分散制約最適化問題の解法 Max Sum における評価関数の緩和手法

川東　勇輝[†a)]　　松井　俊浩[†b)]　　松尾　啓志[†c)]

Methods for relaxing utility-function of Max Sum algorithm

Yuuki KAWAHIGASHI[†a)], Toshihiro MATSUI[†b)], and Hiroshi MATSUO[†c)]

Abstract.
マルチエージェントシステムにおける協調問題解決を表す基本的な枠組みとして，分散制約最適化問題の研究が行われている．分散制約最適化問題の解法 Max Sum は，複雑な制約網の問題に適用した場合に解の精度が低下する．この解の精度の低下は，制約網において自変数と制約がある隣接変数間の制約を評価関数に含める解法 MS-Stable により改善される．しかしながら，評価関数で用いる制約が増えることにより各エージェントにおいて評価される部分問題の規模が拡大する問題がある．そこで本研究では，評価関数において隣接変数間の制約の一部を評価から除外する手法，制約網の構造によって使用する評価関数を切り替える手法，周辺関数の値に基づいてサイクル毎に評価関数を切り替える手法の 3 つの手法を提案する．また，組み合わせ最適化問題である頂点彩色問題を例題として用いて，これら 3 つの手法の解の精度，計算量を比較し，解の精度と計算量のトレードオフが得られることを示す．

Keywords. 分散制約最適化　エージェント協調技術　基礎理論　Max Sum

1. はじめに

近年，無線デバイスやセンサーネットワークなどを用いた自然現象のモニタリング [1] やレスキューロボット [2] [3] などのマルチエージェントシステムが注目されている．マルチエージェントシステムでは，管理コスト，耐故障性，スケーラビリティなどに問題がある，集中的な処理ではなく，複数のエージェントに分散された処理が協調的に実行される．これらのマルチエージェントシステムで協調的に解決されるべき代表的な問題のいくつかは，分散制約充足/最適化問題によって定式化できる [4] [5] [6] [7]．分散制約最適化問題は，マルチエージェントシステムにおける協調問題解決を表す基本的な枠組みであり，理論的な基礎として研究されている．

分散制約充足/最適化問題の解法は厳密解法と非厳密解法の 2 つに分類できる．厳密解法には ADOPT [4] や DPOP [5] がある．これらは必ず最適な解を導くことができるが，探索に要する反復処理，記憶およびメッセージサイズのいずれかが，問題の規模に対して指数関数的に増加する．一方，DSA [8] や Max Sum [6] などの非厳密解法は，必ず最適な解を発見するとは限らないが，計算，記憶資源およびメッセージサイズは比較的少ない．

本研究では，Max Sum に注目する．Max Sum は確率伝搬に基づく手法であり，各エージェントは周囲から伝搬される解の評価値を考慮して自変数値を決定する．そのため問題に対応するグラフ上において隣接エージェントの状態のみに基づいて解を決定する DSA と比較すると，精度の高い解が得られることが期待できる．またマルチエージェントシステムが，電力や演算処理能力，通信帯域幅，メモリ使用量などに制限があるセンサーや無線デバイスなどを用いて構成される場合には，デバイスの性能の制限も満たすアルゴリズムを用いることが望ましい．このような場面では，比較的限られた，計算，記憶，通信資源のもとで実行できる Max Sum は有利であると考えられる．しかし Max Sum は複雑な制約網を持つ問題に適用した場合に，解の精度が低下する問題がある．その解の精度の低下は，制約網において隣接する変数間の制約も評価に含める評価関数 MS-Stable によって改善できるが，隣接変数間の制約を評価に含めることによっ

† 名古屋工業大学, 〒 466-8555 名古屋市昭和区御器所町
a) E-mail: yuuki@matlab.nitech.ac.jp
b) E-mail: matsui.t@nitech.ac.jp
c) E-mail: matsuo@nitech.ac.jp

て，各エージェントが評価する部分問題の規模は増大する [6]．そこで本研究では，評価関数に用いる制約を状況により変更し，解の精度と問題の規模のトレードオフを調整できる評価関数の緩和手法を提案する．

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 Max Sum,MS-Stable およびグラフの頂点彩色問題への適用方法について述べ，4. 節では評価関数の変更により解の精度と計算量を調整する手法 k-GMSS，問題の制約に対応するグラフの構造に応じて異なる評価関数を用いる手法 D-MSS，周辺関数に基づいてサイクル毎に評価関数を切り替える手法 Z-MSS の 3 つの提案手法について述べる．5. 節では従来手法，提案手法の比較・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

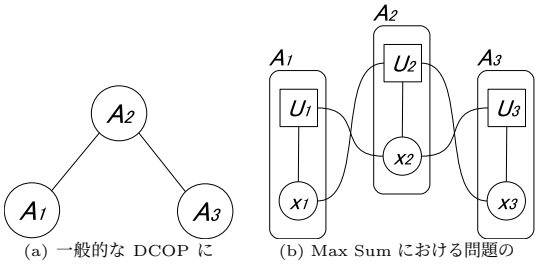
分散制約最適化問題は，エージェントの集合 A，変数の集合 X，各変数 $x_i \in X$ の値域 D_i，制約の集合 C，各制約に対応する評価関数の集合 F からなる．各変数は制約により他の変数と関係する．本研究では，簡単のために二項制約のみを扱う．x_i,x_j に関する制約を $c_{i,j} \in C$ により表す．各制約 $c_{i,j}$ に対応する評価関数 $f_{i,j} \in F$ により，変数値の割り当て $\{(x_i, d)(x_j, d')\}$ の評価値が定義される．すべての制約についての評価関数の値の合計を，最大化するような，変数値の割り当てを求めることが目的である．

制約・評価関数はエージェントに分散されて配置される．変数はエージェントの状態を表し，その変数を保持しているエージェントのみがその値を変更できる．各エージェントは自身と関係する制約の情報のみを持つ．各エージェントは制約で接続されているエージェントと，メッセージを交換しつつ，自変数値を決定する．本研究では，各エージェント $a_i \in A$ は一つの変数 x_i を持つものとする．このため，必要に応じて，表記の簡単のために，エージェントと変数，また後述の頂点彩色問題における頂点を同一のものとして扱う．

3. 従来手法と頂点彩色問題への適用

3.1 Max Sum Algorithm

Max Sum は情報理論の分野で用いられている確率伝搬に基づく手法であるが，これを分散制約最適化問題の近似解を求めるために用いる [6]．Max Sum は関数ノード U と変数ノード x からなる factor グラフ上でメッセージを伝搬する．そのため，一般的な分散制約最適化問題のグラフの表現を，関数ノード U と変数ノード x からなる factor グラフとして表す必要がある．一般的な分散制約最適化問題のグラフ表現では，図 1(a) のようにノードがエージェント，辺が制約を表すように表現される．一方 Max Sum では，図 1(b) のように各エージェントは，factor グラフ上では自身の変数ノード x と，制約と評価関数を表す関数ノード U を持つように表現される．Max Sum では変数ノード x，関数ノード U の間でのメッセージの交換によって，大域的に準最適な評価値となる解を得る．



(a) 一般的な DCOP における問題のグラフ表現　(b) Max Sum における問題のグラフ表現

図 1　問題のグラフ表現

Max Sum の処理を大きく分類すると，変数ノード x_n から関数ノード U_m へのメッセージ $Q_{n\to m}(x_n)$ の計算・送受信，関数ノード U_m から変数ノード x_n へのメッセージ $R_{m\to n}(x_n)$ の計算・送受信，周辺関数 Z_n の計算の 3 つに分けられる．変数ノード x_n から関数ノード U_m へのメッセージは，変数 x_n の各値について評価値 $Q_{n\to m}(x_n)$ を伝達する．関数ノード U_m から変数ノード x_n へのメッセージは変数 x_n の各値についての評価値 $R_{m'\to n}(x_n)$ を伝達する．変数ノード x_n から関数ノード U_m へ伝達される，評価値 $Q_{n\to m}(x_n)$ の値は，それまでに x_n が受信した，$R_{m'\to n}(x_n)$ の総和に基づいて次式のように計算される．

$$Q_{n\to m}(x_n) = \alpha_{nm} + \sum_{m' \in M(n)\setminus m} R_{m'\to n}(x_n) \quad (1)$$

ここで $M(n)$ は factor グラフ上で変数 x_n の近傍である，関数ノードの添え字の集合を表す．前述のようにエージェントは関数ノード x と変数ノード U を持つため，生成される factor グラフは必ずサイクルを含む．サイクルを伝搬するメッセージによって，評価値が無限に増加しないように，$Q_{n\to m}(x_n)$ の計算では α_{nm} を加えて正規化を行う．α_{nm} は次の式を満たすように選ばれる．

$$\sum_{x_n} Q_{n\to m}(x_n) = 0 \quad (2)$$

次に関数ノード U_m から変数ノード x_n へのメッセージ $R_{m\to n}(x_n)$ は，宛先の変数ノード x_n が各変数値を取った時に，その制約によってどの程度の評価を得られるかを送信する．この評価値は宛先の変数ノード x_n を除く，U_m と隣接する各変数ノード $x_{n'}$ からのメッセージ $Q_{n'\to m}(x_{n'})$ の総和と評価関数 U_m に基づいて次式のように計算される．

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m \setminus n} \left(U_m(\mathbf{x}_m) + \sum_{n' \in N(m)\setminus n} Q_{n'\to m}(x_{n'}) \right) \tag{3}$$

ここで $N(m)$ は factor グラフ上で関数ノード U_m の近傍である，変数ノードの添え字の集合を表す．$\mathbf{x}_m$ は評価関数 U_m に関係する全ての変数を表し，$\max_{\mathbf{x}_m\setminus n}$ は $\mathbf{x}_m$ から x_n を除いた変数の値を変化させたうち最大となるものを選択することを意味する．

Max Sum において最も計算量が必要となるのは，関数ノード U_m から変数ノード x_n への評価値 $R_{m\to n}(x_n)$ の計算である．式 (3) は factor グラフ上で U_m と隣接する変数ノードの数に対して指数関数的な計算量を必要とする．その一方で，問題に含まれる変数の総数には影響を受けない．

周辺関数は関数ノード U_m から変数ノード x_n への評価値 $R_{m\to n}(x_n)$ から計算される．周辺関数は変数 x_n が全体に与える影響を表していて，次のように定義される．

$$Z_n(x_n) = \sum_{m \in M(n)} R_{m\to n}(x_n) \tag{4}$$

式 (4) によって計算される周辺関数は，評価値の最大値を計算できるはずであるが，前に述べた通り Max Sum では問題の表現においてサイクルを含み，それに伴い値の正規化を行っているので式 (4) で計算される周辺関数の値は近似値となり，$Z_n(x_n) \approx \max_{\mathbf{x}_m\setminus n} \sum_{m=1}^{M} U_m(\mathbf{x}_m)$ を表す．

3.2 グラフの頂点彩色問題への適用

グラフの頂点彩色問題とは，与えられたグラフにおいて辺で接続された頂点同士が異なる色に彩色されるような，各頂点の組み合わせを求める問題であり，組み合わせ最適化問題の例題として用いられる．分散制約最適化問題では，グラフの各頂点の色は，その頂点に対応する変数を持つエージェントのみが決定できる．頂点彩色問題における頂点の色はエージェントの変数の値として表され，各エージェントは変数値 $x_m \in \{1, \cdots, c\}$ のいずれかを選択する．頂点彩色問題のグラフにおいて隣接する頂点同士の変数値が同じであれば，その変数値の組み合わせは彩色問題の制約に違反する．各頂点に対応するエージェントの評価関数は次のように示される．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m)\setminus m} x_m \otimes x_i \tag{5}$$

$$x_i \otimes x_j = \begin{cases} 1 & (x_i = x_j) \\ 0 & (x_i \neq x_j) \end{cases} \tag{6}$$

ここで $\gamma_m(x_m) \ll 1$ は，各変数値の優先度を表し，同じ違反数となる対称解を除くために用いる．Max Sum ではこの評価関数の大域的な合計を最大化し，違反が最小となるような変数値の組み合わせを求めることを目的とする．しかしこの評価関数を複雑な問題に用いた場合，高い精度の解が得られず，解の収束に多くの反復を要するか振動する場合が多いことが知られている．この問題は後述する評価関数 MS-Stable により改善される．今後必要に応じて，式 (5) を用いて Max Sum アルゴリズムを実行することを単に Max Sum と記述する．

3.3 MS-Stable

Max Sum の解の精度を改善するために拡張された評価関数 MS-Stable は，Max Sum では評価に用いられていなかった，制約網の隣接変数間の制約を評価する．MS-Stable の評価関数は次の式で表される．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m)} \sum_{j \in C(i,m)} x_i \otimes x_j \tag{7}$$

ここで，$C(i,m)$ は次式のように表される(注1)．

$$C(i,m) = \{ l \in N(m) | l > i \wedge (i \in N(l) \vee l \in N(i)) \} \tag{8}$$

MS-Stable の評価関数を使うことによって，複雑な制約網での解の精度の低下，および解の収束速度が改善される．一方，Max Sum では式 (3) の計算のために，factor グラフ上で関数ノード U_m と隣接する複数の変数ノード $\mathbf{x}_m$ からメッセージを受け取り，$\mathbf{x}_m$ の変数値の割り当ての組み合わせから，$R_{m\to n}$ が最大となる組み合わせを探索するため，U_m に隣接する変数ノー

(注1)：式 (8)(12) において，$i \in N(l)$ ならば $l \in N(i)$ となるので冗長となるが，この表記は文献 [6] に基づいている．

頂点数	10	12	15	18	20
変更前 Max Sum	5.03	4.73	4.91	4.58	4.34
変更後 Max Sum	5.04	4.74	4.90	4.58	4.34

表 1　変更前と変更後の Max Sum の平均違反度

ドの数に応じて計算量が増加するが，MS-Stable では U_m に隣接する変数ノード間の制約も考慮に入れるため，さらに多くの変数値の組み合わせを探索する必要があるという問題点がある．具体的には Max Sum では式 (3) の計算に必要な変数値の組み合わせ数は

$$\sum_{i \in N(m) \backslash m} |D_m| \cdot |D_i| \tag{9}$$

である. ただし式 (9) は, 式 (3) の最大化を, 隣接する変数ノードとの制約毎に独立して計算するように変更した場合の組み合わせ数である. 予備実験により, このときの結果が変更前とほぼ等しいことを確認した (表 1). 本研究では Max Sum として上記のものを用いる. 一方,MS-Stable を用いた場合の変数値の組み合わせ数は最大の場合, 次式に増加する.

$$\prod_{i \in N(m)} |D_i| \tag{10}$$

4. 提 案 手 法

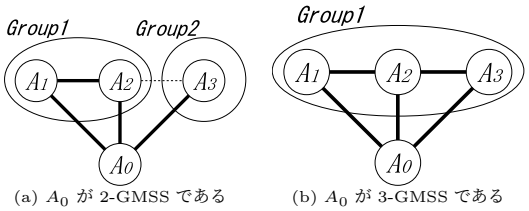
4.1 評価関数の部分的な緩和による問題規模の削減手法

3.2 節で述べたように，Max Sum では複雑な制約網の場合に解の精度が低下し，MS-Stable では解の精度は改善されるが計算量が増加するという問題があった．そこで提案手法 k-GMSS では，頂点彩色問題のグラフ上で隣接する頂点間の制約の一部のみを評価関数で用いることによって，Max Sum で問題であった解の精度と MS-Stable の計算量のトレードオフを実現する．k-GMSS では，アルゴリズムの前処理として，各エージェントによって頂点彩色問題のグラフにおける隣接する頂点を k 個組のグループに分ける．k-GMSS ではグループ内の制約を評価に含め，グループ間の制約は評価に含めない．すなわち同じグループに属する隣接頂点間の制約は評価に考慮されるが，異なるグループに属する隣接頂点間の制約は考慮されない．k-GMSS は次の式で表される．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m)} \sum_{j \in L(i,m)} x_i \otimes x_j \tag{11}$$

ここで，$L(i,m)$ は次の式で表される．

$$L(i,m) = \{l \in N(m) | l > i \wedge (i \in N(l) \vee l \in N(i)) \wedge (G(m,i) = G(m,l))\} \tag{12}$$



(a) A_0 が 2-GMSS である場合のグループ化と考慮される制約

(b) A_0 が 3-GMSS である場合のグループ化と考慮される制約

図 2　k の増減によるグループと用いられる制約の変化

$G(m,i)$ は factor グラフ上で関数ノード U_m の近傍にある，変数ノード x_i が属するグループの番号を表す．

MS-Stable では頂点彩色問題における全ての隣接頂点間の制約を評価に含めるため，隣接する頂点数において指数関数的に計算量が増加するが，k-GMSS では隣接頂点間の制約は一部しか考慮されないため，計算量の増加を抑えられる．また k-GMSS ではグループに含める数 k を増減させると，解の精度と計算量のトレードオフを調整できる．例えば図 2(a) の 4 頂点からなるグラフを例に説明する．A_0 の頂点に対応するエージェントにおける評価関数を 2-GMSS とする．A_0 から見た隣接頂点は A_1,A_2,A_3 の 3 つであり，2-GMSS では最大 2 個をひとつのグループとして扱うので，A_1,A_2 は Group1 に，A_3 は Group2 となる．2-GMSS で評価に用いられる制約を図 2(a) において太線で書かれた辺で表す．2-GMSS では，元々の Max Sum で用いられる隣接頂点との制約と同じグループに属する A_1 と A_2 との制約が評価に用いられる．一方，図 2(b) では同じグラフにおいて A_0 に 3-GMSS を用いた場合の例である．3-GMSS では最大 3 つの頂点をひとつのグループとして扱うので，A_1,A_2,A_3 の 3 つの頂点が Group1 に属する．したがって A_0 の 3-GMSS では 2-GMSS で用いる制約に加えて，A_2 と A_3 との制約も用いられるようになる．このように k を増減させることによって，評価関数において用いられる制約数を増減できる．なお，k を隣接する頂点数の最大数に設定した場合には MS-Stable と同じ制約が用いられる．

k-GMSS を使用する場合には，異なるグループ間の制約を評価に含めない．したがって，式 (3) を最大化する組み合わせを探索するときに，グループ毎に独立して計算できるため，計算量の増加を抑えることができる．k-GMSS における変数値の組み合わせ数は最大

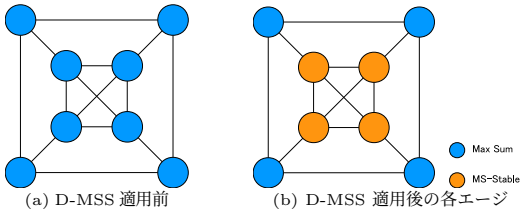
$$\sum_{g \in G(m)} |D_m| \cdot \prod_{i \in g} |D_i| \tag{13}$$

となる．ここで $G(m)$ は U_m に関係するグループの

添え字の集合である．式 (13) は MS-Stable の最大の組み合わせ総数に比べると小さい．

4.2 制約網の構造による評価関数の切り替え手法

3.2 節で述べたように Max Sum は複雑な制約網に適用した場合に性能が低下してしまう [6]．ここでは複雑な制約網とは，充足解が無いような過制約で，サイクルが多く含まれる制約網のことを意味する．5.1 節に示す予備実験により，3 色の頂点彩色問題では，4 頂点以上からなる完全部分グラフに含まれる頂点に対応するエージェントが，評価関数を Max Sum から MS-Stable に変更すると特に違反数の増加の抑制に効果がある傾向がみられた．そこで本研究では，頂点彩色問題におけるグラフ上で，4 頂点以上の完全部分グラフに含まれる頂点に対応するエージェントであるかを判定し，頂点が 4 頂点以上の完全部分グラフに含まれている場合は，MS-Stable などのより制約を多く用いる評価関数に変更する手法 D-MSS を提案する．D-MSS では，Max Sum の前処理として，各頂点が 4 頂点以上からなる完全部分グラフに含まれているかを判定する．各頂点に対応するエージェントは，自らが関係する制約の辺の情報，すなわちどの頂点と制約があるかという情報をメッセージによりフラッディングする．各エージェントは，すべての隣接エージェントからメッセージを受け取った時，周囲の制約の状況から自らが 4 頂点完全部分グラフに含まれるかを判定する．ここで，各頂点に対応するエージェントは 4 頂点の完全部分グラフに含まれていると判定すると，それ以上の探索を行わずに判定の処理を中断する．すなわち 5 頂点以上の完全部分グラフに含まれているかどうかは判定しない．各頂点に対応するエージェントは，対応する頂点が 4 頂点以上の完全部分グラフに含まれていると判断した場合は，MS-Stable や k-GMSS などの制約をより多く評価に用いる評価関数を用いる．このようにすることによって，多くの制約を考慮すべき頂点に対応するエージェントのみで多くの制約を評価するような処理が行われる．その結果，多くの制約を考慮しなくても良いエージェントの計算量の増加が抑えられ，システム全体としては，問題を解くのに必要な計算量を削減できる．本論文では MS-Stable を用いる手法を D-MSS，k-GMSS を用いる手法を D-kGMSS とそれぞれ表記する．また D-MSS の計算量を削減するために，全ての 4 頂点以上の完全部分グラフに含まれる頂点に対応するエージェントが MS-Stable を用いるのではなく，さらに頂点に対応するエージェントの ID が偶数であるエージェントのみが MS-Stable に切り替える手法を D-MSSid と表記する．



(a) D-MSS 適用前の各エージェントの評価関数　(b) D-MSS 適用後の各エージェントの評価関数

図 3 4 頂点の完全部分グラフを含む場合の評価関数の変更

4 頂点の完全部分グラフを含む問題での，D-MSS を適用した例を図 3 に示す．図 3(a) のグラフにおいて，中央部の 4 頂点からなる完全部分グラフに含まれる頂点に対応するエージェントが評価関数を Max Sum から MS-Stable に変更すると，図 3(b) のようになる．

4.3 周辺関数を指標とした評価関数の切り替え

4.1 節では評価関数に用いる制約の数を増減させることにより，計算量と解の精度のトレードオフを行う手法を提案し，4.2 節では頂点彩色問題のグラフ上で頂点がどのような制約の状況にあるかによって，評価関数を変更する手法を提案した．この節では，各エージェントのその時点の計算結果に基づいて，サイクル毎に評価関数を切り替える手法 Z-MSS を提案する．ここでサイクルとは，エージェントの動作の単位を表す．各サイクルでは，各エージェントは変数から関数へのメッセージ送受信，関数から変数へのメッセージ送受信，周辺関数の計算のそれぞれを行う．

Max Sum では，エージェントがある変数値 x を取った時に全体に与える影響を表す周辺関数 $Z(x)$ が最大となる変数値 x を常に推定しその状態を取り続けるが，もし $Z(x)$ が最大となる変数値とその次に大きい変数値の影響が均衡している場合，エージェントの状態が振動することによって，制約違反を起こす場合があると考えられる．そこで Z-MSS では，各エージェントにおいて周辺関数 $Z(x)$ の計算の際に，$Z(x)$ が最大の値となる状態とその次に最大となる状態の差が定数 δ 以内となる時，使用する評価関数を切り替える．$Z(x)$ が最大となるエージェントの変数値 x を x_{max}，その次に大きな値を取る状態を $x_{max'}$ とする．切り替えの判定を Algorithm1 に示す．ここで keepFuncCycle とは，Max Sum から他の評価関数に切り替わった時，その評価関数によるメッセージが周囲に十分伝搬される前に元の Max Sum に戻るのを防ぐ目的で用いる変数である．変数には定数 λ が代入され，切り替わった

Algorithm 1 周辺関数に基づく評価関数の切り替え

```
if Z(x_max) < Z(x_max') + δ then
    use MS-Stable (or k-GMSS) as U_m(x_m)
    keepFuncCycle ← λ
else if keepFuncCycle ≦ 0 then
    use Max Sum as U_m(x_m)
else
    keepFuncCycle ← keepFuncCycle −1
end if
```

評価関数によって評価値が計算されるたびにデクリメントされる．λ が 0 よりも大きい間，すなわち λ サイクル間は Max Sum に戻らないようにする．

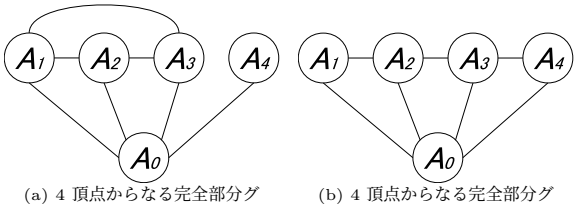
Z-MSS での解の精度と計算量のトレードオフの調整は，周辺関数の均衡を表す閾値 δ と持続サイクル数を表す λ を変化させることで可能である．すなわち δ の値を大きくすると，周辺関数の値が均衡していると判定するエージェントが多くなり，結果多くのエージェントが評価関数を切り替えるので，解の精度は向上し，計算量が増加すると考えられる．また λ の値を大きくすると，一度評価関数を切り替えたエージェントが Max Sum に戻るまで時間を要するので，結果的に評価関数を切り替えているエージェントの割り合いが高くなる．

5. 実験・評価

5.1 予備実験

ここでは 4.2 節で述べた予備実験の詳細について述べる．実験では図 4(a), 図 4(b) に示す 3 色の頂点彩色問題を例題とし，全ての頂点に対応するエージェントが Max Sum を実行した場合と全てのエージェントが MS-Stable を実行した場合の解の精度を比較した．図 4(a) は制約数が 7 の 4 頂点からなる完全部分グラフを含む問題であり，図 4(b) は制約数が 7 の 4 頂点からなる完全部分グラフを含まない問題である．計測の便宜上，マルチエージェントシステム全体の動作を，サイクルを単位として同期した．頂点彩色問題におけるグラフ上で隣接する頂点同士が同じ変数値を選択した場合に制約違反が生じるものとし，各サイクルにおける各エージェントの制約違反の有無を評価した．各アルゴリズムは 50 サイクルで終了するものとし，サイクルあたりの制約違反数を平均した．

図 4(a) において Max Sum, MS-Stable を用いた結果を表 2, 図 4(b) において Max Sum, MS-Stable を用いた結果を表 3 に示す．図 4(a) において Max Sum を用いた場合と MS-Stable 用いた場合では平均違反数に大きな違いが出ているのに対し，図 4(b) において Max Sum を用いた場合と MS-Stable 用いた場合では平均違反数に違いが出なかった．したがって 4 頂点以上からなる完全部分グラフに含まれる頂点に対応するエージェントが MS-Stable などのより詳細に制約を用いる評価関数を用いることは効果があり，逆にそれ以外のエージェントにおいて MS-Stable などの評価関数を用いても効果がないと推測される．



(a) 4 頂点からなる完全部分グラフを含む場合　(b) 4 頂点からなる完全部分グラフを含まない場合

図 4　予備実験における頂点彩色問題のグラフ

手法	平均違反数
Max Sum	1.630
MS-Stable	1.045

表 2　図 4(a) での平均違反数

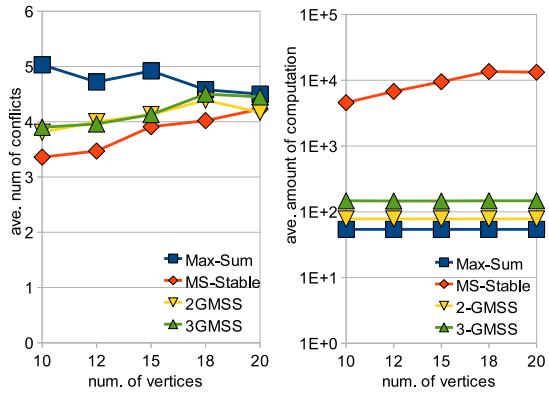
手法	平均違反数
Max Sum	0.039
MS-Stable	0.038

表 3　図 4(b) での平均違反数

5.2 提案手法の評価

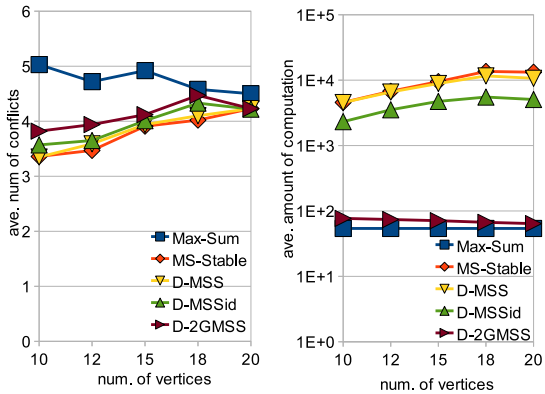
提案手法と従来手法について，解の精度と計算量を 3 色の頂点彩色問題を用いて評価した．予備実験と同様に，各エージェントはサイクルを単位として動作するものとした．また，アルゴリズムは 50 サイクルで終了するものとした．変数の数は 10,12,15,18,20 の 5 つの場合を用いた．制約数は変数の数の 3 倍とした．この制約数の設定では，過制約な問題が生成される．各場合の問題について，50 例をランダムに生成した．また解の精度の指標として，平均違反数を用いた．計算量の指標として，Max Sum の実行で最も計算量が必要となる，1 メッセージで評価される変数値の組み合わせ数をサイクル数および頂点数により平均した値を用いた．なお MS-Stable における変数値の組み合わせ数は，実験の便宜上，式 (10) で示す最大の場合とした．

まず k-GMSS と Max Sum, MS-Stable の 3 つについて評価した．k-GMSS は，ひとつのグループに含める頂点の数を表す定数 k を 2 と 3 に変化させた 2-GMSS と 3-GMSS の 2 つを用いた．平均違反数は図 5(a)，平均の計算量を図 5(b) に示す．図 5(a) では全ての頂点数において，Max Sum の平均違反数が最も多く，MS-Stable の平均違反数が最も少なくなった．頂点数が少ない場合は平均違反数の差が大きく，頂点数が多くなるにつれて差が小さくなった．この理由として，本実験では制約数を変数の個数 (頂点の個数) の



(a) 平均違反数　(b) 平均計算量

図 5　k-GMSS の平均違反数と計算量



(a) 平均違反数　(b) 平均計算量

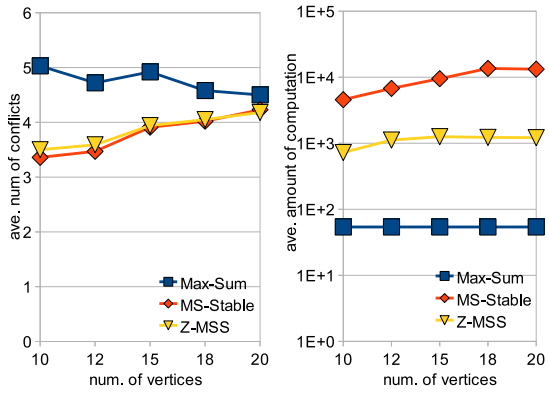
図 6　D-MSS,D-MSSid,D-2GMSS の平均違反数と計算量

3 倍としているため，頂点数が少ない場合には制約の密度が高く，頂点数が多い場合には制約の密度が低くなる．その結果，頂点数が少ないほど頂点彩色問題のグラフ上で隣接する頂点間に制約がある確率が高くなるため Max Sum と MS-Stable の平均違反数の差が大きくなったと考えられ，逆に頂点数が多いと隣接する頂点間に制約がある確率が少なくなり，平均違反数の差が小さくなったと考えられる．

2-GMSS,3-GMSS は全ての頂点数において，Max Sum より平均違反数が少なく，MS-Stable より多くなった．また，図 5(b) で計算量の比較では，上から順に MS-Stable,3-GMSS,2-GMSS,Max Sum となっていて，k-GMSS では平均違反数と計算量のトレードオフがみられる．2-GMSS と 3-GMSS を比較すると，頂点数が少ない場合はわずかに 2-GMSS より 3-GMSS の平均違反数が少ない．頂点数が多い場合には，2-GMSS より 3-GMSS の方が平均違反数が多い．頂点数が大きくなるにつれて制約の密度が下がり，頂点彩色問題のグラフ上で隣接頂点間に制約がある確率が少なくなるのに加え，隣接する頂点数も少なくなるため，3 つの頂点で十分にグループ化できなくなっているものと考えられる．

次に D-MSS,D-MSSid,D-2GMSS の評価関数について評価を行った．D-MSS は頂点彩色問題のグラフ上において 4 頂点以上からなる完全部分グラフに含まれる頂点に対応するエージェントが評価関数 MS-Stable を使用する手法である．D-MSSid は 4 頂点以上からなる完全部分グラフに含まれる頂点で，その頂点に対応するエージェントの ID が偶数であるエージェントが MS-Stable，奇数であるエージェントが Max Sum を使用する手法である．D-2GMSS は 4 頂点からなる完全部分グラフに含まれる頂点に対応するエージェントが 2-GMSS を使用する手法で，4.1 節と 4.2 節の提案手法を組み合わせたものである．図 6(a) に平均違反数，図 6(b) に平均計算量を示す．Max Sum と MS-Stable については，下限と上限の目安として参考に表示した．平均違反数について D-MSS は，全ての頂点数において MS-Stable とほぼ同じ平均違反数となった．計算量について D-MSS を見ると，少ない頂点数では MS-Stable とほぼ同じ計算量となったが，頂点数が増えるにつれ D-MSS の方が計算量が小さくなり，頂点数 20 では約 20%削減できた．D-MSSid の平均違反数は D-MSS よりもわずかに多くなったが，D-MSSid における計算量は，MS-Stable の計算量に比べ 38%〜50%削減することができた．これは 4 頂点の完全部分グラフに含まれる頂点数の約半数のみが MS-Stable を使うことによって計算量が削減されたからであり，またその場合においても解の精度をある程度維持できた．D-2GMSS の平均違反数は D-MSSid と比較してある程度増加したが，その計算量は Max Sum に比べ 18%〜42%の増加に抑えることができた．

最後に Z-MSS について評価を行った．Z-MSS は周辺関数に基づいてサイクル毎に評価関数 MS-Stable と Max Sum とを切り替える手法である．ここで Z-MSS のパラメーターとして，周辺関数がどの程度の差の時均衡とみなすかを決定する定数 $\delta = 0.2$ とし，一度 MS-Stable に切り替わったのち何サイクル間持続させるかを表す定数 $\lambda = 3$ として実験を行った．図 7(a) に Z-MSS の平均違反数を，図 7(b) に計算量を示す．Z-MSS は全ての頂点数において，MS-Stable とほぼ等しい平均違反数となった．さらに計算量においては MS-Stable の 9%〜16%にまで削減できた．この理由として，Z-MSS ではアルゴリズム開始から数サイクルの周辺関数の値が安定していない状況においては，



(a) 平均違反数　(b) 平均計算量

図 7　Z-MSS の平均違反数と計算量

多くのエージェントが MS-Stable 使用するために違反が少ない解にすみやかに到達することが考えられる．逆にアルゴリズム開始からの十分サイクルが経過した，比較的に周辺関数の各値が安定した状況ではほとんどのエージェントが Max Sum を使用する．その結果，解が安定してからは一部の計算を省略することになり，比較的少ない計算量と高い解の精度を両立できたものと考えられる．

6. おわりに

本研究では分散制約最適化問題の解法である Max Sum および MS-Stable の効率化手法を提案した．評価関数の部分的な緩和による問題規模の削減手法である k-GMSS，制約網の構造による評価関数の切り替え手法である D-MSS，周辺関数を指標として評価関数を切り替える手法 Z-MSS とそれらの組み合わせた手法を提案した．提案手法の有効性を頂点彩色問題を用いて評価した．

k-GMSS では，2-GMSS と 3-GMSS において Max Sum と MS-Stable の中間の解の精度と計算量のトレードオフが得られた．しかし頂点数が多い部分においては，3-GMSS よりも 2-GMSS の解の精度が高いことから，同じ k の値における頂点彩色問題のグラフ上での隣接する頂点のグループ化の方法において，まだ検討の余地があると考えられる．D-MSS では MS-Stable に近い解の精度と計算量となったが，D-MSSid や D-2GMSS においてはある程度の解の精度の維持と計算量を削減できた．このことから，ランダムに生成した問題においても，4 頂点以上からなる完全部分グラフに含まれる頂点に対応するエージェントが制約を多く用いる評価関数用いることは効果的であると考えられる．また，その一部の制約を緩和することで解の精度と計算量を調整できることができた．Z-MSS では周辺関数に基づく評価関数の切り替えにより，解が安定しないアルゴリズム開始直後には MS-Stable が多く，解が比較的安定するに従って Max Sum の割り合いが多くなるために，MS-Stable に近い解の精度を少ない計算量で得ることができると考えられる．本研究では，頂点彩色問題について検討したが，異なる評価関数により表される問題にも提案手法を適用できる可能性がある．評価関数が異なる場合，式 (6) を含む一部の定式化が変更される．このときの提案手法の適用方法と評価は今後の課題である．

謝辞

本研究の一部は科学研究費補助金 (基盤研究 C(一般)，課題番号 21500073) の助成による．

文　献

[1] Y. Kim, M. Krainin, and V. Lesser, “Application of Max-Sum Algorithm to Radar Coordination and Scheduling,” Workshop of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems, pp.5–19, 2010.

[2] S. Ramchurn, A. Farinelli, K. Macarthur, M. Polukarov, and N. Jennings, “Decentralised coordination in robocup rescue,” Jan. 2009.

[3] K.S. Macarthur, A. Farinelli, S.D. Ramchurn, and N.R. Jennings, “Efficient, superstabilizing decentralised optimisation for dynamic task allocation environments,” 3rd Int. Workshop on Optimisation in Multiagent Systems at the 9th Joint Conf. on Autonomous and Multiagent Systems, pp.25–21, 2010.

[4] P.J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, “An asynchronous complete method for distributed constraint optimization,” Proc. of 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp.161–168, 2003.

[5] A. Petcu and B. Faltings, “Dpop: A scalable method for multiagent constraint optimization,” Proc. of 19th Int. Joint Conf. on Artificial Intelligence, pp.266–271, 2005.

[6] A.Farinelli, A.Rogers, A.Petcu, and N.R.Jennings, “Decentralised coordination of low-power embedded devices using the max-sum algorithm,” Proc. of 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp.639–646, 2008.

[7] A. Farinelli, A. Rogers, and N. Jennings, “Bounded approximate decentralised coordination using the max-sum algorithm,” IJCAI-09 Workshop on Distributed Constraint Reasoning, pp.46–59, July 2009.

[8] W. Zhang, O. Wang, and L. Wittenburg, “Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance,” Workshop on Probabilistic Approaches in Search AAAI2002, pp.53–59, 2002.