情報処理学会論文誌 Vol.53 No.11 2419–2431 (Nov. 2012)

分散制約最適化問題の解法Max-Sumにおける評価関数の動的な変更手法

川東 勇輝[1] 松井 俊浩[1,a)] 松尾 啓志[1,b)]

受付日 2012年1月27日, 採録日 2012年7月2日

概要：分散制約最適化問題（DCOP）はマルチエージェントシステムにおける分散協調問題解決の基礎的な表現として研究されている．本研究では，近年提案されたDCOPの解法であるMax-Sumアルゴリズムに注目する．Max-Sumは，問題を表現するfactorグラフに含まれるサイクルの数が少なければ比較的精度が高い解を得る傾向があるが，より多くのサイクルを含み，辺の密度が高い部分がある場合には解の精度が低下する．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含める評価関数MS-Stableにより改善されるが，各エージェントが評価する部分問題の規模は拡大する．そこで，解の精度と計算量のトレードオフを利用するために，解法の実行中に2つの評価関数を適応的に切り替える手法Z-MSSを提案する．Z-MSSでは各エージェントの利得の推定値に基づいて，評価関数を動的に変更する．利得の推定値により，詳細な評価が必要な部分に対してMS-Stableが割り当てられるため，計算量の増加を抑えつつ，比較的高い解の精度を得ることが期待される．また，Z-MSSはパラメータによって，ある程度は解の精度と計算量を調整でき，その適切な設定値は問題の種類やグラフの構造に依存する．頂点彩色問題および重み付きの二項制約からなる問題において，Z-MSSおよびそのパラメータの効果を実験により評価する．

キーワード：Max-Sum，分散制約最適化問題，分散協調問題解決，マルチエージェント

Dynamically Changing Utility-functions in Max-Sum Algorithm

Yuuki Kawahigashi[1] Toshihiro Matui[1,a)] Hiroshi Matuo[1,b)]

Received: January 27, 2012, Accepted: July 2, 2012

Abstract: Distributed Constraint Optimization Problems (DCOPs) have been studied as a fundamental model of multi-agents cooperation. We address the Max-Sum algorithm that has been proposed as a solution method for DCOPs. The Max-Sum algorithm generates relatively better approximate solutions when it is applied to less cyclic factor graphs. However, in the case that the graph contains many cycles and dense parts, the quality of the solution decreases. The solution quality can be improved by using extended utility function MS-Stable that additionally evaluates constraints among neighborhood nodes. On the other hand, the MS-Stable increases the size of the local problem that is computed in each agent. To employ this trade off, we propose Z-MSS that suitably switches the both types of utility-functions while the solution method is running. In Z-MSS, each agent switches utility functions based on the estimation values of its utilities. As a result, MS-Stable is applied to the parts of the problem that need detailed evaluation. The effects of Z-MSS and its parameters are experimentally evaluated applying it to graph coloring problems and weighted binary constraint problems.

Keywords: Max-Sum, DCOP, distributed cooperative problem solving, multi-agent

1. はじめに

近年，低消費電力の無線デバイスやセンサネットワークを用いた自然現象のモニタリング [1] や，自律的に動作す

[1] 名古屋工業大学
Nagoya Institute of Technology, Nagoya, Aichi 466–8555, Japan
a) matsui.t@nitech.ac.jp
b) matsuo@nitech.ac.jp

るロボットを用いた災害救助 [2], [3] を行うための技術として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，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ため，集中的な処理システムに比べ，管理コスト，耐故障性，スケーラビリティの点において比較的優れている．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], [8]．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 [5] や DPOP [6] 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 [9] や Max-Sum [7] 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．後述するように，Max-Sum は問題によっては最適解を得るが，一般には局所最適解に収束したり振動したりすることがあるため，ここでは非厳密解法に分類した．

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，高い解の精度が得られる場合があると考えられる．このような比較は文献 [7] の実験にも示されている．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 を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．ここで，複雑な制約網とは，サイクルを比較的多く含み，制約辺が密な部分を持つような問題を意味する．また，このような制約網の問題を単に複雑な問題と呼ぶ．Max-Sum 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 によって改善することができるが，隣接変数間の制約を評価に含めることによって，各エージェントの計算量は増大する [7]．これらの解の精度と計算量のトレードオフを調整する手法として，Z-MSS を提案する．Z-MSS は各エージェントの利得の推定である周辺関数の値を指標として，評価関数を動的に切り替えて用いる手法である．利得の推定を指標として評価関数を適切に選択することにより，計算量の増加を抑えつつ，高い解の精度が得られる．また Z-MSS は評価関数の変更のパラメータを調整することにより，ある程度，解の精度と計算量を調整できる．そのパラメータを厳密に設定しなくても，Max-Sum と MS-Stable がある程度は切り替わる範囲に設定できれば，両者の中間的な性質による効果が多少は得られると考えられる．しかし，より性能を向上させるためには利得の推定に影響を与えるグラフの構造や制約の評価値などに応じた適切な値を設定することが望ましい．そこで，従来手法と提案手法を頂点彩色問題と二項制約からなる最適化問題において評価する．さらに提案手法については，複数の種類の問題やグラフの構造を用いて実験し，適切なパラメータについて調査する．

Max-Sum [7] の関連研究には，連続値変数の問題への適用 [10]，近似アルゴリズム [8]，動的な問題の変化に追従する手法 [3]，多目的最適化問題のための拡張 [11] などが提案されている．また，応用的な問題として，センサネットワークや無人航空機による観測システム [1], [12]，ロボットによる災害救助 [2] などへの適用が研究されている．本研究の方針はこれら関連研究とは異なる．提案手法の新規性は，文献 [7] において提案された MaxSum と MS-Stable を，エージェントの局所的な知識に基づいて，解法の実行中に切り替えることにより，両者のトレードオフを利用しようとする点にあると考えられる．

本論文は以下のように構成される．2 章では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 Max-Sum と MS-Stable および，グラフの頂点彩色問題と二項制約の最適化問題への適用方法について述べる．4 章では評価関数を動的に変更する手法 Z-MSS を提案する．5 章では，従来手法と提案手法について頂点彩色問題と制約の重みを変化させた二項制約の問題での評価を行い，Z-MSS の効果について考察する．6 章においては，異なる種類の問題と特徴的なグラフの構造に対して実験し，Z-MSS のパラメータを変化させた場合の変化について考察する．7 章においてこれらをまとめる．

2. 分散制約最適化問題

分散制約最適化問題（DCOP）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，簡易化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，変数値の割当て $\{(x_i, d)(x_j, d')\}$ の評価値が定義される．ただし，評価値は非負の実数である．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，変数値の割当てを求めることが目的である．

一般的な DCOP の表現では，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決定できる．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 1 つの変数 x_i を持つものとする．このため，必要に応じて，表記の簡易化のために，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．

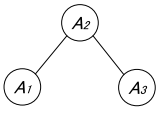
3. 従来手法–Max-Sum と DCOP への適用

3.1 Max-Sum Algorithm

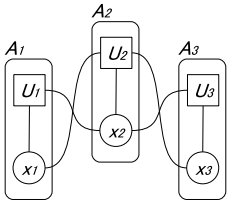
Max-Sum は情報理論の分野で用いられている確率伝搬に基づく手法であり，これを分散制約最適化問題の近似解を求めるために用いる [7]．Max-Sum は関数ノード U と変数ノード x からなる factor グラフ上でメッセージを伝搬する．そのため，一般的な分散制約最適化問題のグラフの表現を，関数ノード U と変数ノード x からなる factor グラフとして表す必要がある．

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と変数，辺が制約を表すように表現される．その一方で Max-Sum では，factor グラフによる表現が用いられる．図 1 (a) に対応する factor グラフの例を同図 (b) に示す．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，大域的な評価値が計算され，その評価値に基づいて各エージェントの変数値が決定される．factor グラフにサイクルがない場合は，大域的に最適な解が得られる．しかし，サイクルがある場合には，準最適解に収束したり，振動したりする場合がある．

なお，DCOP のグラフ表現である制約網が木であれば，サイクルを含まない factor グラフを作ることは可能である．ただし本研究では，文献 [7] に従い，図 1 (b) のような factor グラフを用いる．また，制約網にサイクルを含む問題を対象とする．そのため本論文では，factor グラフにサイクルが含まれる場合についてのみ議論する．



(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

Fig. 1 Representations of problems.

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．

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

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，本研究で用いる factor グラフはサイクルを含む．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．

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

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 が各変数値をとったときに，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．

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

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 に関係するすべての変数を表し，$\max_{\mathbf{x}_m\setminus n}$ 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，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，ここでは factor グラフがサイクルを含むこととともない，評価値を正規化している．そのため，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．

各エージェントは，$Z_n(x_n)$ が最大となる x_n の値を選択することにより，（準）最適解を得る．

3.2 二項制約の問題への適用

2 章で述べたように，二項制約の分散制約最適化問題では，変数 x_i，x_j についての制約は $c_{i,j}$ で表され，$c_{i,j}$ に対応する評価関数 $f_{i,j}$ により，変数値の割当て $\{(x_i, d)(x_j, d')\}$ の評価値が定義される．そして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，変数値の割当てを求めることが目的である．これに Max-Sum を適用する場合，各エージェントにおける評価関数は次のように定義される．

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

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．変数値の優先度は，$\gamma_m(x_m)$ により任意に与えられる．一般には，優先度と変数値の相関は特に考慮されない．

また，Max-Sum は，評価関数の値の合計を最大化するような，変数値の割当てを求めるアルゴリズムである．そのため，コストの最小化問題などに適用する場合には，評価関数値の大小関係を逆転する．そして，評価関数の値の合計を最大化することで，コストの合計を最小化する．

3.3 頂点彩色問題への適用

二項制約で表される問題の 1 つに，グラフの頂点彩色問題がある．この問題では，与えられたグラフにおいて，辺で接続された頂点どうしを異なる色に彩色することが目的である．分散制約最適化問題では，頂点の色はそれぞれの頂点に対応するエージェント m が持つ変数 x_m の値として表される．隣接する 2 頂点の変数値が異なれば評価値が大きくなるように，評価関数を定義する．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．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m) \setminus m} x_m \otimes x_i \tag{6}$$

このとき

$$x_i \otimes x_j = \begin{cases} 1 & (x_i = x_j) \\ 0 & (x_i \neq x_j) \end{cases} \tag{7}$$

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合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る場合，高い解の精度が得られず，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) もしくは式 (6) 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．

3.4 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 [7] は，Max-Sum の評価関数では考慮されていなかった，自エージェントの変数と制約網で隣接する 2 変数間の制約を考慮する．二項制約による最適化問題の場合，MS-Stable の評価関数は次の式で表される．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) + \sum_{i \in N(m)} \sum_{j \in C(i,m)} f_{i,j}(x_i, x_j) \tag{8}$$

特に，彩色問題へ適用する場合には

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m)} \sum_{j \in C(i,m)} x_i \otimes x_j \tag{9}$$

となる．ここで，$C(i,m)$ は次式のように表される[*1]．

$$C(i,m) = \{l \in N(m) | l > i \wedge (i \in N(l) \vee l \in N(i))\} \tag{10}$$

ただし，$l > i$ は，式 (8) において，同一の変数の組合せに対する $f_{i,j}$ が 2 回現れることを避けるための，タイブレークである．このタイブレークのために，エージェントの識別名の順序関係が仮定される．

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，$\mathbf{x}_m$ の変数値の割当ての組合せから，$R_{m \to n}$ が最大となる組合せを探索するため，U_m に隣接する変数ノー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合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合せ数は

$$\sum_{i \in N(m) \setminus m} |D_m| \cdot |D_i| \tag{11}$$

である．ここで D_m は，式 (5) で定義される評価関数 $U_m(\mathbf{x}_m)$ に対応する，エージェント m が持つ変数 x_m の値域である．また，D_i は，エージェント m と制約で関係す

[*1] 式 (10) 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，この表記は論文 [7] に基づいている．

るエージェント i が持つ変数 x_i の値域である．計算量は主に組合せ数に依存し，$|D_i|$，$|D_m|$ が一定であれば，$|N(m)|$ について線形のオーダとなる．

ただし式 (11) は，次のように冗長な計算を省いた Max-Sum を使用したときの組合せ数である．まず，式 (3) に式 (5) を代入し変形すると次式となる．

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m \setminus n} \Bigg(Q_{m\to m}(x_m) + \gamma_m(x_m) + \sum_{i\in N(m)\setminus m,n} \big(f_{m,i}(x_m, x_i) + Q_{i\to m}(x_i) \big) + f_{m,n}(x_m, x_n) \Bigg) \tag{12}$$

ここで $\max_{\mathbf{x}_m \setminus n}$ を計算するうえで，上段の項は x_m にのみ依存し，中段の項は x_m と x_i に依存し，下段の項は x_m と x_n に依存する．ここで x_n は引数として与えられるので上段と下段の項は x_m にのみ依存する．そこで依存関係のない項の max を削除すると，式 (12) は

$$R_{m\to n}(x_n) = \max_{x_m} \Bigg(Q_{m\to m}(x_m) + \gamma_m(x_m) + \sum_{i\in N(m)\setminus m,n} \big(\max_{x_i} (f_{m,i}(x_m, x_i) + Q_{i\to m}(x_i)) \big) + f_{m,n}(x_m, x_n) \Bigg) \tag{13}$$

となり，式 (11) の計算量が導きだされる．一方，MS-Stable を用いた場合には前述のような省略ができないため，隣接するエージェントの変数すべての組合せを計算しなければならない．そのため，計算すべき変数値の組合せ数は最大の場合

$$\prod_{i\in N(m)} |D_i| \tag{14}$$

まで増加する．計算量は主に組合せ数に依存し，$|N(m)|$ について指数のオーダとなる．

4. 提案手法

本研究では，高い解の精度と計算量の削減を両立させるために，実行の途中で動的に評価関数を変更する手法 Z-MSS を提案する．

4.1 基本的なアイデア

Max-Sum は比較的小さい計算量で高い解の精度が得られる手法であるが，複雑な制約網の問題に適用した場合には解の精度は低下する．解の精度の低下の原因の 1 つとして，式 (5) に示される評価関数が比較的簡単であることがあげられる．この評価関数では，自エージェントの変数と制約網において隣接する，2 変数間にある制約が考慮されない．このような 2 変数間の制約は，式 (8) に示されるように MS-Stable の評価関数では考慮されるため，MS-Stable は Max-Sum よりも解の精度の低下を抑制できると考えられる．しかし，3.4 節で述べたように，解法における最も計算量が必要となるメッセージ R の計算において，このような 2 変数の組すべての制約を考慮することが大幅な計算量の増加の原因となる．

そこで，Max-Sum における複雑な制約網における解の精度の低下と，MS-Stable における計算量の大幅な増加を系全体では抑制する方法として，部分的に MS-Stable を割り当てる方法が考えられる．そのとき問題となるのが，何を指標としてどの部分に MS-Stable を割り当てることが有効であるかということである．理想としては，大域的な情報，すなわちグラフの構造や現在の利得，各エージェントの計算結果などに基づいて判断し，最適なエージェントにおいて MS-Stable を用いることが望ましい．しかし，分散環境では，大域的な情報はメッセージを用いて取得する必要があり，そのような情報の取得は通信遅延などの観点から難しい場合があると考えられる．そのため，局所的な情報に基づいて評価関数の変更を判断することが望ましい．より局所的な情報を用いる方法として，周囲のエージェントの制約の状況や評価値を収集することが考えられるが，いずれも追加される処理のための通信と計算が新たに生じる一方で，必ずしも有用な情報が得られるかは不明である．そこで Z-MSS では，Max-Sum で用いている周辺関数 $Z_i(x_i)$ に基づいて評価関数を変更することを提案する．周辺関数 $Z_i(x_i)$ は，各エージェントが状態 x_i を選択したときの利得の推定値を表す．Z-MSS は，周辺関数 $Z_i(x_i)$ に基づいて評価関数の変更をするため，新たなメッセージなどを追加する必要はない．

4.2 周辺関数の均衡

Z-MSS では，周辺関数 $Z_i(x_i)$ による評価関数の変更の判断基準として，周辺関数の均衡という概念を用いる．周辺関数 $Z_i(x_i)$ は，式 (4) で表され，エージェントが状態 x_i を選択した場合の全体の利得の推定値を表す．Max-Sum に基づく解法では，周辺関数 $Z_i(x_i)$ が最大となる状態 x_i を選択することで，準最適解を導く．周辺関数の均衡とは，周辺関数 $Z_i(x_i)$ が最大となる状態 x_i^{max} と次に最大となる $x_i^{max'}$ がある程度近い値であることを意味し，次式により定義される．

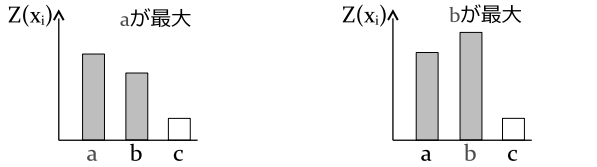
$$Z_i(x_i^{max}) - Z_i(x_i^{max'}) < \delta \tag{15}$$

ここで，x_i^{max} と $x_i^{max'}$ は

$$x_i^{max} = \arg\max\{Z_i(x_i) \mid x_i \in D_i\}$$

$$x_i^{max'} = \arg\max\{Z_i(x_i) \mid x_i \in D_i \setminus x_i^{max}\}$$

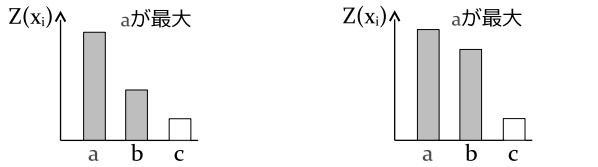
である．δ は周辺関数の均衡の検出の閾値を表す定数で，評価関数の切り替わりやすさを制御するパラメータである．



(a) Max-Sum で評価した場合の $Z_i(x_i)$　(b) MS-Stable で評価した場合の $Z_i(x_i)$

図 2 周辺関数が最大となる変数値が異なる例

Fig. 2 Different maximum assignments between Max-Sum and MS-Stable.



(a) Max-Sum で評価した場合の $Z_i(x_i)$　(b) MS-Stable で評価した場合の $Z_i(x_i)$

図 3 いずれの評価関数でも周辺関数が最大となる変数値が変化しない例

Fig. 3 The same maximum assignments between Max-Sum and MS-Stable.

周辺関数が均衡している場合に，大域的な解の精度が低下する場合について考察する．

周辺関数の均衡している状況で，大域的な解の精度が低下する場合として，Max-Sum に基づいて計算された周辺関数の最大である値 $Z_i(x_i^{max})$ よりも，Max-Sum で評価されていない制約を含めた場合の $Z_i(x_i^{max'})$ が大きい場合が考えられる．この場合では，本来は $x_i^{max'}$ を選択する方が利得が大きいことになる．たとえば，Max-Sum のメッセージに基づいて計算された周辺関数 $Z_i(x_i)$ が均衡している場合を**図 2** (a) に示す．図 2 (a) では x_i^{max} は状態 a で，$x_i^{max'}$ は b である．ここで MS-Stable を用いたメッセージに基づいて計算された周辺関数 $Z_i(x_i)$ が図 2 (b) となったとする．図 2 (b) では，Max-Sum では評価しないという制約により，b の値が大きくなり，b が x_i^{max} となる．このような場合には，図 2 (a) の Max-Sum による状態 a の選択は間違っていることになり，本来は b を選択するべきである．このように，周辺関数が均衡している場合には，Max-Sum ではエージェントが間違った選択をする可能性が高く，大域的な解の精度が低下することがあるため，周辺関数の均衡が見られる部分では MS-Stable の解の改善効果が得られやすいと考えられる．逆に，周辺関数 $Z_i(x_i)$ が均衡していない場合，すなわち $Z_i(x_i^{max})$ と $Z_i(x_i^{max'})$ の値に大きな差がある場合は，大域的な解の精度にあまり影響がない．なぜなら，Max-Sum で評価されていない制約によって，多少 $Z_i(x_i^{max'})$ が大きい値となるとしても，状態の選択に変化がなく，間違った状態の選択がされないからである．

たとえば不均衡の場合として，Max-Sum のメッセージに基づいて計算された周辺関数が**図 3** (a) となったとする．図 3 (a) では，$x_i^{max} = a$，$x_i^{max'} = b$ で，$Z_i(x_i^{max})$ と $Z_i(x_i^{max'})$ の値に大きな差がある．この場合に Max-Sum で未評価の制約により，$Z_i(x_i^{max'})$ が大きい値となり，図 3 (b) となるとする．図 3 (b) の例では，図 3 (a) の $Z_i(x_i^{max})$ と $Z_i(x_i^{max'})$ の値に大きな差があったために，$Z_i(x_i^{max'})$ がある程度大きい値となっても状態 a を選択するということは変わらない．このように，周辺関数が不均衡であるような場合においては，Max-Sum で選択する状態と MS-Stable で選択する状態に変化が起きる可能性が比較的小さい．そのため，MS-Stable による解の精度の改善効果が得られにくく，計算コストの小さい Max-Sum を適用すべきである．

なお，周辺関数が均衡する場合であっても必ずしも MS-Stable の解の改善効果が得られるわけではない．なぜならば，Max-Sum で未評価の制約によって，必ずしも $Z_i(x_i^{max'})$ が大きくなるとは限らないからである．すなわち，$Z_i(x_i^{max})$ やその他の状態の周辺関数の値が大きくなる場合には，Max-Sum と MS-Stable とで選択する状態に変化が現れないからである．つまり周辺関数の均衡による評価関数の切替えでは，解の改善効果がある場合のみを完全に抽出して MS-Stable を割り当てることはできないが，MS-Stable の解の改善効果が期待できる部分に対して MS-Stable を割り当てることはできる．

周辺関数の値の大小についての具体的な数値は，問題に依存すると考えられる．周辺関数がとりうる値の範囲は式 (1) から (4) により見積もることはできると考えられるが，その差分の大小の指標を解析することは簡単ではないと予想される．本研究では，6 章においてパラメータ δ の値を実験的に評価する．

4.3 動的な評価関数の変更の効果

ここでは，評価関数の変更によって，どのように周辺関数の均衡が解消されるのかを説明する．まず，周辺関数は自身のエージェントの周囲のエージェントからのメッセージ R の和をとることによって計算され，その各値はエージェントが各値を選択した場合の利得の推定値となり，次の式で表される．

$$Z_n(x_n) = \sum_{m \in M(n)} R_{m \to n}(x_n) \approx \max_{\mathbf{x}_m \backslash n} \sum_{m=1}^{M} U_m(\mathbf{x}_m) \tag{16}$$

周辺関数の均衡とは，$Z_n(x_n)$ の最大となる x^{max} と次に最大となる $x^{max'}$ が均衡する場合である．よって周辺関数の均衡を解消するためには，$Z_n(x_n)$ の計算に用いる周囲のエージェントからのメッセージ R の各値に差をつける必要がある．そのメッセージ R は，評価関数 U によって計算されるので，評価関数 U を変更することによって，メッセージ R の値が変動する．すなわち，周辺関数の均衡が見られるエージェントでは評価関数 U を MS-Stable に，均

衡が見られないエージェントでは評価関数 U を Max-Sum を用いることによって，周辺関数の均衡が解消されることが期待でき，計算コストを削減しつつ解の精度の向上が期待できる．

しかし，前述のとおり MS-Stable を用いる計算コストは Max-Sum に比べてきわめて大きいため，周辺関数の均衡が見られるエージェントすべてにおいて，つねに MS-Stable を用いることは効率的ではない．そこで，できる限り少ない計算コストで，効果的に MS-Stable を用いるために，時系列に周辺関数の均衡を検出し，動的に MS-Stable を割り当てる工夫を施す．動的に MS-Stable を割り当てることによって，つねに MS-Stable が使用されることによる計算コストの増大をある程度軽減できると考えられる．

4.2 節で述べたように，Z-MSS では周辺関数の均衡により，MS-Stable が有効である状況とそうでない状況とを判断する．さらに，時系列における評価関数の変更においても，周辺関数に基づいて判断を行う．すなわち，時系列において周辺関数の均衡が見られる場合にのみ，評価関数に MS-Stable を用いる．つねに MS-Stable を用いる手法では，つねに Max-Sum で未評価の制約も評価することで解の精度の点では有利となる．しかし，MS-Stable の計算コストが高いことに加え，4.2 節で述べた周辺関数の均衡が見られない比較的 MS-Stable の効果が小さくなった状況においても MS-Stable を使い続けるため，計算コストの点では無駄が多いと考えられる．このように，MS-Stable の使用により周辺関数の均衡が見られなくなったエージェントにおいては，MS-Stable で計算されたメッセージ R が自身を含む周囲のエージェントに伝搬され各々の周辺関数に影響を与えているので，評価関数を MS-Stable から一時的に Max-Sum に戻しても，解の精度は急には低下しないと考えられる．しかし，Max-Sum に基づく解法では，メッセージ R は評価関数 U と現在に届いている最新のメッセージ Q によって計算されるので，長い期間 Max-Sum に変更したままにしておくと再び周辺関数の均衡が見られ，解の精度が低下する可能性がある．まとめると，つねに MS-Stable を使うことは計算コストの無駄が多いと考えられるが，MS-Stable を長期間使わない場合にも解の精度に問題がある．そのことから，周辺関数の均衡を基準として動的に評価関数を切り替えて用いる手法は有効であると考えられる．

4.4 Z-MSS の実装

Z-MSS は周辺関数に基づいて評価関数 Max-Sum と MS-Stable を切り替えて用いる．そのため，Max-Sum に基づく解法でのメッセージ R の計算，メッセージ Q の計算，周辺関数の計算の 3 つの処理に加え，Z-MSS では周辺関数の計算の際に，周辺関数の均衡を検出する処理を行う．均衡の検出は式 (15) に従って計算する．周辺関数の均衡を検出し，評価関数を切り替える処理を**図 4** に示す．ここで keepFuncCycle とは，Max-Sum から MS-Stable に切り替わったとき，MS-Stable により計算されたメッセージが周囲に十分伝搬される前に元の Max-Sum に戻ることを防ぐ目的で用いる変数である．KeepFuncCycle には定数 λ が代入され，切り替わった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 MS-Stable により λ 回メッセージの計算が行われるまでは Max-Sum に戻らないようにする．

```
if Z_i(x_i^max) < Z_i(x_i^max') + δ then
  use MS-Stable as U_i(x_i)
  keepFuncCycle ← λ
else if keepFuncCycle ≤ 0 then
  use Max-Sum as U_i(x_i)
else
  keepFuncCycle ← keepFuncCycle −1
end if
```

図 4 周辺関数に基づく評価関数の切替え手法 Z-MSS
Fig. 4 Z-MSS that dynamically changes utility-functions based on marginal-function.

4.5 Z-MSS のパラメータ

Z-MSS では解の精度と計算コストの調整を，評価関数の切り替わりやすさを制御するパラメータ δ と，MS-Stable での計算回数を表すパラメータ λ の 2 つのパラメータを用いて行う．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，その結果として多くのエージェントが評価関数を MS-Stable に切り替えるので，解の精度は向上し，計算コストが大きくなると考えられる．また，λ の値を大きくすると，評価関数を MS-Stable に切り替えたエージェントが Max-Sum に戻るまでに時間を要する．そのため，ある期間において，評価関数を MS-Stable としているエージェントが全体に占める割合が多くなる．これにより解の精度は向上し，計算コストは大きくなる．

基本的には，δ と λ の値は大きくするほど MS-Stable の特性に近づき，高い解の精度と高い計算コストとなる．逆に値を小さくするほど Max-Sum の特性に近づき，低い解の精度と低い計算コストとなる．しかし，高い解の精度と低い計算コストを両立させるためには，これらの値を適切に調整する必要がある．たとえば，MS-Stable に切り替えている時間を表す λ の値を小さくしすぎた場合には，MS-Stable の計算結果が周囲に到達せず，解の精度が低下すると考えられるが，逆に λ の値を大きくしすぎた場合には，周辺関数の均衡が解消された後にも MS-Stable を使用してしまうため無駄が多くなる．また δ に関しては，制約の評価値のスケールやグラフの構造によって周辺関数が大きく変化すると考えられるため，問題に合わせて値を設定することでより効率的に MS-Stable を用いることができる

と考えられる．

4.6 異なる評価関数を用いる影響についての考察

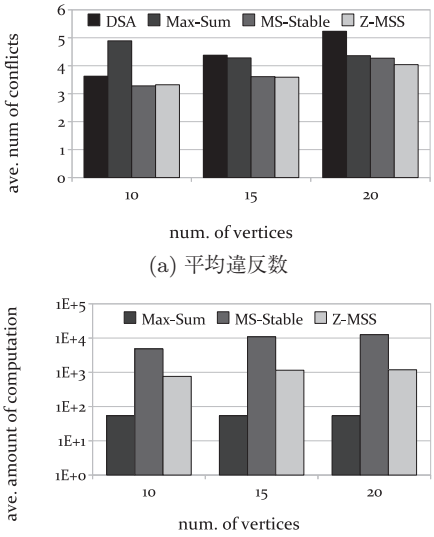
Z-MSS では Max-Sum と MS-Stable の 2 つの評価関数を切り替えて用いるため，各エージェントでは Max-Sum と MS-Stable の評価関数によって計算されたメッセージ R が区別されずに受信される．Max-Sum と MS-Stable では，メッセージ R を計算する際に用いる制約が違うため，メッセージ R の各値のスケールは異なる．すなわち，式 (5) の Max-Sum の評価関数の値と，式 (8) の MS-Stable の評価関数を合計すると，式 (8) に含まれる $f_{i,j}$ の合計値が式 (5) における合計値を上回る場合が起こりうる．このような状況により，式 (5) の評価値が式 (3) の $R_{m\to n}$ の合計値に占める割合は相対的に小さくなる．このような状況が継続すれば，特定の評価関数値が優先されると考えられる．しかし，Max-Sum に基づく解法では，各エージェントで式 (1) および (2) に従ってメッセージ Q を計算する際に正規化を行う．この正規化により，Max-Sum と MS-Stable とで計算されたメッセージ R のスケールの違いが吸収される．その結果，Max-Sum と MS-Stable とで計算された評価値の影響の差異は次第に小さくなると考えられる．もともと Max-Sum では正規化したメッセージを用いて問題を解くため，正確な評価値を必要としておらず，評価値の近似値を用いて準最適解となる状態を選択する．このように，Max-Sum と MS-Stable とを切り替えて用いても，評価値の近似値を用いて準最適解を導くというアルゴリズムの大枠の動作には影響を与えない．

5. 評価

5.1 頂点彩色問題での評価

ここでは，従来手法である Max-Sum と MS-Stable，提案手法の Z-MSS について 3 色の頂点彩色問題を用いて評価する．各頂点数 10，15，20 の問題を各 100 例題ランダム生成し，各例題において 10 回試行した．制約数は変数の数（頂点数）の 3 または 4 倍の数とした．すなわち，制約数を変数の数 n の定数倍とした．このとき，変数の数の増加とともに，2 頂点の組合せの数 ${}_nC_2$ に対する，制約辺の数の比は減少する．この比の減少とともに，グラフ中のクリークに近い部分の数が相対的に減少し，クリークに近い部分的な構造に対して有効と考えられる MS-Stable の効果も，相対的に小さくなると考えられる．以降では，上記の比の概念を単に「制約の密度」と呼ぶ．制約辺は単一連結成分となるようにランダムに生成した．

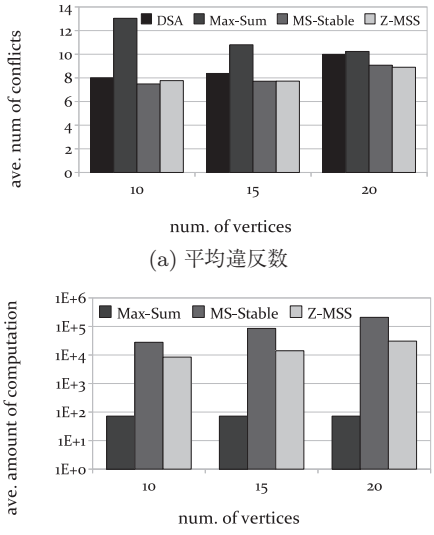
各手法における平均の違反数と計算量を比較した．違反数は制約辺の両端の変数が同じ状態を選択した数を表し，計算量は式 (3) の計算に必要な変数の組合せの数を用いた．計測の便宜上，マルチエージェントシステム全体の動作を，サイクルを単位として同期した．Z-MSS において，均衡を



(a) 平均違反数

(b) 平均計算量

図 5 彩色問題における評価（変数の数の 3 倍の制約数）
Fig. 5 Results of graph coloring ($3n$ constraints).



(a) 平均違反数

(b) 平均計算量

図 6 彩色問題における評価（変数の数の 4 倍の制約数）
Fig. 6 Results of graph coloring ($4n$ constraints).

表すパラメータ $\delta = 0.4$，切替えサイクル数 $\lambda = 3$ とした．これらの数値は，δ を 0.1 から 1.0，λ を 1 から 4 に変化させて実行した結果，最も良いものを選択した．

また，非厳密解法であるである DSA [9] についても比較した．DSA では隣接エージェントが変数値を交換しつつ，基本的には山登り法を実行するが，変数値の変更するか否かは定められた確率に従う．ただし，本実験では明示的な制約違反の数ではなく，Max-Sum と同じ制約の評価値の局所的な合計を用いた[*2]．

平均の違反数を図 **5** (a) と図 **6** (a) に，平均計算量を

[*2] 本実験では，DSA のパラメータを予備実験により設定したが，多くの問題では，確率 1 として山登り法を実行する場合の解の精度が高かった．これは，平均の違反数においては，変数値の変更を抑制するよりも，積極的に局所解に収束させることの効果があるためと考えられる．

図 5 (b) と図 6 (b) に示す．

図 5 (a) を見ると，頂点数 10 の場合では Max-Sum の違反数が高く，MS-Stable と Z-MSS が同程度の違反数となった．この問題は制約の密度が大きく，ほぼすべてのエージェントにおいて，Max-Sum で評価されないが，MS-Stable では評価されるという制約がある．その結果，MS-Stable の解の改善効果が違反数に現れたと考えられる．Z-MSS については，違反数は Max-Sum より 32%改善され，MS-Stable とほぼ同程度となった．計算量は MS-Stable の 16%程度に抑えられた．これは周辺関数による評価関数の切替えによって，解が安定した状態における MS-Stable の割合が減少したからだと考えられる．図 5 (a) の頂点数 15 の場合においても Z-MSS の違反数が MS-Stable とほぼ同程度となったが，制約の密度が頂点数 10 の場合よりも少ないために，Max-Sum からの解の改善の差は小さくなった．頂点数 20 の場合では，図 5 (a) を見ると Z-MSS の違反数が MS-Stable より小さくなった．また図 5 (b) を見ると，計算量すなわち変数値の組合せ数は，MS-Stable の 12,550 に対して，Z-MSS は 1,182 であり，約 90%抑えられた．

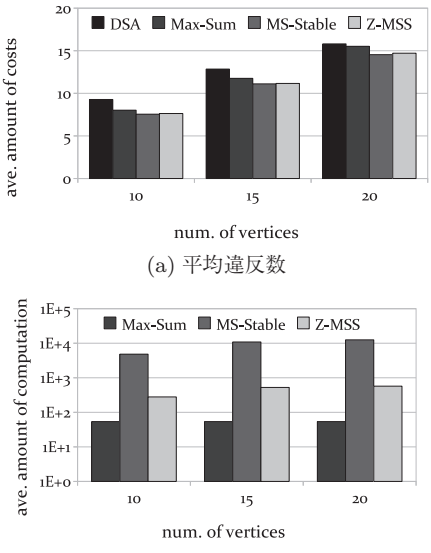
これらの傾向が見られた理由としては，制約数を頂点数の 3 倍と設定しているため，頂点数 15，20 の問題は頂点数 10 の問題と比べて制約の密度が小さいことがあげられる．すなわち Max-Sum で評価されない制約が少ないことが，Max-Sum と MS-Stable の解の精度の差を小さくしたと考えられる．また，MS-Stable ではエージェントの周囲の制約を強く評価値に反映させるため，局所解に陥りやすいと考えられる．そのため，制約の密度が小さい問題では MS-Stable よりも Z-MSS が高い解の精度を示したと考えられる．変数の数に対する制約数が多い図 6 (a) では，図 5 (a) と異なり，頂点数 15 の場合でも Max-Sum と MS-Stable の平均違反数の差が比較的大きい．

これらの結果では DSA は Max-Sum より平均違反数が少ない場合が多いが，頂点数が増加するにつれて，他の手法に対して相対的に平均違反数が高くなっている．これは，DSA のような山登り法では，近傍の変数が比較的多い小規模な問題において，変数値を変更する機会が増加するためであると考えられる．

5.2 評価値に変化がある二項制約の問題での評価

頂点彩色問題は二項制約で構成される問題の 1 つであるが，より一般的な条件における効果を調べるために，ここでは二項制約の評価値を変化させた場合において各手法を比較する．評価用の問題として，ランダムに与えられたすべての制約のコストの合計値を最小化する問題を用いた．

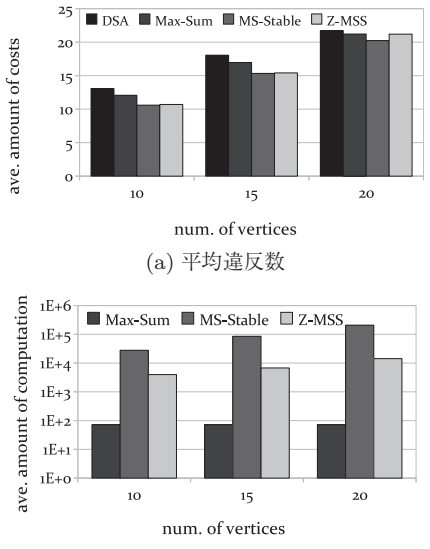
実験では，5.1 節の評価と同じく頂点数 10，15，20 の各場合においてランダムに生成した 100 例題を用いた．ただし，変数 x_i，x_j 間のコスト値を $f_{i,j}$ とし，$f_{i,j}$ には 0.1 から 0.9 のランダムなコスト値を割り当てた．制約に方向はなく $f_{i,j} = f_{j,i}$ とする．また各エージェントの変数の値域の大きさは 3 とした．各手法における平均の評価値の和を**図 7** (a) と**図 8** (a) に，平均の計算量を図 7 (b) と図 8 (b) に示す．



(a) 平均違反数

(b) 平均計算量

図 7 二項制約の最小化問題における評価（変数の数の 3 倍の制約数）
Fig. 7 Results of minimization problems ($3n$ constraints).



(a) 平均違反数

(b) 平均計算量

図 8 二項制約の最小化問題における評価（変数の数の 4 倍の制約数）
Fig. 8 Results of minimization problems ($4n$ constraints).

図 7 (a), 図 8 (a) では，図 5 (a) の彩色問題における結果と異なり，すべての頂点数において Max-Sum と MS-Stable の差が小さくなり，解の精度の改善量が小さくなった．その一方で，Z-MSS と MS-Stable はほぼ同等の評価値となった．また図 7 (b) から，計算量は頂点数 20 の場合に約 95%削減できた．これより，各提案手法が一般的な二項制約の問題においても有効であると考えられる．また頂点数 15，20 において，図 5 (a) の彩色問題では MS-Stable より Z-MSS の方が違反数が少なくなったが，図 7 (a)，図 8 (a)

表 1 グラフの構造と問題を変化させた場合のコストの合計
Table 1 Cost values for different graph structures and cost functions.

problem	stucture	DSA	Max-Sum	Z-MSS $\alpha = 0.001$	Z-MSS $\alpha = 0.01$	Z-MSS $\alpha = 0.1$	Z-MSS $\alpha = 1$	MS-Stable
graph-coloring	complete	12.00	17.21	16.61	15.80	12.14	12.14	12.00
	4clique	5.12	7.85	7.81	5.06	5.06	5.06	5.09
	4cliq.-rand. $m = 60$	7.03	8.69	6.88	6.12	6.02	5.99	6.05
	4cliq.-rand. $m = 80$	11.03	12.50	10.60	9.90	9.84	9.83	10.00
weighted graph-coloring	complete	2.00	3.13	2.99	2.80	2.28	2.04	2.01
	4clique	0.66	1.56	1.44	1.08	0.71	0.62	0.63
	4cliq.-rand. $m = 60$	1.14	1.26	1.03	0.97	0.95	0.96	0.96
	4cliq.-rand. $m = 80$	1.84	1.99	1.69	1.63	1.62	1.66	1.63
minimization-problem	complete	15.35	14.05	13.66	13.48	13.17	12.73	12.51
	4clique	10.16	10.15	9.42	9.25	9.10	8.67	8.65
	4cliq.-rand. $m = 60$	16.12	16.15	15.58	15.18	15.11	15.11	15.02
	4cliq.-rand. $m = 80$	21.90	22.40	21.79	21.18	20.86	20.86	20.45

のランダムにコスト値を設定した場合には，わずかながら MS-Stable の方が少ない結果となった．これはコスト値の変化が，Z-MSS の均衡を表すパラメータ δ に影響を与えたためであると考えられる．この点については，δ をコスト値のスケールに連動させるなどの改良の余地がある．図 7 (b)，図 8 (b) では，図 5 (b) の彩色問題における計算量と同様の傾向が見られた．最小化問題での解の改善幅が小さい点については，最小化問題では彩色問題と異なり，すべての変数値の組に対してコスト値が設定されているため，MS-Stable で導き出される質の良い解と Max-Sum で導き出される解との間に差が生まれにくくなったと考えられる．このことから，MS-Stable および Z-MSS を問題に適用する際には，問題の種類をある程度考慮することが必要であると考えられる．

6. Z-MSS の適切なパラメータ

Z-MSS は周辺関数 $Z(x)$ を指標に評価関数を変更する．その周辺関数 $Z(x)$ の各値は，問題の設定やグラフの構造などによって変化すると考えられる．そのため，より適切な評価関数の切替えを行うためには，グラフの構造や問題設定に合わせた，適切なパラメータを設定することが望ましいと考えられる．そこで Z-MSS のパラメータの変化による挙動と 5.2 節で述べた MS-Stable と Z-MSS が有効な問題を調べるために，グラフの構造と問題を変化させ，Z-MSS のパラメータを増減させて実行し，評価した．

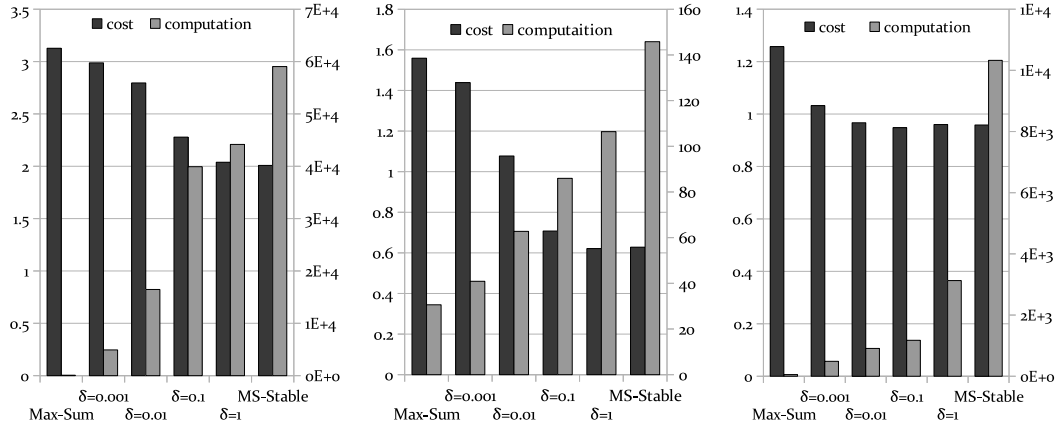
Z-MSS のパラメータは δ と λ の 2 種類である．まず予備実験において，サンプルとして選んだ δ の値を固定し，λ の値を変更した．その結果，いずれの δ の値でも，MS-Stable の効果が得られるためには，λ は数サイクル程度は必要であった．また，さらに λ の値を増加しても，解の精度に顕著な改善は見られなかった．計算量を抑制することを考慮すれば，λ の値は MS-Stable の効果が得られる範囲で小さくすべきであるが，その選択の影響が顕著な範囲は数サイクル前後までと見られたことから，本実験では，5 章の実験と同様に $\lambda = 3$ とした．その一方で，δ の値は問題の評価関数値による変化が比較的大きいと考えられるため，パラメータ δ を変化させて実験を行った．

グラフの構造は complete，4clique，4clique-random の 3 つの構造について調べた．complete は，すべての頂点間に制約辺があるグラフで，最も制約密度が高く，MS-Stable で評価される制約が多くなるグラフである．4clique は，4 頂点からなる完全グラフ複数個を線形に連結したグラフで，解の精度が低下しやすいクリークから構成されるグラフである．4clique-random は 4clique の問題に対してランダムに制約を追加したグラフである．頂点数は complite が 10，4clique と 4clique-random では 20 である．制約数 m は complite が 45，4clique が 34，4clique-random は 60 および 80 とした．これらは，制約数が頂点数の 3 および 4 倍の密な問題を含む．問題の種類としては彩色問題（graph-coloring），重み付き彩色問題（weighted graph-coloring），コスト最小化問題（minimization-problem）の 3 つを用いた．重み付き彩色問題は，制約辺の両端の変数値が同じ場合にのみ，0.1 から 0.3 の違反値を設定したものである．彩色問題とコスト最小化問題については，5 章と同様である．比較手法は Max-Sum，Z-MSS，MS-Stable を用い，Z-MSS についてはパラメータ δ を 0.001 から 1 に変化させた．グラフの構造と問題を変化させた場合の解の精度を**表 1** に，計算量を**表 2** に示す．表内では右側の列の手法ほど，全体における MS-Stable の割合が高くなる．特にパラメータの変化による解の精度の変化が大きかった重み付き彩色問題について，解の精度と計算量をグラフ化したものを**図 9** に示す．

問題による違いの傾向として，表 1 の problem ごとに Max-Sum，MS-Stable，Z-MSS の欄を比較すると，重み付き彩色問題，彩色問題，コスト最小化問題の順に，Max-Sum と MS-Stable および Z-MSS の解の精度の差が大きくなっ

表 2 グラフの構造と問題を変化させた場合の計算量
Table 2 Amount of computation for different graph structures and cost functions.

problem	stucture	Max-Sum	Z-MSS $\alpha = 0.001$	Z-MSS $\alpha = 0.01$	Z-MSS $\alpha = 0.1$	Z-MSS $\alpha = 1$	MS-Stable
graph-coloring	complete	81	4,893	15,448	57,870	57,870	59,049
	4clique	31	42	141	143	143	146
	4cliq.-rand. $m = 60$	54	525	1,009	1,239	1,495	10,326
	4cliq.-rand. $m = 80$	72	8,482	17,739	20,664	23,878	159,477
weighted graph-coloring	complete	81	4,916	16,462	39,920	44,177	59,049
	4clique	31	41	63	86	106	146
	4cliq.-rand. $m = 60$	54	487	909	1,175	3,127	10,326
	4cliq.-rand. $m = 80$	72	8,050	15,833	18,276	43,292	159,477
minimization-problem	complete	81	2,440	5,648	14,269	21,982	59,049
	4clique	31	33	40	49	70	146
	4cliq.-rand. $m = 60$	54	159	370	521	1,084	10,326
	4cliq.-rand. $m = 80$	72	1,931	6,020	9,440	16,224	159,477



(a) complete の場合 (b) 4clique の場合 (c) 4clique-random の場合

図 9 重み付き彩色問題における解の精度と計算量
Fig. 9 Solution quality and amount of computation in weighted graph-coloring.

た．この理由として，コスト最小化問題においては，前述のとおり解の質に違いが出にくいことが考えられる．重み付き彩色問題については，両端が同じ変数値となる場合でも各変数値により違反値が異なるため，優先されるべき変数値が存在する．そのため問題が難しくなり，解の質に差ができやすくなったと考えられる．また，Z-MSS のパラメータについては，彩色問題，重み付き彩色問題，コスト最小化問題の順に大きい値で MS-Stable に近い解の精度を示し，彩色問題では $\delta = 0.01$〜0.1 付近，重み付き彩色問題では $\delta = 0.1$〜1 付近，コスト最小化問題では $\delta = 1$ 付近となった．

グラフの構造による違いとして，表 1 の structure ごとに比較すると，完全グラフの場合は，δ の値が増えるにつれ解の改善が見られた．また表 2 で完全グラフについて比較すると，解の改善にともない計算量も大幅に増加した．重み付き彩色問題では図 9 (a) のように，Z-MSS は $\delta = 1$ のとき MS-Stable に近い解の精度となり，Max-Sum から 35%解の精度が改善されたが，計算量の削減は MS-Stable から 25%にとどまった．これは完全グラフは制約の密度が高いグラフであることから，すべてのエージェントで MS-Stable の効果が期待できるため，計算量の削減が難しかったと考えられる．次に表 1 の 4clique について Max-Sum から MS-Stable までを比較すると，他の問題と同様に，δ の値が増加するほど解が改善されたが，隣接エージェント数が小さいため，表 2 の 4clique では Max-Sum と MS-Stable の計算量の差が小さい．また，重み付き彩色問題では図 9 (b) のように，Z-MSS は $\delta = 1$ のとき MS-Stable に近い解の精度となり，計算量は MS-Stale に比べ 27%改善されたが，Max-Sum と MS-Stable の計算量の差が小さいために，計算量の削減効果は小さいといえる．4clique のような問題では，あらかじめ MS-Stable を用いるか，δ の値を大きく設定することが有利となると考えられる．最後に表 1 の 4clique-random について，Max-Sum から MS-Stable までを比較すると，他の問題と同様に δ の値が増えるにつれ解が改善されたが，MS-Stable の計算量が大幅に増加した．これはランダムに制約が追加されたことにより，一部の隣接エージェント数が大きくなったからだと考えられる．重み付き彩色問題の場合では，図 9 (c) のように，

Z-MSS は $\delta = 0.1$ のとき MS-Stable に近い解の精度を示した．Max-Sum から解の精度は 25%改善され，計算量は MS-Stable から 88%削減できた．Z-MSS の効果が高い理由として，クリークを多く含むような，ある程度 MS-Stable の効果が見られる問題であることに加えて，ランダムに制約が追加されることによって，各エージェントでの探索の優先度に差ができ，周辺関数からの推定によって効果的に評価関数の切替えが行われたためであると考えられる．

また，4clique-random では，制約数が多い 80 制約の問題の方が，Max-Sum と MS-Stable のコスト値の差が比較的顕著であった．

7. おわりに

本論文では，分散制約最適化問題の解法 Max-Sum アルゴリズムにおける，2 つの評価関数の解の精度と計算量のトレードオフを利用するために，周辺関数に基づいて評価関数を動的に変更する手法 Z-MSS を提案した．Z-MSS は比較的簡単なアイデアに基づく手法であるが，評価関数の詳細さの動的な切替えおよび，その指標としてエージェントの局所的な知識の不確実さに関係する量を用いる点は，マルチエージェントシステムのための最適化手法として一定の意義があると考える．Z-MSS では，Max-Sum で問題であった複雑な制約網における解の精度の改善と，MS-Stable で問題であった計算量の大幅な増加を抑制した．さらに，グラフの構造や問題による変化が既存手法，提案手法に与える影響について調べた．その結果，Z-MSS がパラメータにより解の精度と計算量をある程度調整可能であること，Z-MSS が有効である問題とその場合におけるパラメータを確認できた．コスト最小化問題では，MS-Stable の効果が小さいこともあり，解の改善効果が小さい場合もあったが，特に彩色問題においては，計算量を削減しつつ，解を改善することができた．このことから，彩色問題に定式化できるようなクラスの分散制約最適化問題においては，Z-MSS は一定の有用性があると考える．

本研究では，動的に評価関数を切り替えつつ，解の精度の改善と維持を目指す提案手法の観点から，ある期間における平均的な解の精度に注目した．収束性に関する詳細な検討と評価は，停止検出手法の導入とともに，今後の課題の 1 つにあげられる．Z-MSS のパラメータを問題の特徴に応じて動的に決定する方法や Any-time 性を活用した動的な問題の変化への対応も今後の課題である．

参考文献

[1] Kim, Y., Krainin, M. and Lesser, V.: Application of Max-Sum Algorithm to Radar Coordination and Scheduling, *12th Int. Workshop on Distributed Constraint Reasoning at the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, pp.5–19 (2010).

[2] Ramchurn, S.D., Farinelli, A., Macarthur, K.S. and Jennings, N.R.: Decentralized Coordination in RoboCup Rescue, *The Computer Journal*, Vol.53, No.9, pp.1447–1461 (2010).

[3] Macarthur, K.S., Farinelli, A., Ramchurn, S.D. and Jennings, N.R.: Efficient, Superstabilizing Decentralised Optimisation for Dynamic Task Allocation Environments, *3rd Int. Workshop on Optimisation in Multi-agent Systems at the 9th Joint Conf. on Autonomous Agents and Multiagent Systems*, pp.25–32 (2010).

[4] Yokoo, M. and Hirayama, K.: Algorithms for Distributed Constraint Satisfaction: A Review, *Autonomous Agents and Multi-Agent Systems*, Vol.3, No.2, pp.185–207 (2000).

[5] Modi, P.J., 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).

[6] Petcu, A. and Faltings, B.: DPOP: A scalable method for multiagent constraint optimization, *19th Int. Joint Conf. on Artificial Intelligence*, pp.266–271 (2005).

[7] Farinelli, A., Rogers, A., Petcu, A. and Jennings, N.R.: Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm, *7th International Joint Conf. on Autonomous Agents and Multiagent Systems*, pp.639–646 (2008).

[8] Rogers, A., Farinelli, A., Stranders, R. and Jennings, N.R.: Bounded Approximate Decentralised Coordination via the Max-Sum Algorithm, *Artificial Intelligence*, Vol.175, No.2, pp.730–759 (2011).

[9] Zhang, W., Wang, O. and Wittenburg, L.: Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance, *Workshop on Probabilistic Approaches in Search at AAAI Conf.*, pp.53–59 (2002).

[10] Stranders, R., Farinelli, A., Rogers, A. and Jennings, N.R.: Decentralised coordination of continuously valued control parameters using the max-sum algorithm, *8th International Conference on Autonomous Agents and Multiagent Systems*, Vol.1, pp.601–608 (2009).

[11] Delle Fave, F.M., Stranders, R., Rogers, A. and Jennings, N.R.: Bounded decentralised coordination over multiple objectives, *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol.1, pp.371–378 (2011).

[12] Delle Fave, F.M., Xu, Z., Rogers, A. and Jennings, N.R.: Decentralised Coordination of Unmanned Aerial Vehicles for Target Search using the Max-Sum Algorithm, *Workshop on Agents in Real Time and Environment at the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, pp.35–44 (2010).

川東 勇輝

2010 年名古屋工業大学工学部情報工学科卒業．2012 年同大学大学院工学研究科創成シミュレーション工学専攻修士課程修了．マルチエージェントシステム，分散システム等に興味を持つ．

松井 俊浩 （正会員）

1995 年名古屋工業大学電気情報工学科卒業．1999 年同大学大学院博士前期課程修了．2006 年同博士後期課程修了．同年名古屋工業大学情報基盤センター助手．2007 年同助教．2011 年同准教授，現在に至る．分散協調処理，マルチエージェントシステム，分散制約最適化問題に関する研究に従事．博士（工学）．電子情報通信学会，人工知能学会各会員．

松尾 啓志 （正会員）

1983 年名古屋工業大学情報工学科卒業．1989 年同大学大学院博士課程修了．同年名古屋工業大学電気情報工学科助手．講師，助教授を経て，2003 年同大学院教授．2006 年情報基盤センターセンター長（併任）．2011 年付属図書館長，現在に至る．分散システムに関する研究に従事．工学博士．電子情報処理学会，人工知能学会，IEEE 各会員．