t = \min_{d \in D_k} l(a, d) \tag{6.1}$$

t は，t 回目のサンプリングを意味する．全ての子からコストメッセージを受信したエージェントは，同様に，自身の変数値と親および擬似親の変数値の間のコスト（式 6.2）を計算する．

$$y_k^t = l(a, d) + \sum_{k' \in C(k)} y_{k'}^t \tag{6.2}$$

ここで，$C(k)$ は，エージェント k の子エージェントの集合を表す．$\sum_{k' \in C(k)} y_{k'}^t$ は，子エージェント $C(k)$ から受信したコスト y_k^t の総和を表す．この時，葉ノード以外のエージェントは，サンプリングのためのバウンドと呼ばれる値を同時に計算する．バウンドは，受信したコンテキスト a，エージェント k 自身の変数値 d に対して次のように定義される．（式 6.3）

$$B_{a,d}^t \triangleq l(a, d) + \max \left\{ \hat{\mu}_{a,d}^t - L_{a,d}^t, \sum_{k' \in C(k)} B_{k'}^t \right\} \tag{6.3}$$

ここで，$B_{k'}^t$ は，エージェント k の子エージェント k' から送られてきたバウンド (式 6.4) である．

$$B_k^t = \min_{d' \in D_{k'}} B_{a',d'}^t \tag{6.4}$$

a' は，コンテキスト a を受信し変数値 d をとったエージェント k の子エージェント k' が受信したコンテキストを意味し，式 6.5 のように定義される．

$$a' = a \cup \{x_k = d\} \tag{6.5}$$

$\hat{\mu}_{a,d}^t$ は，コンテキスト a を受信したエージェントが変数値 d を選択した時に発見された最小のコストを意味し，式 6.6 のように定義される．

$$\hat{\mu}_{a,d}^t \triangleq \min \left\{ y_k^l | l \leq t : a_k^l = a, x_k^l = d \right\} \tag{6.6}$$

$L^t_{a,d}$ は，UCB スタイル [11] のバウンドを意味する．$L^t_{a,d}$ は，式 6.7 のように定義される．

$$L^t_{a,d} = \sqrt{\frac{2\lambda_a \ln \tau^t_a}{\tau^t_{a,d}}} \tag{6.7}$$

ここで，$\hat{\mu}^t_a$ は，コンテキスト a を受信した時，エージェントがこれまでに発見した最小のコストを意味する．

$$\hat{\mu}^t_a \triangleq \min_d \hat{\mu}^t_{a,d} \tag{6.8}$$

λ_a は，最も深い葉エージェントへのパスの長さを意味する．

葉エージェント以外のエージェントはコストとバウンドをコストメッセージとして親へ送信する．この操作を繰り返して，根エージェントが全ての子からコストメッセージを受信すると，１通りの割り当てに対してそのコストが計算できたことになる．

DUCT アルゴリズムでは変数値の決定とコストの計算を繰り返すことで，解空間を探索する．２回目以降の割り当てでは，変数の選択時に，ランダムではなく，これまでの探索結果を用いる．エージェント k の管理する変数 x_k に割り当てる値は，次の式に従う．(式 6.9)

$$x^t_k \triangleq \arg_{d \in S^t_a} \min B^t_{a,d} \tag{6.9}$$

S^t_k は，サンプル可能な変数値の集合を意味する．(式 6.10)

$$S^t_k = \left\{ d \in D_k | B^t_{a,d} \neq l(a,d) + \hat{\mu}^t_{a,d} \right\} \tag{6.10}$$

各エージェントが停止する条件を説明する．各エージェントは，自身の親エージェントがすでに停止していて，かつ，次の式 6.11 を満たしたとき停止する．

$$\max_{d \in S^t_k} \hat{\mu}^t_a - (\hat{\mu}^t_{a,d} - \sqrt{\frac{\ln \frac{2}{\delta}}{\tau^t_{a,d}}}) \leq \epsilon \tag{6.11}$$

ϵ と δ は，停止条件に関してアルゴリズムに与えられるパラメータである．擬似木の根に位置するエージェントから，この条件を満たした順に，葉へ向かって停止する．

6.2 DUCT の保存するコンテキスト

DUCT アルゴリズムの動作を各エージェントが保存する情報の点から説明する．前節で説明したアルゴリズムの動作のうち，各エージェントがサンプリングのために保

コンテキスト	x_1	0	0	1
	x_2	0	0	2
	$d = x_3$	2	1	0
回数	τ_a^t	2	2	1
	$\tau_{a,d}^t$	1	1	1

図 6.2: エージェント 3 のメモリ

存する情報は，受信したコンテキスト a に対する受信回数と，あるコンテキスト a を受信したとき自身の変数値として d を選択した回数である．図 6.1 の制約ネットワーク上の変数 x_3 を管理するエージェント 3 の持つメモリの内容の例を，図 6.2 に示す．

図 6.2 のテーブルの縦方向は，コンテキストメッセージ中の変数の数であるので，先祖エージェントの数に比例する．テーブルの横方向は，コンテキストの規模に対して指数的に増加する．横方向が指数的に増加するのは，最大サイズが変数の値域と先祖エージェント数の積で決まるためである．このため，制約最適化問題の規模が大きくなると使用するメモリのサイズが大きくなりすぎて，解くことができなくなる．また，この影響は，先祖エージェントの数に関わるため，擬似木上の下部に位置するエージェントほど大きくなる．

第7章

DUCTを改良する提案手法1: コンテキスト中の必要な変数値の保存

7.1 基本的なアイデア

6.2 で指摘したエージェントがメモリを使いすぎる問題を改善するために，コンテキスト中の必要な変数値のみを保存する手法を提案する．既存の DUCT アルゴリズムにおいてコンテキストは，サンプリングの結果を区別すること，および，割り当てた変数値に対するコストの計算のために用いられる．しかし，既存の DUCT アルゴリズムでは，コストの計算に必要のない変数値もコンテキストメッセージに含めて下位のエージェントに対して送信している．そして，このコンテキストメッセージを受信したエージェントは，この不要な変数値を含むコンテキストについて保存している。図 7.1,7.1 を用いて説明する．既存の DUCT アルゴリズムでは，エージェント 1 から 3 の管理する変数がそれぞれ $x_1 = 0, x_2 = 2, x_3 = 0$ をとっている場合，図 7.1 の矢印で示されるようなコンテキストメッセージが送られる．しかし，x_4,x_5 については，コストの計算に必要なのは x_3 の値だけである．そのため，エージェント 4,5 は x_2, x_3 の値をコンテキストとして保存する必要はない．よって，エージェント 4 が必要のない変数値の保存をやめると，図 7.1 のようにメモリを削減できる．

そこで，本提案では，コストの計算に不要な変数値をコンテキストメッセージとして送信することを止め，コストの計算に必要な変数値のみを保存することで，使用す

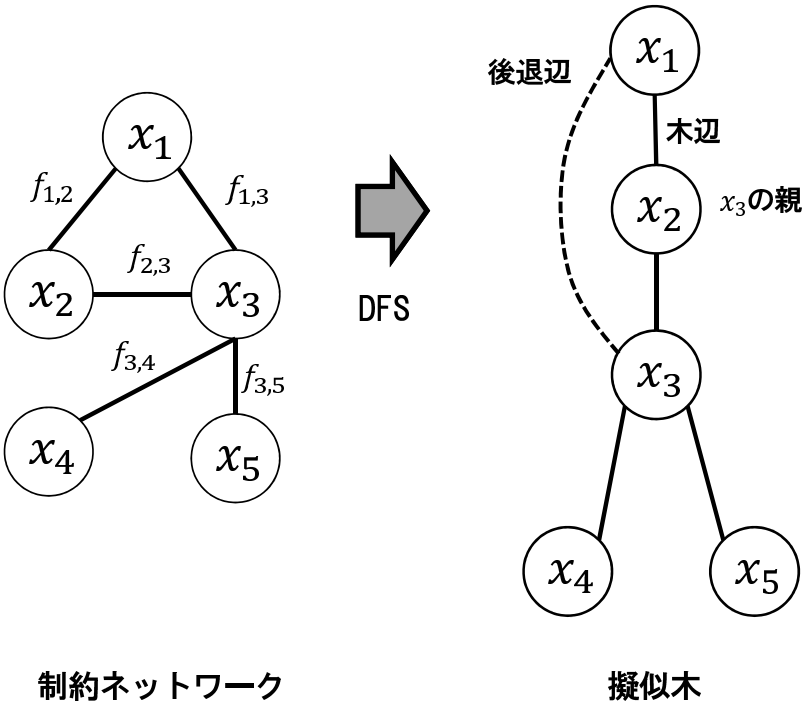

図 7.1: コンテキストメッセージの例

るメモリ量を削減する．

7.1.1 コストの計算に必要な変数値

エージェントがコストの計算に必要な変数値について説明する．各エージェントは，擬似木上で自身から下の部分木のコストを計算する．ここには，親と自身の間のコスト，擬似親と自身の間のコストも含まれる．このうち，子を根とする部分木のコストは，コストメッセージに含まれて，子から伝えられる．そのためエージェントが直接計算するコストは，親および擬似親との間のコストである．そして，子を根とする部分木が必要とする変数値も再帰的に必要とされる．以上をまとめると，コストの計算

削減

コンテキスト	$\boldsymbol{x_1}$	0	0	0
	$\boldsymbol{x_2}$	0	0	2
	$\boldsymbol{x_3}$	2	1	0
	$d = \boldsymbol{x_4}$	0	1	0
回数	τ_a^t	2	2	1
	$\tau_{a,d}^t$	1	1	1

図 7.2: 削減されるエージェント 4 のメモリ

に必要な変数値は次のとおりである．

- 子が必要とする変数値
  - 子がコストの計算に必要な変数値
  - 部分木がコストの計算に必要な変数値
- コストの計算に必要な変数値
  - 親の変数値
  - 擬似親の変数値

図 7.3 を例に説明する．エージェント 3 がコストの計算に直接必要な変数値は，親の持つ変数 x_2 の値と，後退辺 $f_{1,3}$ で繋がれた擬似親が持つ変数 x_1 の値である．x_4, x_5 以下の変数値と x_3 より根側の変数値は，後退辺でつながっていないので，これ以上の変数値は，コストの計算に必要でない．既存の DUCT アルゴリズムでは，x_4, x_5 を管理

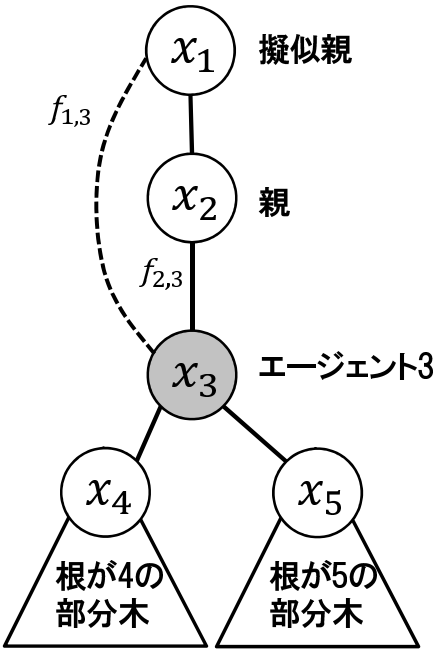

図 7.3: コストの計算にエージェント 3 が必要とする変数の値について

するエージェントは，それぞれ，x_3 の値だけでなく x_1, x_2 の値もコンテキストとして保存する．

7.1.2 効果が高い場合と低い場合

本手法の効果は，制約網から生成された擬似木の構造に依存する．長い後退辺が多く存在する場合，本手法の効果は少なくなる．これは，後退辺が長い場合は，後退辺がまたぐ経路中にあるすべてのエージェントが，後退辺の擬似子エージェントが必要とする変数値として後退辺の擬似親エージェントの変数値を保存する必要があるためである．

7.1.3 解に与える影響

本手法により，コストの計算に必要な変数値のみを保存すると，コンテキストとして既存の DUCT アルゴリズムと異なる変数値を保存することになる．例えば，エージェント k は，既存の DUCT では，コンテキストとして変数 x_1, x_2, x_3 の値を保存していたとする．ここで，コストの計算に必要な変数が x_3 だけだった場合，これまで区別されていたコンテキスト $[x_1 = 0, x_2 = 0, x_3 = 0]$ と $[x_1 = 0, x_2 = 1, x_3 = 0]$ は，どちらも $[x_3 = 0]$ として区別されなくなる．

このようにコンテキストの区別をなくすことで，得られる解の精度が悪化する懸念がある．この点については，影響がないことを実験的に確認した．この点については，第 9 章で述べる．

第8章

DUCTを改良する提案手法2: 保存するコンテキスト量の制限

6.2で指摘したエージェントがメモリを使いすぎて解を出せなくなる可能性がある問題に対して，各エージェントが保存するコンテキストの量を一定量以下にする手法を提案する．実際のアプリケーションを想定した場合，エージェントが動作する環境には，メモリ量の制限があることが考えられる．コンテキスト量が規定値を上回った場合は，コンテキストを1つ追い出すことで，保存するコンテキスト量を一定量以下にし，メモリを使いすぎて解を出せなくなる状況を回避する．

8.1 追い出すコンテキストの選択基準

メモリから追い出すコンテキストを決める基準について，第1に，バウンドを用いることを提案する．エージェントが変数値を選択する際には，式6.9に従い，バウンド $B^t_{a,d}$ を最小化する変数値 d を選択する．そのため，バウンド $B^t_{a,d}$ が大きいコンテキスト a については，今後参照される可能性が低いと考えられる．そこで，新たなコンテキストをコンテキストメッセージとして親エージェントから受信したときに保存できるコンテキスト数を上回った場合は，最大のバウンド $B^t_{a,d}$ を持つコンテキスト a をメモリから追い出す．DUCTアルゴリズムでは，過去に受信したコンテキスト a についてのバウンドを保存していないので，本手法では，コンテキスト a に対するバウンドの最大値 B_a を保存する．

第 2 に，コンテキスト a の受信回数 τ_a^t をメモリから追い出すコンテキストを決める基準に用いることを提案する．エージェント k の先祖エージェントは，式 6.9 に従いバウンドに基づいて変数値を選択する．そのため，エージェント k があるコンテキスト a 受信する回数 τ_a^t は，先祖エージェントが変数値を選択する段階でバウンドの影響を受けている．よって，選択基準にコンテキストの受信回数 τ_a^t を用いることは，間接的にバウンドを用いていると言える．先に提案したバウンドを用いる手法では，新たにバウンドの値を保存する必要があった．これに対して，コンテキスト a の受信回数 τ_a^t は，既存の DUCT アルゴリズムでも保存している値である．そのため，新たに保存する情報を増すことはないメリットがある．

8.2 解の悪化

保存するコンテキスト量を制限することで解コストが悪化する場合がある．追い出すコンテキストが，今後一切参照されない場合は，解の悪化が発生しないことは自明である．しかし，今後参照されないコンテキストを事前に正確に予測することは不可能である．追い出したコンテキストが，後に参照された場合，バウンドが正しく計算されずサンプリングに悪い影響を与えるため，解の精度が悪化する．保存するコンテキスト量を増やすほど，必要なコンテキストを追い出してしまう可能性が減り解の悪化が抑えられるがが，メモリ使用量は増加する．これは，トレードオフである．

第9章

評価:
DUCTのコンテキスト中の
必要な変数値の保存による効果

本章では，既存手法 DUCT アルゴリズムと DUCT アルゴリズムに対する 1 つ目の提案手法であるコンテキスト中の必要な変数値のみを保存する手法を実験により比較し，考察する．既存手法との比較は，得られた解のコストを用いた．

9.1 問題の生成方法

今回の実験では制約網の構造に，木構造に落とし込みやすい構造として BA モデルを選択した．1つのエージェントが管理する変数は1つとした．エージェント i の管理する変数 x_i のとりうる値の数は 5 値とした.（式 9.2, 9.2）

$$D_i = \{0, 1, 2, 3, 4\} \tag{9.1}$$

$$x_i \in D_i \tag{9.2}$$

辺 $e_{i,j}$ に対応するコスト関数 $f_{i,j}$ は，頂点 v_i, v_j に対応する変数 x_i, x_j の組み合わせに対して，0 から 24 の整数値から同じ値を選ばないようにランダムに評価値を割り当てた．例として，9.1 に示すようなコスト関数が生成される．このコスト関数 $f_{i,j}$ は，$x_i = 4$ かつ $x_j = 3$ のとき，最小のコスト値 0 を返す.

表 9.1: コスト関数の例

$x_i \setminus x_j$	0	1	2	3	4
0	16	15	11	4	7
1	3	19	6	20	2
2	24	5	8	18	12
3	21	9	22	23	14
4	1	10	17	0	13

9.2 グラフの構造による削減効果の違い

予備実験として，グラフ構造によるコンテキスト量の削減効果の違いを確認した．9.1 の方法で作成した BA モデルのグラフと，ランダムに制約辺を繋いだランダムなグラフを比較した．エージェント数 $n = 50$，制約辺の密度は 0.3 とした．擬似木の生成は，深さ優先探索を用いた．

既存手法では，グラフの種類によらず保存する変数の数は，エージェントの深さに比例して単調に増加する．これは，先祖エージェントの変数を全て保存するためである．一方，提案手法では，コストの計算に必要な変数のみを保存するため，既存手法に比べて，保存する変数の数が少ない．BA モデルグラフでは，深さ 28 をピークに保存する変数の数が減少する．ランダムなグラフでは，保存する変数の数の現象は BA モデルグラフに比べて深く，深さ 37 をピークに減少する．保存する変数の数の減少が，深い位置にあるエージェントほど大きくなる理由は，根に近くなるにつれて子孫エージェントと後退辺で繋がる可能性が減るためである．

BA モデルグラフに比べて，ランダムなグラフにおいて保存する変数の数が多くなる理由は，後退辺が長くなりやすく，コストの計算に不要な変数が少ないためである．

9.3 DUCT アルゴリズムとの比較

対象とする問題は，9.1 の方法で作成した BA モデルグラフを用いる．エージェント数 n は，$n = 10, 15, 20, 25$ の 4 種類を用意した．各エージェント数に対して，10 個の

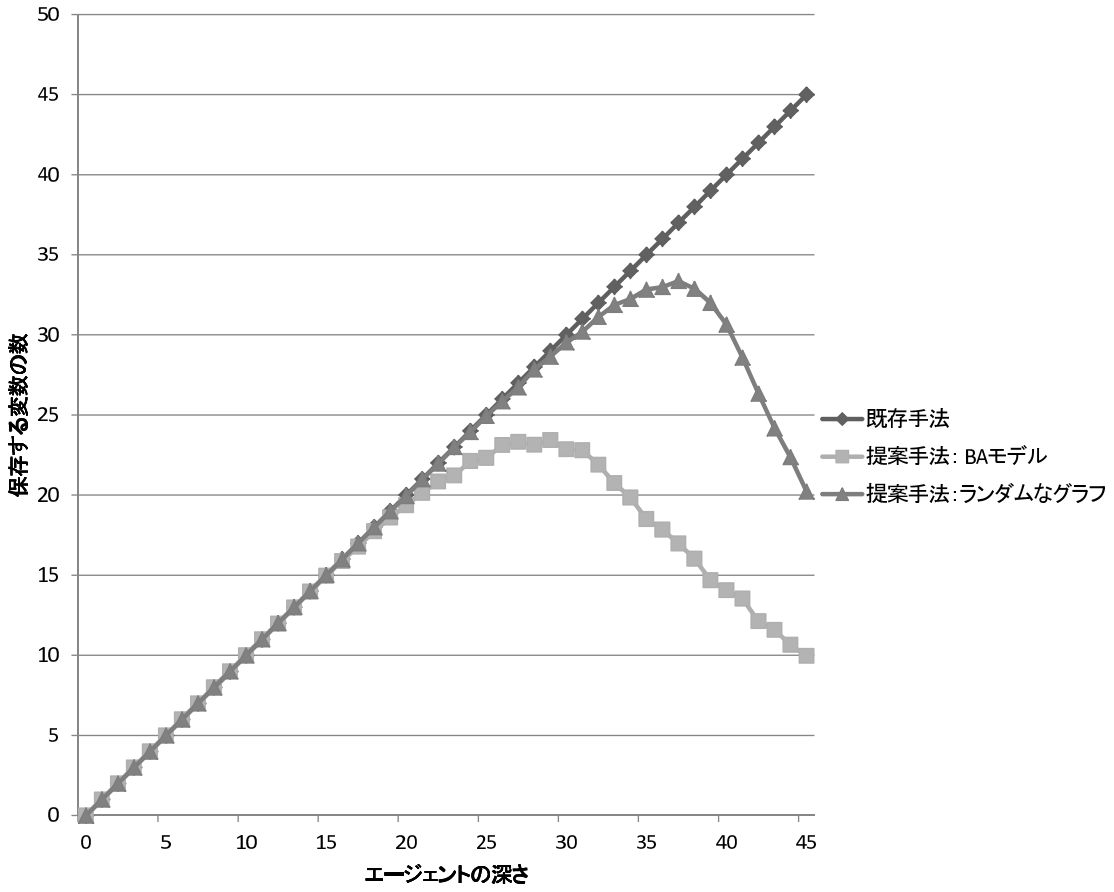

図 9.1: グラフ構造による削減効果の違い

グラフを生成した．各問題について，乱数シードを変えて 4 回ずつ実行した．得られた解のコストの平均を 9.2 に示す．

いずれのエージェント数の場合においても，既存の DUCT アルゴリズムと提案手法は同等の解コストを得られていると言える．コストの計算に不要な変数は，保存を行わななくとも，サンプリングの過程は既存の DUCT と異なるが，解のコストには影響しない．

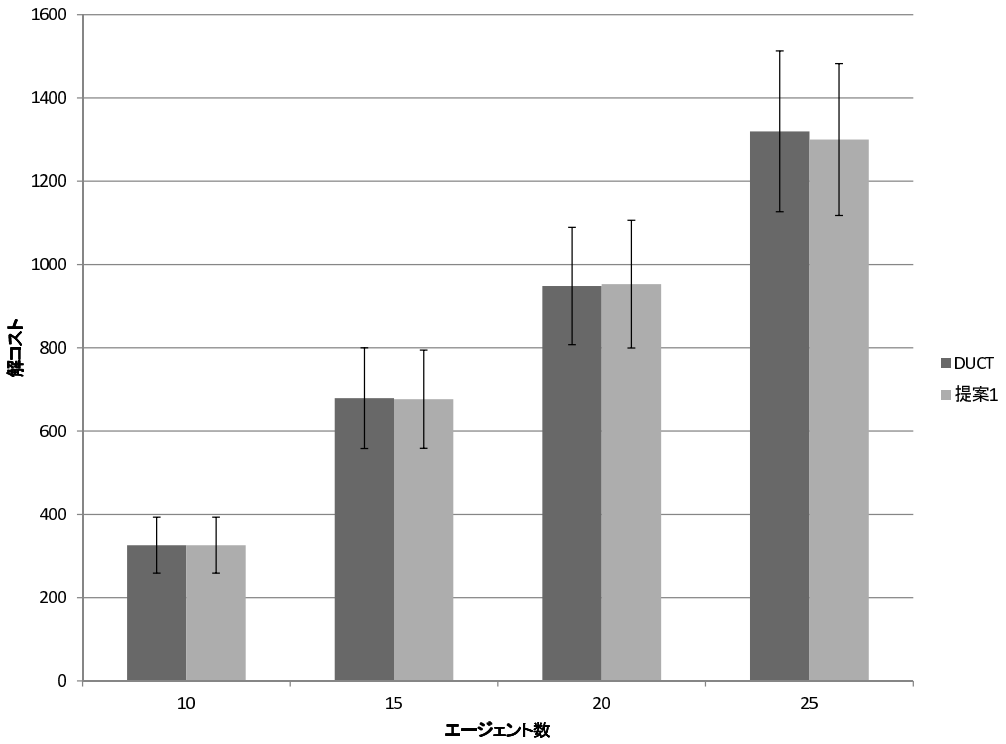

図 9.2: 既存の DUCT アルゴリズムとの比較: 得られるコスト

第10章

評価: DUCTの保存するコンテキスト量の制限

本章では，DUCT アルゴリズムと 2 つ目の提案手法である保存するコンテキスト量を制限する手法を実験により比較し，考察する．評価に用いるグラフは，前章で作成した BA グラフモデルのグラフを用いた．エージェント数 $n = 10, 15, 20, 25$ のグラフについて，保存するコンテキスト数を 10 から 2000 まで変化させて，得られる解のコストを比較した．メモリから追い出すコンテキストの選択は，先に説明したバウンド $B^t_{a,d}$ が高いコンテキスト，受信回数 τ^t_k が多いコンテキストに加えてランダムに選択されたコンテキストの 3 種類を用いた．

エージェント数毎の結果を，図 10.1 から図 10.4 に示す．横軸を保存するコンテキスト数，縦軸を解コストとした．問題の規模に対して，保存するコンテキスト数が小さいと解コストが悪くなる．保存するコンテキストを増やしていくと，解コストは良くなり，十分に多いコンテキストの保存を許すと，それ以上保存コンテキスト数を増やしても解コストの改善がみられなくなった．保存するコンテキスト量と解コストの関係は指数的であり，保存コンテキスト量を問題規模に対して小さくしすぎると解コストが急激に悪化した．追い出す基準については，バウンド $B^t_{a,d}$ と受信回数 τ^t_a は同等の解コストを得られてた．ランダムに追い出すコンテキストを選択する方法では，エージェント数 $n = 10, 15$ のグラフ場合においてバウンド $B^t_{a,d}$ と受信回数 τ^t_a と同等か悪い結果を示す場合があった．しかし，エージェント数 $n = 20, 25$ のグラフにおいて，特に保存コンテキスト数が少ない場合，他の基準よりもよい解が得られた．バウンド $B^t_{a,d}$

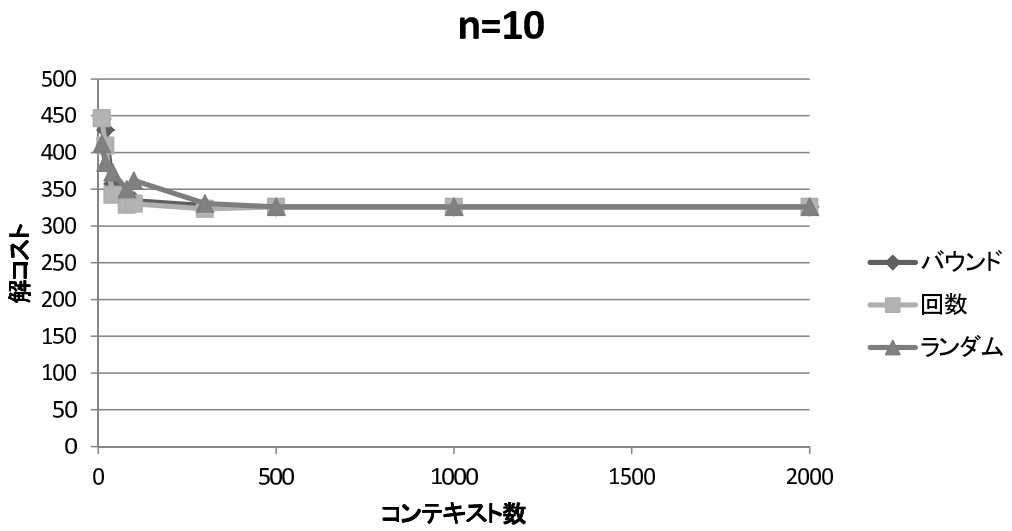

図 10.1: 追い出し基準による違い（n=10）

や受信回数 τ_a^t を基準にするよりも，ランダムに選択した追い出す対象を基準にしたほうが，よい結果が得られる場合がある理由については，保存コンテキスト数が少ない場合は，バウンドや受信回数が十分に信頼できる程度に探索が進む前に，保存上限数を超えてしまい，偏りのないサンプリングが行えなくなるためと考えられる．この問題を解決するためには，保存コンテキスト数が特に少ない場合には，追い出すコンテキスト数の選択にタブーリストやランダムな手法を併用することで，サンプリングの偏りを減らす方法が考えられる．

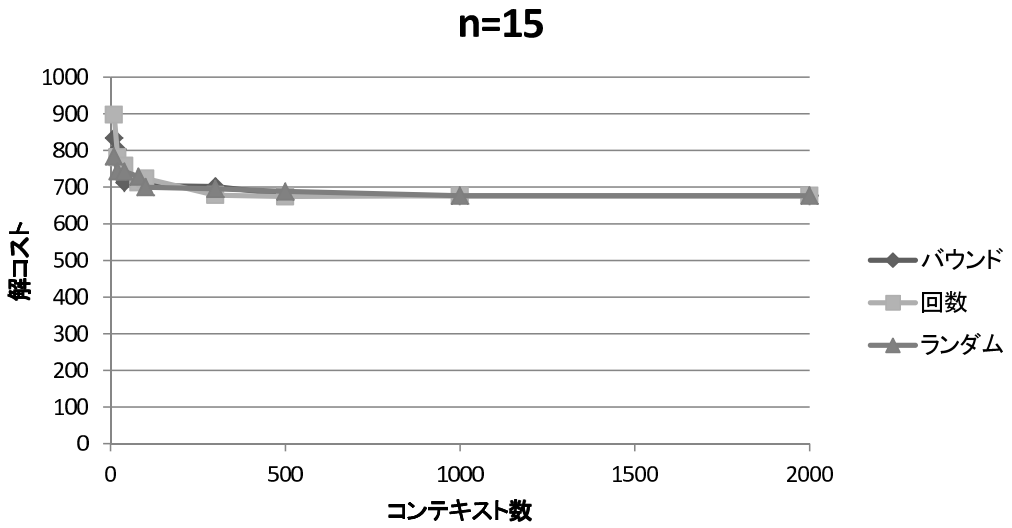

図 10.2: 追い出し基準による違い（n=15）

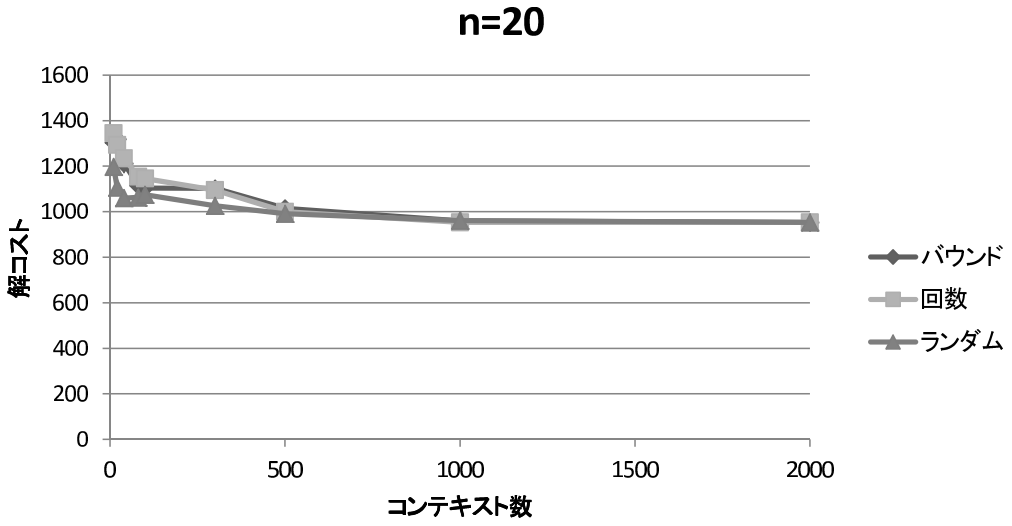

図 10.3: 追い出し基準による違い（n=20）

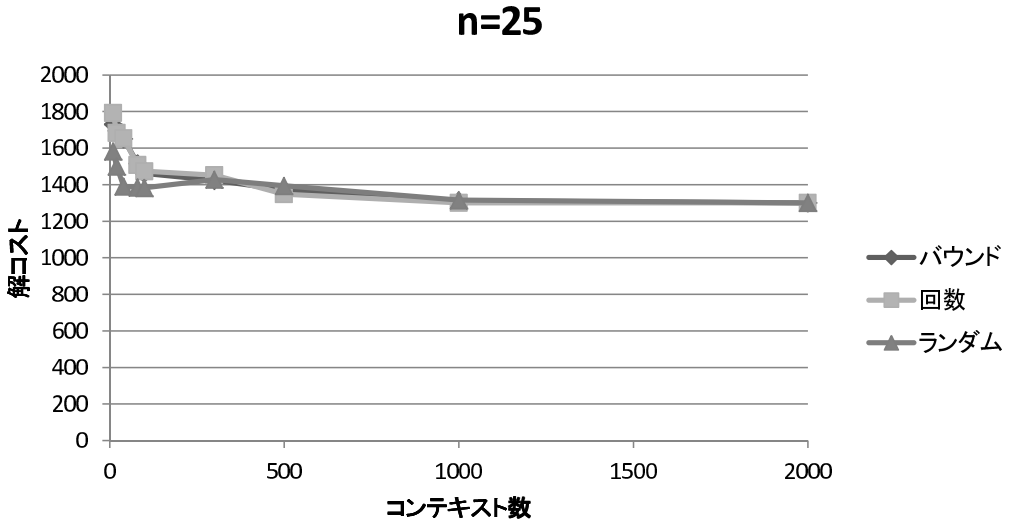

図 10.4: 追い出し基準による違い（n=25）

第11章

おわりに

本論文では，分散制約最適化問題の非厳密解法の改良手法を提案し，評価した．第1に，KOPT アルゴリズムのもつ解品質と探索時間抑制を両立するため，探索パラメータ k を変えた探索を多重化して実行する方法を提案した．計算機シミュレーションにより，その有効性を確認した．

第2に，DUCT アルゴリズムのメモリ使用量を削減する手法を提案し，計算機シミュレーションにより評価を行った．コスト計算に冗長な変数値の保存を止めることで，メモリ使用量の削減が可能である．コンテキスト保存量を一定に制限した場合の，コンテキスト保存量と解コストの関係を計算機シミュレーションにより確認した．メモリから追い出すコンテキストの選択基準については，既存の指標のみによらない改善の余地があることが示された．

謝辞

本研究のために，多大な御尽力を頂き，御指導を賜った名古屋工業大学の松尾啓志教授，津邑公暁准教授，齋藤彰一准教授，松井俊浩准教授に深く感謝致します．また，本研究の際に多くの助言，協力をして頂いた松尾・津邑研究室，齋藤研究室および松井研究室の方々に深く感謝致します．

本研究に関する発表

1. 柳田 大輝, 松井 俊浩, 松尾 啓志: ”分散制約最適化問題の k-Optimality に基づく解法の多重化”, 合同エージェントワークショップ&シンポジウム (JAWS2011) ロング発表論文 (Oct,2011)

参考文献

[1] Modi, P., Shen, W., Tambe, M. and Yokoo, M.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees, *Artificial Intelligence*, Vol. 161, No. 1-2, pp. 149–180 (2005).

[2] Farinelli, A., Rogers, A., Petcu, A. and Jennings, N.: 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-Volume 2*, pp. 639–646International Foundation for Autonomous Agents and Multiagent Systems (2008).

[3] Petcu, A. and Faltings, B.: A scalable method for multiagent constraint optimization, in *International Joint Conference on Artificial Intelligence*, Vol. 19, p. 266Citeseer (2005).

[4] Zhang, W., Wang, G. and Wittenburg, L.: Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance, in *Proceedings of AAAI Workshop on Probabilistic Approaches in Search* (2002).

[5] Katagishi, H. and P.Pearce, J.: KOPT : Distributed DCOP Algorithm for Arbitrary k- optima with Monotonically Increasing Utility, in *Proceedings of the Ninth Distributed Constraint Reasoning workshop(DCR)* (2007).

[6] Ottens, B., Dimitrakakis, C. and Faltings, B.: DUCT: An Upper Confidence Bound Approach to Distributed Constraint Optimization Problems, in *Proceedings of the 26th conference of the AAAI*AAAI (2012).

[7] Pearce, J. P. and Tambe, M.: Quality Guarantees on k-Optimal Solutions for Distributed Constraint Optimization Problems, in *International Joint Conference on Artificial Intelligence (IJCAI), 2007* (2007).

[8] 飯塚泰樹, 鈴木浩之, 竹内郁雄:分散制約充足問題のための Multi-agent Tabu Search 手法の効果 (モデル/理論, <特集>ソフトウェアエージェントとその応用論文), 電子情報通信学会論文誌. D，情報・システム, Vol. 90, No. 9, pp. 2302–2313 (2007).

[9] P.Gomes, C. and Selman, B.: Algorithm Portfolios, *Artificial Intelligence Journal*, Vol. 126, pp. 43–62 (2001).

[10] 天太沖本, ヨンジュンジョ, 敦岩崎, 真横尾：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).

[11] Pete, A., NICOL'O, C.-B. and PAUL, F.: Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning*, Vol. 47, pp. 235–256 (2002).

[12] 清一西原：制約充足問題の基礎と展望 (<特集> 制約充足問題の基礎と応用), 人工知能学会誌, Vol. 12, No. 3, pp. 351–358 (1997).

[13] 飯塚泰樹, 竹内郁雄：分散制約最適化問題近似解法の多重実行の効果, 情報処理学会論文誌, Vol. 50, No. 12, pp. 3136–3149 (2009-12-15).