JAWS2011

分散制約最適化問題の k-Optimality に基づく解法の多重化

Multiplexing k-Optimal search on Distributed Constraint Optimization

柳田 大輝 Daiki YANAGITA
名古屋工業大学 Nagoya Institute of Technology
yanagita@matlab.nitech.ac.jp

松井 俊浩 Toshihiro MATSUI
（同　上）
matsui.t@nitech.ac.jp

松尾 啓志 Hiroshi MATSUO
（同　上）
matsuo@nitech.ac.jp

keywords: DCOP, multi-agents, k-Optimality, optimization, search

Summary

Distributed constraint optimization problems (DCOP) have been studied as a basic framework of multi-agent cooperation. Algorithms for solving distributed constraint optimization problems can be classified into complete algorithms and incomplete algorithms. As the scale of DCOPs becomes larger, it is difficult to solve these problems using complete algorithms. Therefore incomplete algorithms are selected to solve large scale DCOPs. In general, incomplete algorithms can generate sub-optimal solutions with smaller amount of computation or messages than complete algorithms. Recently, the k-Optimality has been proposed as new measure for DCOP solutions. A solution that can not be improved by changing k or less number of variables is defined as a k-Optimal solutions. As a DCOP solver that can produce arbitrary k-Optimal solutions, KOPT algorithm has been proposed. However, KOPT algorithm needs a large number of iterations to compute higher quality solutions. To reduce the search time, we introduce a multiplexed search of KOPT algorithm without debasing the quality of solutions. In this algorithm, we choose two or more different values of k to multiplex the searches. We evaluated the algorithm using simulation. The results show that our algorithm obtains the same quality of solutions as the single KOPT while it reduces the number of iterations.

1. 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．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．[Modi 05, Farinelli 08]

分散制約最適化問題を解くアルゴリズムは，ADOPT[Modi 05] や DPOP[Petcu 05] などの厳密解法と，DSA[Zhang 02] や Max Sum[Farinelli 08] などの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．

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いなかったが，近年，k-Optimality という指標が提案された [Pearce 07]．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 [Katagishi 07] 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要する．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．

本論文は以下のように構成される．2 節では分散制約最適化問題について述べる．3 節では従来手法である KOPT アルゴリズムについて述べ，4 節では探索を多重化する方法を提案する．5 節では従来手法と提案手法を比較した評価について述べ，6 節でまとめる．

2. 分散制約最適化問題

分散制約最適化問題は，与えられた制約をできる限り満たすような解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対して 1 つの変数を配置する．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．i 番目の変数 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 (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．

$$R(a)=\sum_{f_{i,j}\text{ for all constraints}} f_{i,j}(a_i,a_j) \qquad (1)$$

このために，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．

3. 既存手法 – 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 [Pearce 07]．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 [Katagishi 07] の特徴と動作について述べる．

3·1 k-Optimality

k-Optimality 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 (2) のように定義される．

$$\forall\tilde{a}\ \text{ s.t. } d(a,\tilde{a})\le k, R(a)-R(\tilde{a})\ge 0 \qquad (2)$$

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こととする．

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 の実行例を図 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_3, 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．

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 $\lfloor(k+2)/2\rfloor$

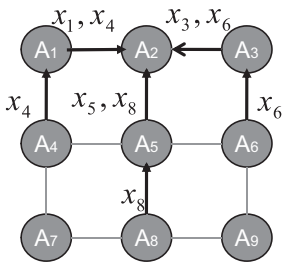

図 1 エージェント A_2 が受信する変数値（$k = 2$ のとき）

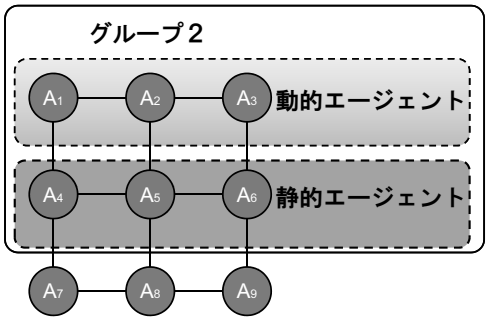

図 2 A_2 を中心とするグループ 2 の一例 (k=3 のとき)

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．

図 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

表 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	$A_1, A_2, A_3, A_4, A_5, A_6$

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，表 1 のようになる．

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) 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 (表 2) の合計である．

$$2 \times \lfloor \frac{k}{2} \rfloor + k + 1 \qquad (3)$$

表 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	$k-1$

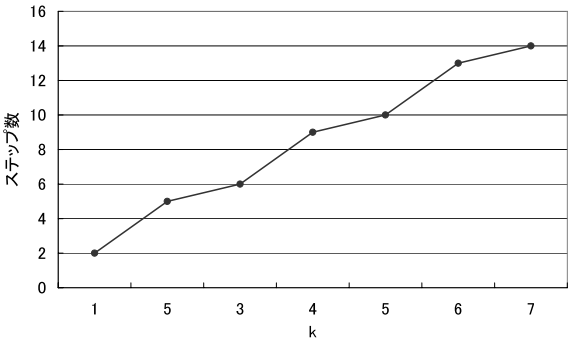

図 3 k に対する 1 回の反復処理に必要な通信ステップ数

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) 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図 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.

4. 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並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改善し，解品質の向上と探索時間の抑制を両立させる.

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 [P.Gomes 01, 飯塚 15].

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.

一般に，分散制約最適化問題の解法においては，各エージェントの処理における計算時間が，メッセージ通信時間に比べて，十分に短いと仮定する．このため，アルゴリズムの性能の指標として，しばしばメッセージ交換による通信の回数が用いられる [Modi 05, 飯塚 07]．KOPT は計算量とメッセージのサイズが比較的小さいアルゴリズムであるため，その処理をある程度まで多重化しても，同時に送信されるメッセージを単一のメッセージに統合すれば，多重化の影響は 1 回の通信に要するオーバヘッドよりも十分に小さい場合があると考えられる．このように，1 回の反復処理に必要な計算時間が，通信時間に比べて十分に短い場合に，提案手法は比較的精度の高い近似解を得るまでの実行時間を削減できる.

4·1 多重化の方法と同期の周期

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) 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

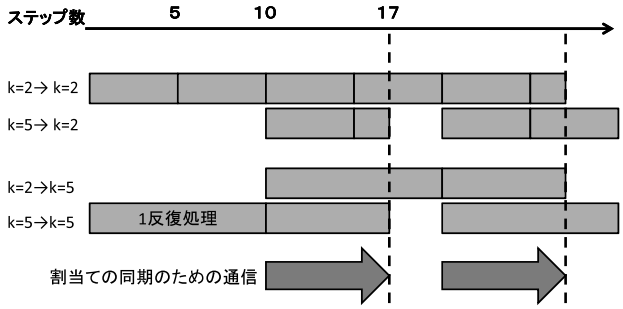

図 4 通信をオーバラップさせた場合の動作概要

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，問題の制約網の直径 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，制約網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である．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) 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，制約網の直径 d を事前に与えられているとする．

$$2 \times \lfloor \frac{k_{max}}{2} \rfloor + k_{max} + 1 + d \qquad (4)$$

4·3 通信のオーバーラップ

4·2 節の方法では，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通信なしに $R(a)$ を求められる理想的な場合と，同じステップ数 (すなわち式 (3)) に抑えることができる．

アルゴリズムの動作を，制約網の直径が 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 (3) 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，制約網の直径が 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．動作の様子を図 4 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，制約網の直径 d は式 (5) を満たさなければならない．

$$d \leq 2 \times \lfloor \frac{k_{max}}{2} \rfloor + k_{max} + 1 \qquad (5)$$

5. 実験と評価

本節では，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 [Modi 05, 飯塚 07]，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 $e_{i,j}$

表 3 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 4 生成した問題

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

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，[Katagishi 07] を参考にした．

例として，表 3 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，制約網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 $k = 1, k = 2, k = 5$ の 1 反復実行に必要なステップ数は，式 (3) 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図 5 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図 6 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．

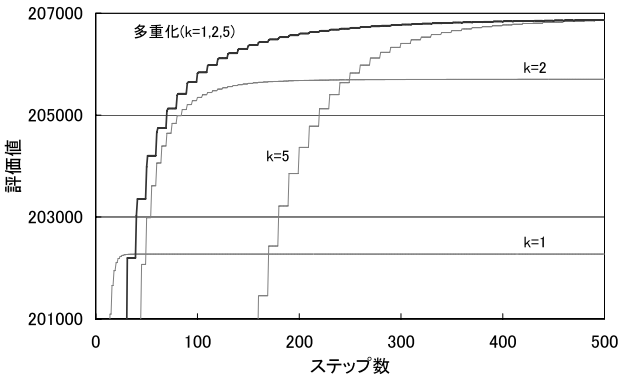

図 5 既存の KOPT アルゴリズムとの比較：ステップ数に対する評価値の推移

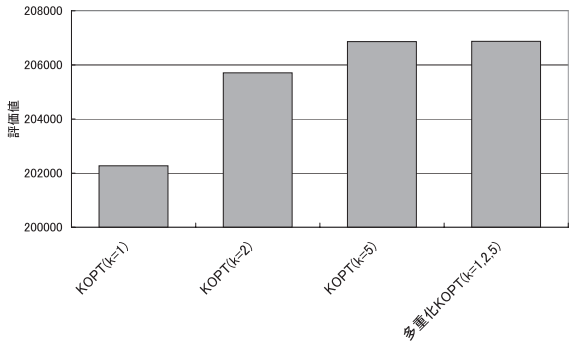

図 6 既存の KOPT アルゴリズムとの比較：停止時点での評価値

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．結果を図 7 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

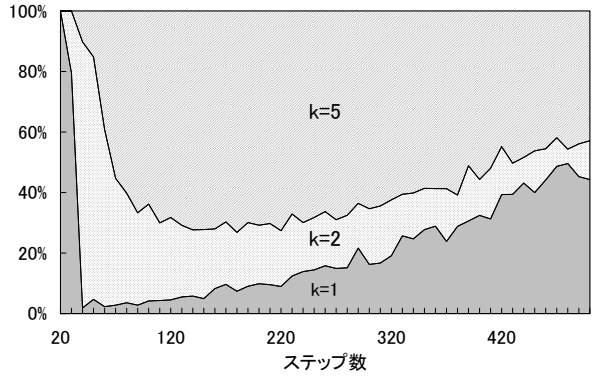

図 7 同期の際に選択されるパラメータの割合

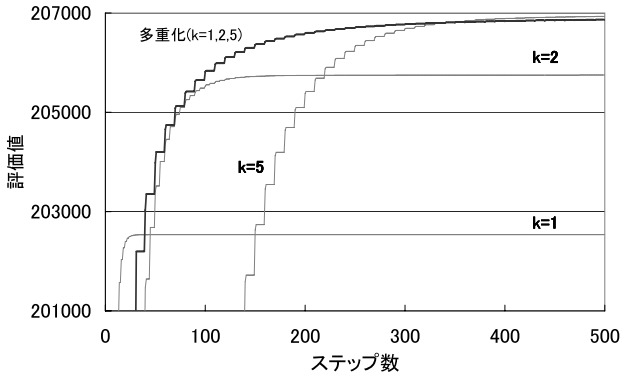

図 8 同一パラメータで多重化した場合との比較：ステップ数に対する評価値の推移

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．結果を図 8 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 節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

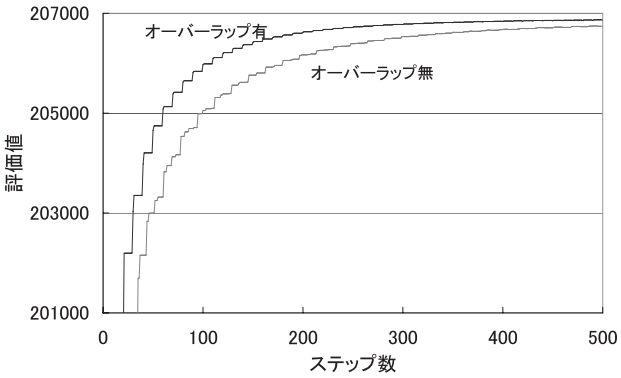

図 9 オーバーラップを行わない場合との比較：ステップ数に対する評価値の推移

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．その結果を図 9 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，評価に用いた制約網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)）が制約網の直径 d に依存し，評価に用いた問題には直径が 7 のものと 8 のものの 2 種類が存在したためである．

5·6 多重化の組み合わせについての考察

本研究の実験では多重化の数を 3 としたが，多重化の数やパラメータ k の組合わせには，検討の余地があるといえる．より大きな k の解法を加えて多重度を増加する場合，解の精度が向上する効果が得られる可能性がある一方で，同期の間隔の調整や，計算量および通信量の増加が問題点となる可能性がある．

例えば，$k=10$ の解法を加えて 4 重化する場合を考えると，式 (3) より，$k=10$ では 1 回の反復に必要なステップ数は 21 であり，同期する間隔を $k=1,2,5$ の場合の 10 から増加する必要がある．このとき，同期の間隔を各 k の反復に要するステップ数の公倍数とするのではなく，最大のステップ数の 21 に近い値に丸めることが適切であると考えられる．また，割り当ての同期をとる通信を隠ぺいするためのオーバラップを行う場合は，投機的に並行する解法の数が 3^2 から 4^2 に増加する．

近似の精度の観点からは，k の値が増加するにつれて精度の改善の程度が小さくなることが予想されるため，ある程度の大きさの k までを適切に組み合わせることが有望である可能性がある．

今後，多重化における組み合わせの影響，および適切な組み合わせについて検討したい．

6. ま と め

本研究では，分散制約最適化問題の k-Optimality[Pearce 07] に基づく解法である KOPT アルゴリズム [Katagishi 07] に注目し，その効率化手法を提案した．既存の KOPT アルゴリズムでは，パラメータ k が大きいほど，1 回の反復処理に必要な通信の範囲が広くなり，解を更新するまでに長い通信時間を必要とする．そこで，KOPT アルゴリズムに対して，複数の異なるパラメータを設定した探索を多重化して実行することにより，解品質の向上と探索時間の抑制を両立する手法を提案した．また，多重化した探索間で同期を取るために必要な通信のステップを，投機的に実行する探索とオーバーラップさせることで隠蔽する手法を提案した．

提案手法を評価するために，計算機シミュレーションを行いその有効性を確認した．探索を多重化した手法は，従来手法と比較して得られる解の品質を保ったまま解の探索時間を削減することができたといえる．このことは，移動する対象物を観測するセンサネットワークにおける観測資源の割り当てなど，ある程度の実時間性のために比較的少ない通信回数で近似解を得たい場合や，評価値が収束する前に探索を打ちきる可能性がある場合などに有用であると考えられる．

本研究では，k-Optimality に基づく解法である KOPT アルゴリズムのみを対象に探索の多重化を行ったが他の Optimality の概念 [Kiekintveld 10] に基づく解法などを組み合わせて多重化する手法や，探索に用いるパラメータ k を問題に応じて動的に選択する手法は今後の課題である．

◇ 参 考 文 献 ◇

[Farinelli 08] 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)

[Katagishi 07] 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)

[Kiekintveld 10] Kiekintveld, C., Yin, Z., Kumar, A., and Tambe, M.: Asynchronous algorithms for approximate distributed constraint optimization with quality bounds, in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pp. 133–140, International Foundation for Autonomous Agents and Multiagent Systems (2010)

[Modi 05] 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)

[Pearce 07] 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)

[Petcu 05] Petcu, A. and Faltings, B.: A scalable method for multi-agent constraint optimization, in *International Joint Conference on Artificial Intelligence*, Vol. 19, p. 266Citeseer (2005)

[P.Gomes 01] P.Gomes, C. and Selman, B.: Algorithm Portfolios, *Artificial Intelligence Journal*, Vol. 126, pp. 43–62 (2001)

[Zhang 02] 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)

[飯塚 15] 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)

[飯塚 07] 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)