平成22年度　　卒業研究論文

分散制約最適化問題のk-optimalityにもとづく近似解法の多重化の検討

指導教官
松尾 啓志 教授
津邑 公暁 准教授

名古屋工業大学　工学部　情報工学科
平成19年度入学　19115152番

柳田 大輝

平成23年2月8日

目 次

第 1 章　はじめに **1**

第 2 章　研究の背景 **3**

2.1　分散制約最適化問題 3

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 5

第 3 章　既存手法: KOPT アルゴリズム **7**

3.1　k-optima 7

3.2　KOPT アルゴリズムによる得られる解 7

3.3　アルゴリズムの動作 8

3.3.1　フェーズ 1 – 情報の交換 8

3.3.2　フェーズ 2 – グループの構成と割当ての計算 8

3.3.3　フェーズ 3 – 合意の形成と変数値の更新 10

3.4　KOPT アルゴリズムの問題点 11

第 4 章　提案手法: 探索の多重化 **12**

4.1　多重化の方法と同期周期 12

4.2　計算と通信に関する複雑度 13

4.2.1　計算量 13

4.2.2　メモリ使用量 14

4.2.3　メッセージサイズ 14

第 5 章　実験と評価 **15**

5.1　ランダムグラフ 15

5.2　格子状グラフ 16

5.3　予備実験 – 既存手法 KOPT アルゴリズムの評価 17

5.3.1　収束までのステップ数の評価 17

5.3.2　パラメータ k によるステップ数に対する評価値の推移の評価 . . 18

5.4　提案手法の評価 19

5.4.1　ランダムグラフに対する実験 19

5.4.2　格子状グラフに対する実験 20

5.4.3　採用されるパラメータの評価 21

5.5　考察 22

第 6 章　まとめ **24**

謝辞 **25**

参考文献 **26**

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．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．

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．

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 [1]．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-optimality 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 [2] が提案された．

従来の KOPT アルゴリズムでは，パラメータ k により，得られる解の質と解探索時間に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を持つ探索を同時に実行する探索の多重化を提案する．提案手法により，解探索時間の短縮や得られる解の品質の改善が期待される．計算機シミュレーションによる評価にもとづいて，提案手法の有効性を考察した．

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述べる．4 章では KOPT の探索を多重化する手法を提案する．5 章では，提案手法を実験により評価する．最後に 6 章で，本研究についてまとめる．

第2章

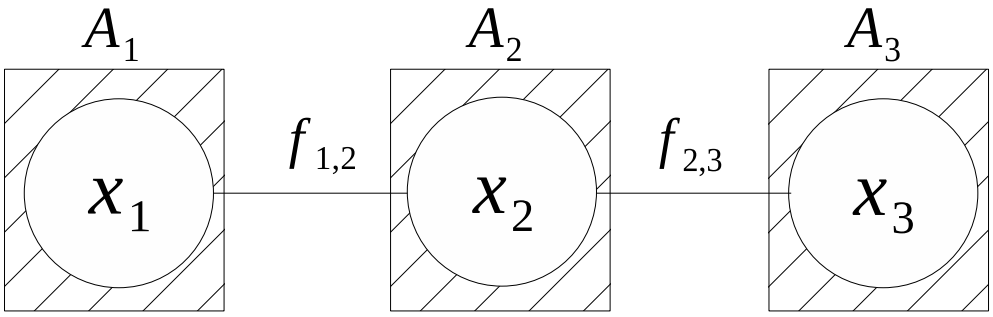
研究の背景

本研究では，分散制約最適化問題の解法の改良手法を提案する．ここで，まず分散制約最適化問題について概説し，従来研究で提案された代表的な解法について述べる．

2.1 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 に含まれる値をとる．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，一つのエージェントに対して一つの変数を配置する．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信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値を a_i と表し，変数の組 $[x_1, \cdots, x_n]$ に対する割当てを $a = [a_1, \cdots, a_n]$ 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．各制約 $c_{i,j}$ に対応する評価関数を $f_{i,j}$ と表す．本研究では，制約で関係する変数を持つ二つのエージェントの間に通信路があるものとする．制約で結ばれた変数をもつエージェントの間に通信路が設定されているものとする．

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 $R(a)$ が最大となるような，大域的に最適な変数値の割当てを求めることである．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



	0	1
0	2	0
1	0	3

$f_{1,2}$

	0	1
0	1	0
1	0	2

$f_{2,3}$

図 2.1: 分散制約最適化問題の例

を変更し，探索処理を実行する．

分散制約最適化問題の例を図 2.1 に示す．図中のグラフの頂点がエージェントと変数を表し，辺が制約と通信路を表す．

この例では，エージェント数は 3，変数の値域は 2 である．エージェント A_1，A_2 がそれぞれ変数 x_1, x_2 を持つ．変数 x_1 と x_2 は，制約で関係する．この制約の評価関数は $f_{1,2}$ により表される．$f_{1,2}$ は，図中の表に示すように，x_1，x_2 が取る値に応じた評価値を返す．同様に，変数 x_2，x_3 に関する制約の評価関数は $f_{2,3}$ により表される．制約で関係する変数を持つエージェント A_1，A_2 間および A_2，A_3 間には通信路がある．変数の組合せ $[x_1, x_2, x_3]$ に対する変数値の割当てが $a = [1, 1, 0]$ であるとき，$f_{1,2}(1, 1) = 3$，$f_{2,3}(1, 0) = 0$ となり，評価値の和は，$R(a) = 3$ となる．この問題で最適解は $a = [1, 1, 1]$ であり，その時の評価値の和は $R(a) = 5$ である．

表 2.1: 厳密解法と非厳密解法の比較

	厳密解法	非厳密解法
計算量 メモリ使用量 通信量	指数関数的に増大	線形に増加
得られる解	最適解	近似解
アルゴリズムの例	DPOP,ADOPT	DSA, Max-Sum

2.2 分散制約最適化問題の解法

従来研究における分散制約最適化問題の解法について説明する．これらの解法は，必ず最適解を得ることができる厳密解法と，近似解を得る非厳密解法に分類される（表2.1）．この分類について詳しく述べる．

2.2.1 厳密解法

必ず最適解を得ることができる厳密解法には，ADOPT[3]，DPOP[4] などが提案されている．これらのアルゴリズムは，エージェント数や制約数，変数の値域といった問題規模に伴い，探索に必要な反復処理，メモリ使用量，総メッセージ数，メッセージサイズ，もしくはそのいずれかが指数関数的に増加するという問題がある．そのため，問題の規模が大きな問題を対象とする場合や，アルゴリズムの実装におけるデバイスの性能が大きく制限される場合には適用できない．

2.2.2 非厳密解法

非厳密解法は，最適解に到達するとは限らないが，比較的少ない計算量やメモリ，メッセージサイズで高速に準最適な解を得ることができる．そのため，問題の規模や複雑さ，アルゴリズム実装における計算，通信資源の制限がある場合は非厳密解法の有効性が期待される．非厳密解法には，DSA[5] や Max Sum[6] などのアルゴリズムが提案されている．

DSA は確率的なアルゴリズムであり，各エージェントは隣接するエージェントの状態を基に，自身の変数値を確率的に変更する．このアルゴリズムでは，メッセージの内容が変数値の情報のみであるので，通信コストを小さく抑えることができる．また，各エージェントが行う計算量やメモリ使用量も小さい．そのため，大規模な問題に対しても適用することができる．しかし，隣接するエージェントの変数値のみを考慮するので，局所的に最適な解に陥りやすい．また，評価値の合計が増減をくり返し収束しない場合がある．

第3章

既存手法: KOPTアルゴリズム

非厳密解法によって得られる解の品質の指標として，近年，k-optimality[1] が提案された．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．本章では，最適性の指標である k-optima について説明する．次に，既存手法である KOPT アルゴ リズム [2] の特徴と動作について述べる．

3.1 k-optima

k-optimality[1] 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，評価値の和 $R(a)$ 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-optimum な割当ては，次のように定義される．

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

この指標によると，厳密解法により得られる解は n-optimal といえる．

3.2 KOPTアルゴリズムによる得られる解

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 k-optimal な解を導くアルゴ リズムである．また，評価値の合計が

メッセージ通信回数に伴い，単調に増加する．そのため，DSA などの確率的アルゴリズムと異なりどの時点でアルゴリズムの動作を停止させても，その間の変数に対する割当てで最も高い評価値を得る割当てで停止する．

3.3 アルゴリズムの動作

KOPT アルゴリズムでは，各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，順序関係のある ID が予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の単位として，くり返し処理を実行する．1回の反復処理に必要な通信ステップ数は k に依存し，$2*\lfloor k/2 \rfloor + k + 1$ ステップである．各フェーズの動作を順に説明する．

3.3.1 フェーズ 1 – 情報の交換

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 3.1 に示す．ここで $k = 2$ である．変数 x_1 を管理するエージェント A_1 に注目すると，A_1 が入手する情報は変数 x_2, x_3, x_4 の値である．この情報は，次のように A_1 に伝えられる．まず 1 ステップ目に A_3, A_4 から x_3, x_4 の情報が A_2 に送られる．このとき A_1 は，A_2 とのメッセージ通信により x_2 の情報を入手する．2 ステップ目に A_2 は，1 ステップ目に受信した情報を中継して隣接するエージェントに送信する．こうして，A_1 は，A_2 によって中継された x_3, x_4 の情報を入手する．

3.3.2 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 2 で情報を入手できた変数を持つエージェントの中から，$k-1$ 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

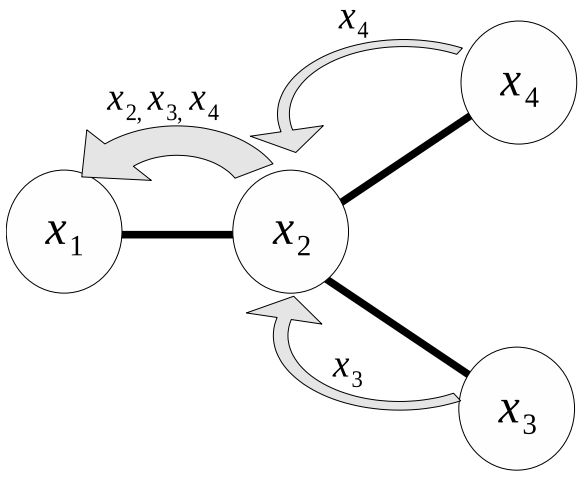


図 3.1: エージェント A_1 が受信する変数値（$k = 2$ のとき）

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，アクティブエージェント (active agent) と呼ぶ．そして，このアクティブエージェントに隣接しているアクティブエージェントでないエージェントをスタティックエージェント (static agent) と呼ぶ．アクティブエージェントとスタティック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，自身をメディエータとす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であるグループのメンバーでもある．

6 個のエージェントが順に制約で結ばれており $k = 3$ のとき，変数 x_3 を管理するエージェント A_3 がメディエータとなるグループは図 3.2 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，アクティブエージェントの持つ変数値のみを変更可能とし，スタティックエージェントの持つ変数値は変更しない．この計算には，制約最適化問題を解く任意の厳密解法を解法を用いることができる．ここでは，最適な割

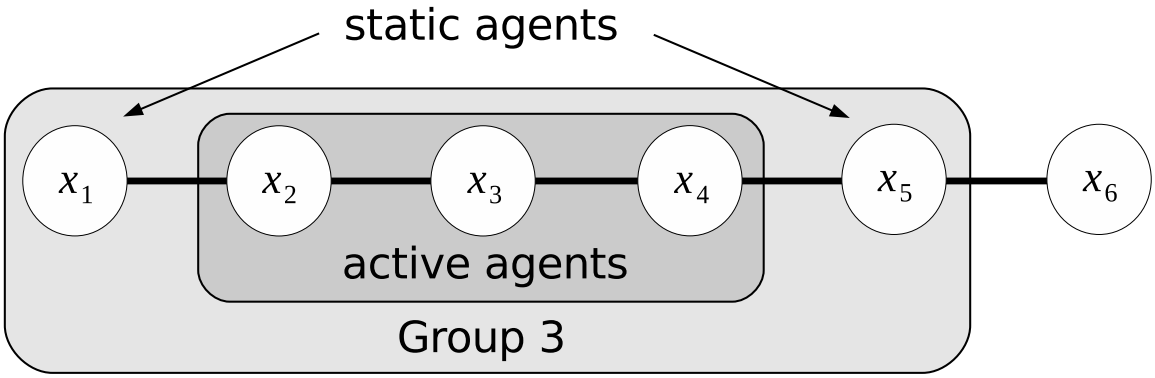


図 3.2: x_3 を中心とするグループ (k=3 のとき)

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．

3.3.3 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，各エージェントは，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，グループの調停者となるエージェントの ID が最も小さいものを選ぶ．

続いて，選択したグループが提示した自身の変数値に対する割当てを，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アクティブ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，そのグループの全てのアクティブメンバー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，変数値を変更することはしない．この 3 フェーズを繰り返すことで，k-optimality な解に近づく．

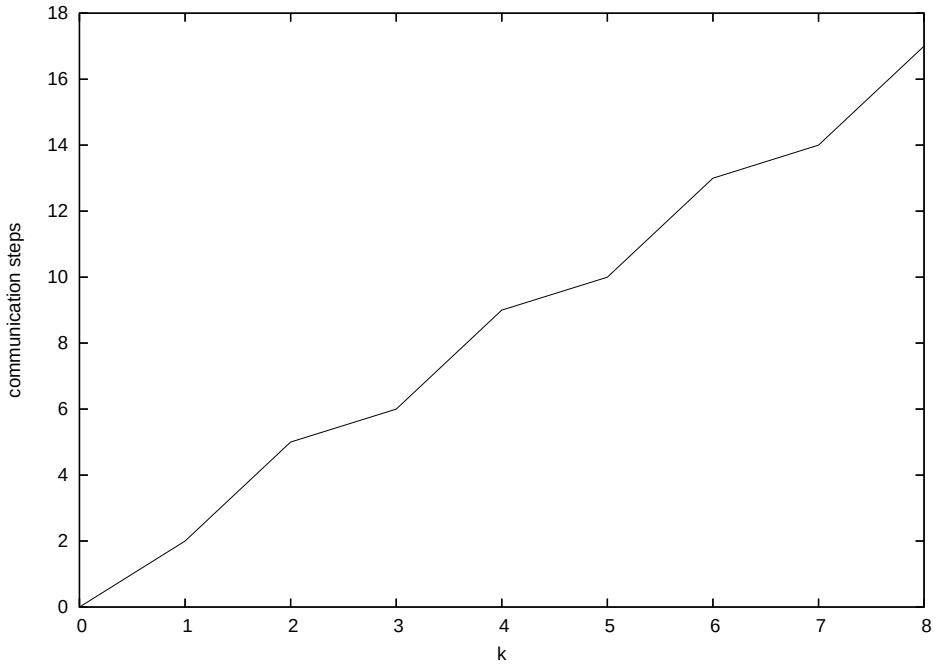


図 3.3: k に対する 1 回の反復処理に必要な通信ステップ数

3.4 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，$2\lfloor k/2 \rfloor + k + 1$ の式により増大する．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，変数値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 3.3 に示す．

また，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，評価値の和が上昇する速度が遅くなる．

第4章

提案手法: 探索の多重化

本研究では，KOPT を拡張し，パラメータ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．

前章で述べた既存の KOPT アルゴリズム [2] の問題点を解消し，得られる解の品質を維持しつつ，評価値の和の上昇速度を既存手法に比べ向上させるため，KOPT アルゴリズムに対してパラメータの異なる探索の多重化を導入する．アルゴリズムに対して個別にパラメータを設定したものを並行して動作させ最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 [7].

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 アルゴリズムでは，パラメータ k を 1 つだけ選択して実行する．これに対し，多重化するする手法では，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，各パラメータで実行した割当てのうち最もよい割当てを採用し，これを全てのパラメータの実行に反映させる．この操作を「同期を取る」と呼ぶ．この操作は一定周期毎にフェーズ 3 の最後に行うこととする．

本論文では，大局的な情報を取得できるものとして，評価値の和が最も高くなったパラメータでの実行により得られた割当てを，最もよい割当てとした．

同期を取るタイミングは，選択したパラメータ k の 1 回の反復処理に必要なステッ

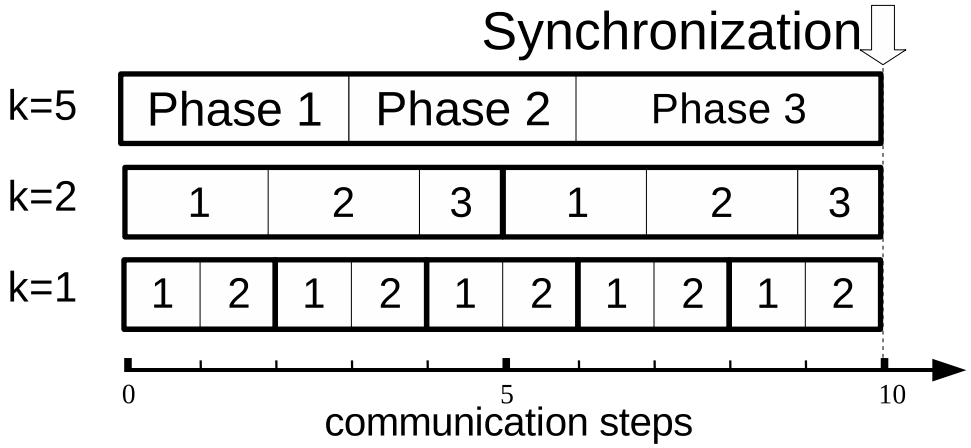


図 4.1: 同期が行われるステップ数（$k = 1, 2, 5$）

プ数の公倍数周期で行う．多重化して実行するパラメータとして $k = 1, 2, 5$ を選択した場合の同期は，10 ステップ毎に行う．これは，1 回の反復処理に必要なステップ数が，$k = 1$ 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 ステップである.（図 4.1）

4.2 計算と通信に関する複雑度

KOPT アルゴリズムにパラメータの多重化を適用した場合に増加する計算資源について説明する．計算量，メモリ量，メッセージサイズについて検討した．これらの使用量は多重化の度合を大きくする程増加するが，これは多重化により期待する効果とのトレードオフであるといえる．

4.2.1 計算量

各エージェントで実行される計算の量は最大で，多重化して実行するパラメータを個別に実行する場合の和になる．これは，各パラメータについての計算は個別に行わなければならないためである．

4.2.2 メモリ使用量

各エージェントが必要とするメモリの量は，多重化して実行するパラメータを個別に実行する場合の和になる．これは，各パラメータにつての状態を個別に記憶しなければならないためである．

4.2.3 メッセージサイズ

1 回の通信で交換されるメッセージサイズは，次のように増加する．フェーズ 1 では変数の値とその制約に関する情報を交換するので，同期を取った直後は複数のパラメータ間で同じ共有できる．しかし，その後はフェーズ 1 が実行されるタイミングがパラメータ間でずれるので，個別に変数の値を交換しなければならない．フェーズ 2 では，各エージェントが計算したグループについての割当てとそれにより増加するグループの評価値の情報を交換する．この情報は，パラメータ間で独立であるので共有できない．フェーズ 3 では，フェーズ 2 を受けて次にとる予定の変数値の情報を交換する．この情報もパラメータ間で共有できない．

以上の理由から，各パラメータについて個別に実行した場合のメッセージサイズの和となる．

第5章

実験と評価

既存の KOPT アルゴリズム [2] と，パラメータの多重化を行った KOPT アルゴリズムを，反復処理のステップ数に対する評価値の和の推移と最終的な値の観点から比較した．分散アルゴリズムの処理の性能の評価は，メッセージ交換の通信回数で行うことが一般的とされており [8]，本論文における評価もこれに従った．

実験は 1 台の計算機上で，シミュレーションによりアルゴリズムを動作させ行った．評価値の和は，シミュレータにより計算した．評価値の和の計算は，既存の KOPT アルゴリズムについては 1 回の反復処理ごとに，多重化した KOPT アルゴリズムについては同期を行う毎に行った．KOPT アルゴリズムには大域的停止アルゴリズムが含まれていないため，予備実験 1 により解が収束するまでに十分なステップ数を調べた．以後の実験では，この結果から 300 ステップに達した時点でシミュレータにより実行を打ち切ることとした．変数の初期値は，エージェント毎にランダムな値で初期化した．

2 種類の問題を対象に実験を行った．以下では，先ず使用した問題の設定について説明する．そして，実験の結果を示す．

5.1 ランダムグラフ

不規則な制約の構造を持つ問題として，ランダムに制約が張られたグラフを用意した．エージェント数 n と制約数 m を設定し，制約が m 本に達するまでランダムに 2 エージェントを選択し，そのエージェントがもつ変数間に制約を張る．

評価関数は，$f_{i,j}$ における変数 x_i，x_j の組み合わせに対して，1 から 10 までの整数

表 5.1: 評価関数の例

$x_i \setminus x_j$	0	1	2
0	3	1	5
1	4	9	6
2	10	2	7

値から同じ値を選ばないようにランダムに評価値を割当てる．例として図 5.1 に示すような評価関数が作成される．この評価関数は，$x_i = 2$ かつ $x_j = 0$ のとき評価値 10 を返す．評価関数は制約ごとに作成した．変数の値域は 3 である．この問題の作成方法は [2] に従った．

5.2 格子状グラフ

規則的な制約の構造を持つ問題として，図 5.1 のように 4 方向に制約辺が存在する格子状のグラフを用いた．評価関数はランダムグラフの場合と同様の方法で作成した．変数の値域は 3 である．

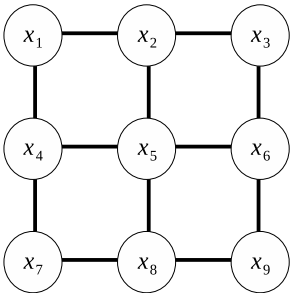


図 5.1: 格子状グラフ（$n = 9$）

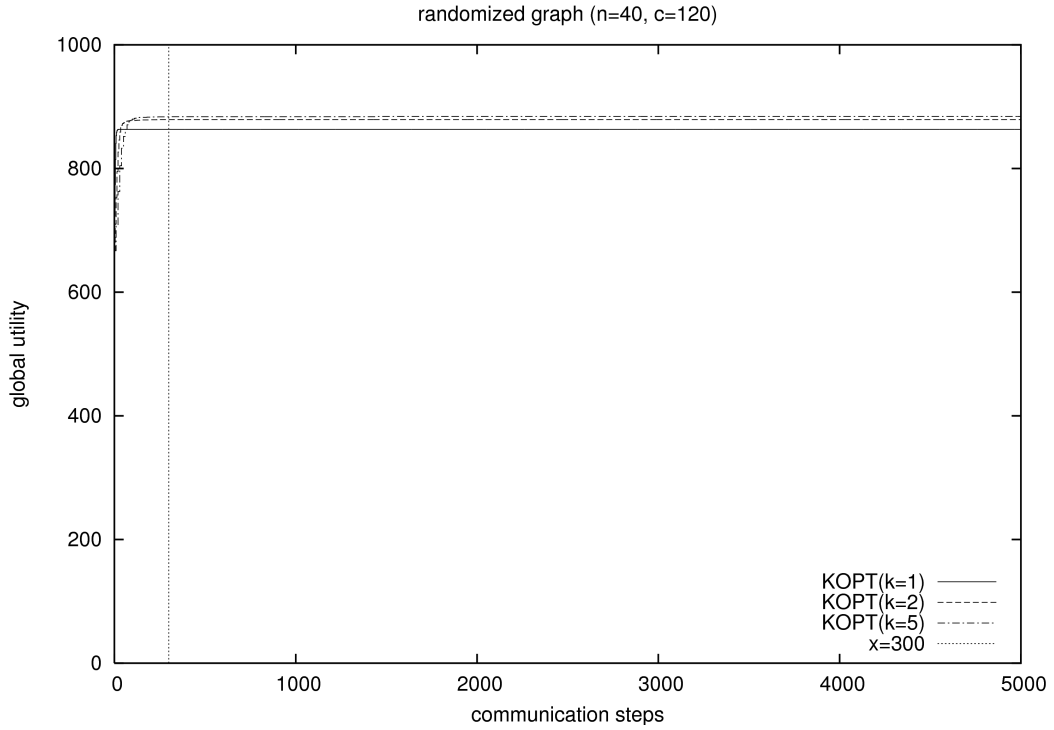


図 5.2: 予備実験 1 収束に必要なサイクル数

5.3 予備実験 – 既存手法 KOPT アルゴリズムの評価

5.3.1 収束までのステップ数の評価

以後の実験で用いるシミュレーションの打ち切りステップ数を決定するため，既存の KOPT アルゴリズムがどの程度のサイクル数で収束するかを確認した．対象とするグラフは，ランダムグラフとしエージェント数 $n = 40$，制約数はその 3 倍の $m = 120$ とした．グラフは 10 個用意し，各グラフについてそれぞれパラメータ $k = 1$ と 5 を設定し 10 回ずつ実行した．打ち切りステップ数は 5000 ステップとした．結果を図 5.2 に示す．この結果より，$k = 5$ の場合，解を収束させるためには 300 ステップ実行すれば十分であると考えられる．したがって，以後の実験では打ち切りステップ数を 300 ステップと設定した．

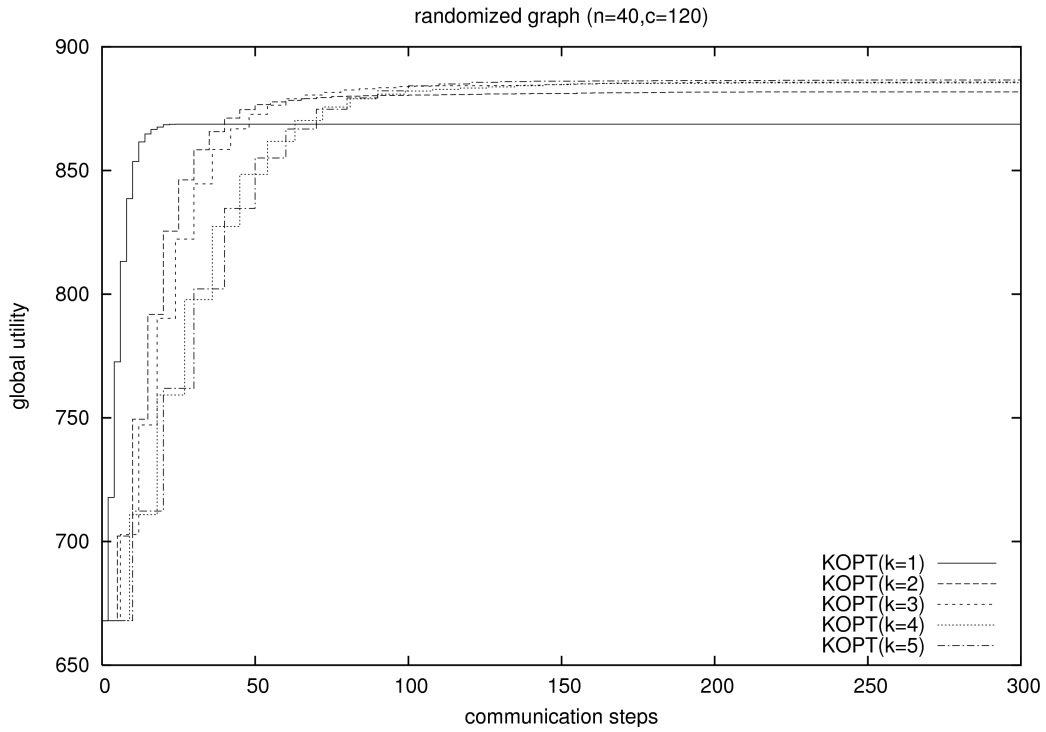


図 5.3: 予備実験 2 パラメータ k 間の関係（ランダムグラフ）

5.3.2 パラメータ k によるステップ数に対する評価値の推移の評価

ランダムグラフに対して既存手法のパラメータ k を変更した場合に，ステップ数に対する評価値の推移がどのように変化するかを調べた．ステップ数に対する評価値の推移をプロットした結果を図 5.3 に示す．また，最終的な評価値の和として 300 ステップ目の値を表 5.2 に示す．

$k = 1$ の場合，評価値は開始直後から大きく増加したが，24 ステップ目ごろにその増加は止まった．したがって，k が小さい場合は評価値の和の増加が速く，k が大きい場合は最終的な評価値の和が大きいといえる．

$k = 2, 3$ および $= 4, 5$ の評価値の和はそれぞれ近い伸びかたをした事がわかる．これは図 3.3 で示した通り，k を 2 増やす毎に 1 くり返し単位に必要なステップ数が大きく増加するためである．

表 5.2: 予備実験 2 最終的な評価値の和

手法	評価値の和
KOPT $(k=1)$	868.77
KOPT $(k=2)$	881.81
KOPT $(k=3)$	885.84
KOPT $(k=4)$	885.48
KOPT $(k=5)$	886.62

5.4 提案手法の評価

従来手法と比較して，評価値の収束速度と収束時の評価値がどの程度変化するかを調べた．

5.4.1 ランダムグラフに対する実験

多重化による効果を調べるため，ランダムグラフに対して実験を行った．グラフの設定は予備実験 2 と同様とした．多重化を適用した手法についてはパラメータ $k=1,2$ および $k=1,2,5$ の組み合わせについて実験した．実験の結果を図 5.4 に示す．既存の KOPT アルゴリズムと比較するために，予備実験 2 の結果も重ねてプロットした．

パラメータ $k=1,2$ で多重化した場合に収束した評価値は，既存手法の $k=2$ で実行した場合に収束する評価値と近い値に収束した．

表 5.3: 実験 1 最終的な評価値の和

手法	評価値の和
KOPT $(k=1)$	868.77
KOPT $(k=2)$	881.81
KOPT $(k=5)$	886.62
KOPT 多重化 (k=1,2,5)	884.76
KOPT 多重化 (k=1,2)	881.00

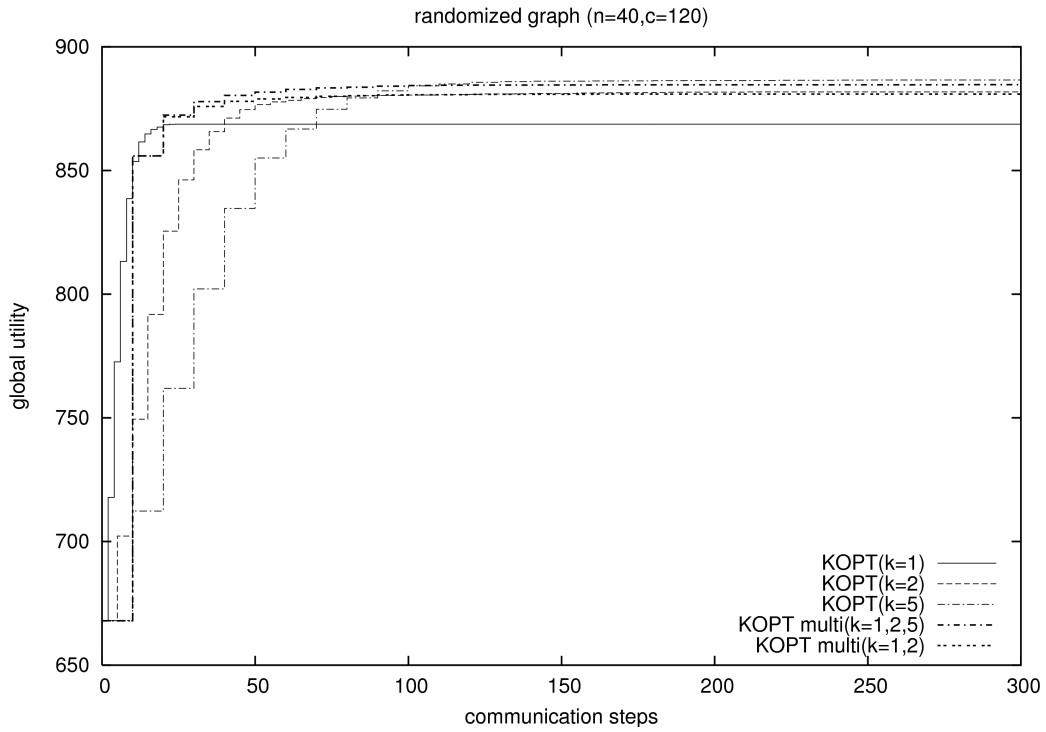


図 5.4: 多重化を適用（ランダムグラフ）

5.4.2 格子状グラフに対する実験

異なる制約構造を持つ問題に対する効果を調べるため，格子状グラフに適用した場合について実験した．エージェント数 $n = 40$，制約数 $m = 67$ と設定した．

既存手法については $k = 1$ から 5 まで，多重化を適用した手法についてはパラメータ $k = 1, 2$ および $k = 1, 2, 5$ の組合せで実行した．

結果を図 5.5 に示す．実験 1 と同様に最終的な評価値の和は，多重化した手法の結果と，並列実行するパラメータ k のうち最も大きな k を単独で実行した結果がほぼ一致する．実験 1 のランダムグラフに対する場合と同様に有効であるといえる．

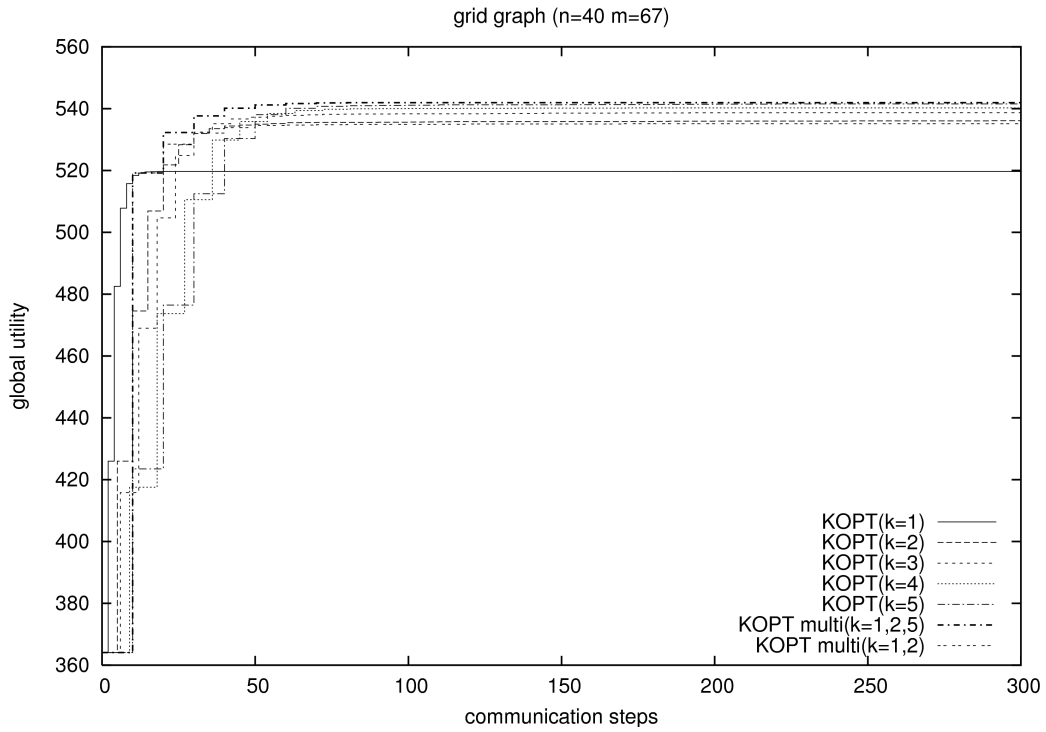


図 5.5: 多重化を適用（格子状グラフ）

5.4.3 採用されるパラメータの評価

ステップ数に対して，同期を取る際にどのパラメータの割当てが採用されるかの推移を調べた．実験に用いる問題は 5.4.1 と同様に作成し，パラメータ k は，1,2,5 で多重化した．横軸を実行開始からのステップ数，縦軸をそのパラメータが採用された回数とした．10 個の問題インスタンスに対し，それぞれ 10 回ずつ実行した．結果を図 5.6 に示す．実行開始直後の同期 (10 ステップ目) では，100 回とも $k = 1$ の割当てが採用された．2 回目の同期（20 ステップ目）では，$k = 2$ の採用回数が 42 回と最も多く，$k = 1$ が 37 回，$k = 5$ が 17 回と続いた．4 回目の同期（40 ステップ目）では，$k = 5$ の採用回数が 25 回と最も多かった．この結果から，はじめは小さな k によって導かれた割当てが採用されるが，狭い範囲である程度割当てが調整されると，小さな k では解の改善が行えなくなり，大きな k の割当てが採用されるようになったと考えられる．

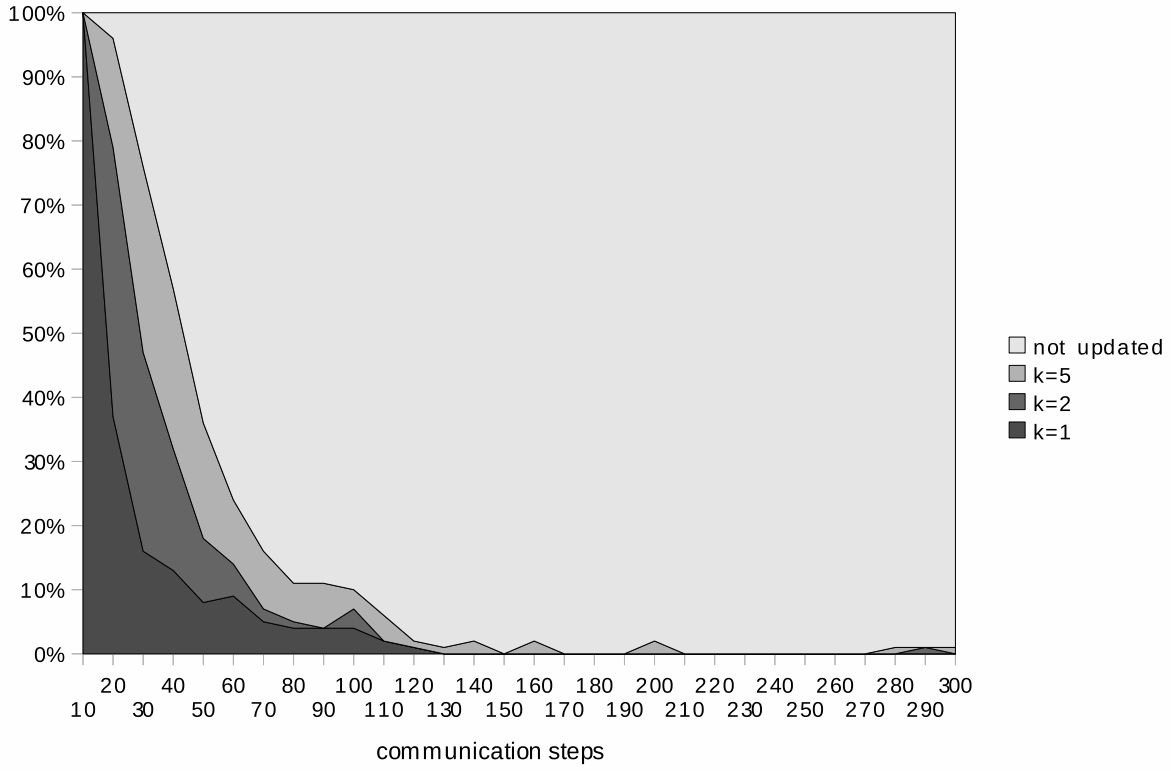


図 5.6: ステップ数に対する採用されるパラメータ k の推移

5.5 考察

多重化して実行した手法では，短い時間で近似解を得たい場合，収束する前に計算を打ち切る可能性がある場合などに有用であると考えられる．

評価実験の結果から多重化した手法では，収束した評価値の和は，並行して実行するパラメータのうち最も大きな k を用いて既存の KOPT アルゴリズムを実行した場合に近い値を示した．多重化することで，収束する解の質が大きく低下することは起こらないと言える．

また，評価値の和が上昇する速度は，既存の手法に比べて速くなった．これは，既存手法では大きな k で実行を開始した場合に初期状態から合意を形成して評価値を改善するのに時間がかかるのに対して，多重化することで序盤は小さな k により狭い範囲で合意を形成することで解の改善を短時間に行えるためと考えられる．

本実験では，同期を行う際に大域的な情報を使用しており，実際の分散環境に適用する場合は，この点について改良が必要である．

第6章

まとめ

本論文では，分散制約最適化問題の k-optimality[1] にもとづく近似解法である KOPT アルゴリズム [2] に注目し，複数の異なるパラメータを設定したアルゴリズムを並列に実行する手法を提案した．提案手法により解の収束速度および収束時の品質に与える影響を検証するために，複数の異なる構造を持つ問題を対象に計算機シミュレーションを行った．実験により，収束時の品質を維持したまま，解の収束速度を向上せる事が可能であることを示した．

今後の課題としては，大域的な情報を使用せずエージェント間のメッセージ通信のみから，どのパラメータの割当てを採用し同期を取るかを決定する方法の検討が挙げられる．また，本研究では並行する各探索におけるパラメータ k は静的に決定されているが，実行中にエージェントの判断により動的に変更することで，より効率よく計算を行うことができるのではないかと考えられる．

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，松井俊浩助教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・津邑研究室および齋藤研究室の方々に深く感謝致します．

参考文献

[1] 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).

[2] 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).

[3] 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).

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

[5] 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).

[6] 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).

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

[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).