JAWS2011

JAWS2011

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による効果

Effects of dynamically changing utility-functions in the Max-Sum algorithm

川東 勇輝 Yuuki Kawahigashi
名古屋工業大学 Nagoya Institute of Technology
yuuki@matlab.nitech.ac.jp

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

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

keywords: DCOPs, Max-Sum, Cooperation, Optimization, Search

Summary

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 an approximate method for the DCOPs. The Max-Sum algorithm generates relatively better approximate solutions when it is applied to less cyclic constraint graphs. However, in the case of over-constrained graphs, 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 computation of each agent. To employ this trade off, we propose methods that suitably switches the both type of utility-functions while the algorithm is running. In this paper, we study about the characteristics of the Max-Sum and MS-Stable. Then, effects of the methods that switches the utility-functions are considered. Those methods are applied to graph coloring problem and random weighted binary constraints problem in the experiments. The result shows that the method using both utility-functions keeps relatively high quality of the solution with a lower computation up to 17% of the MS-Stable.

1. は じ め に

近年，低消費電力の無線デバイスやセンサーネットワークを用いた自然現象のモニタリング [Kim 10] や，自律的に動作するロボットを用いた災害救助 [Ramchurn 09][Macarthur 10] 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 [Modi 03][Petcu 05][Farinelli 08][Farinelli 09]．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[Modi 03] や DPOP[Petcu 05] 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[Zhang 02] や Max-Sum[Farinelli 08] 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

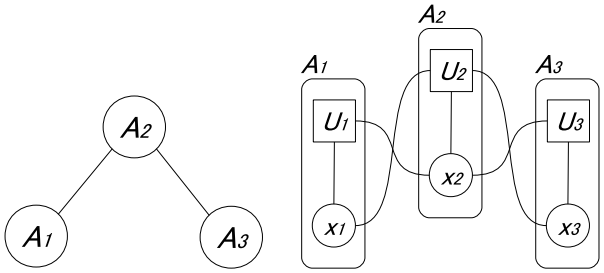
は，制約網において隣接する変数間の制約も評価に含める評価関数 MS-Stable によって改善することができるが，隣接変数間の制約を評価に含めることによって，各エージェントが評価する部分問題の規模は増大する [Farinelli 08]．これらの解の精度と問題の規模のトレードオフを調整する手法として，Z-MSS[Kawahigashi 10] が提案されている．Z-MSS は周辺関数の値を指標として，解の不安定な部分を重点的に探索する手法であり，一定の効果が得られることは示されていた．しかし，具体的にどのような特徴を持つ問題に効果があるか，またアルゴリズムの実行中に評価関数を切り替えた場合の影響などが詳細には示されていなかった．そこで本研究では，はじめに Max-Sum，MS-Stable が効果的である問題がどの程度存在するのかを統計的に解析行う．また MS-Stable から Max-Sum へと切り替えた際に解の変化を調査する．それらの解析結果から，問題の特徴により評価関数を切り替える予備的な手法 Global,Local の 2 手法を提案する．Global,Local,Z-MSS を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において評価・比較を行い，考察する．

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 Z-MSS の二項制約の最適化問題への適用方法とグラフの頂点彩色問題への適用方法について述べる．4. 節では Max-Sum と MS-Stable の解析と，評価関数を切り替えた際の影響について述べる．5. 節では，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う．6. 節では，4. 節と 5. 節の結果を用いて，Z-MSS の効果について考察する．7. 節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．



(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

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 [Farinelli 08]．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．

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．

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

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．

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

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，$\max_{\mathbf{x}_m\backslash n}$ 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，$Z_n(x_n) \approx \max_{\mathbf{x}_m\backslash n} \sum_{m=1}^{M} U_m(\mathbf{x}_m)$ を表す．

3·2 二項制約の問題への適用

2. 節で述べたように，分散制約最適化問題における二項制約の問題は，変数 x_i, x_j における制約は $c_{i,j}$ で表され，$c_{i,j}$ に対応する評価関数 $f_{i,j}$ により，変数値の割り当て $\{(x_i,d)(x_j,d')\}$ の評価値が定義される．そして全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問題である．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．

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

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．

3·3 頂点彩色問題への適用

二項制約で表される問題のひとつに，グラフの頂点彩色問題がある．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．

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

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) 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 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．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]．

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

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

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

である. ただし式 (11) は, 計算を省いた Max-Sum を使用した時の組み合わせ数である. 計算を省いた Max-Sum とは，式 (3) に式 (5) を代入し変形すると次式となる．

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

ここで $\max_{\mathbf{x}_m \setminus n}$ を計算する上で，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

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

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み合わ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み合わせ数は最大の場合

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

まで増加する．

*1 式 (8)(9) 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 [Farinelli 08] に基づいている.

3·5 Z-MSS

Max-Sum と MS-Stable を切り替えて使う手法として，Z-MSS[Kawahigashi 10] が提案されている．Z-MSS は周辺関数 $Z(x)$ の値を指標として，アルゴリズムの実行中に Max-Sum と MS-Stable の評価関数を切り替えて使う手法である．この手法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．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してしまうことによって，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．

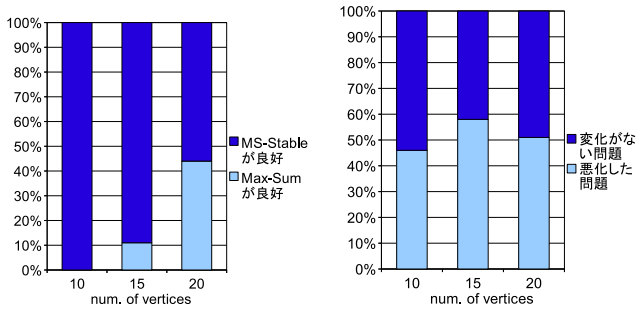
4. 評価関数の切り替えによる効果

4·1 問題の特徴による評価関数への影響

前述のとおり，拡張された評価関数 MS-Stable は，複雑な制約網での解の精度を高めることができる．しかし，複雑でない制約網の問題に適用した場合には Max-Sum よりも大幅に計算量が増大するが，解の精度も同程度にしかならないと予想される．そこで，どの程度の問題に

Algorithm 1 周辺関数に基づく評価関数の切り替え

if $Z(x_{max}) < Z(x_{max'}) + \delta$ **then**
  use MS-Stable as $U_m(x_m)$
  keepFuncCycle $\leftarrow \lambda$
else if keepFuncCycle ≤ 0 **then**
  use Max-Sum as $U_m(x_m)$
else
  keepFuncCycle $\leftarrow$ keepFuncCycle -1
end if



(a) Max-Sum が良好となる問題と (b) 評価関数の切り替えにより解が MS-Stable が良好となる問題の割合 変化した問題の割合

図 2 100 例題における問題の傾向と評価関数を切り替えた場合の解の変化

おいて MS-Stable が有効であるかを確かめるための予備実験 1 を行った．予備実験 1 では，3 色の頂点彩色問題を用いて，Max-Sum と MS-Stable の平均の違反数を比較した．変数の数は 10,15,20 の 3 つの場合を用いた．制約数は変数の数の 3 倍とした．各問題を 100 例用意し，各例において 10 回試行した平均をとった．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．100 例題の内，Max-Sum が良好であった問題と MS-Stable が良好であった問題の割り合いは図 2(a) のようになった．頂点数 10 の場合，全ての問題において Max-Sum よりも MS-Stable が良好な解を示した．頂点数 15 の場合では 11 例の問題で Max-Sum が MS-Stable より良好な解を示し，頂点数 20 の場合では 44 例の問題で Max-Sum が良好な解を示した．このような結果となった理由として，制約の密度が挙げられる．予備実験 1 では，頂点数の 3 倍の制約をランダムに設定しているため，頂点数が多くなるにつれ，制約の引き得る組み合わせに対して，制約の密度が小さくなる．そのことから，各変数が持つ制約の数に偏りが生じる．すなわち，頂点数 20 の例題において MS-Stable の効果が小さかったことから，MS-Stable が良好な解を示す問題は，制約の密度や完全グラフをどの程度含むかに左右されると考えられる．またこの結果から，MS-Stable を実行する必要のない問題においてできるだけ MS-Stable を実行しないようにすることで，不必要な計算を抑えることができる余地がある．

4·2 評価関数の切り替えによる効果

実行途中に評価関数を切り替えた際には，以前に使用していた評価関数で計算されたメッセージが，factor グラフのネットワーク上に伝搬され，他のエージェントにおいてのメッセージ計算に用いられる．そのことから，以前に使用していた評価関数の影響が，評価関数を切り替えた後にも影響すると考えられる．そこで，以前に使用していた評価関数の影響がどの程度あるのか，特に MS-Stable で計算され，維持されていた解が，Max-Sum に切り替えたあと，どの程度その解を維持できるのかに注目した予備実験 2 を行った．予備実験 2 では，評価関数を MS-Stable から Max-Sum に切り替えたあとに解が悪化する例題を調べた．実験諸元は 4.1 節の予備実験 1 と同様である．また評価関数の切り替えは，10 サイクル目で行った．100 例題中悪化した問題と変化がなかった問題の割り合いを図 2(b) に示す．頂点数 10 の場合では 54 例題，頂点数 15 では 42 例題，頂点数 20 では 49 例題において，MS-Stable から Max-Sum に切り替えても解が悪化しなかった．この結果から，約半数の例題においては一度 MS-Stable で解が収束したのち，そのまま MS-Stable を使い続けることは効果がないことがわかる．すなわち，このような問題においては，MS-Stable で解が収束したのち，適宜 Max-Sum に切り替えることで不必要な計算を抑えることができる余地がある．

4·3 大域的な違反数に基づく手法

4. 節で述べたように MS-Stable は，ある一部の問題に対しては Max-Sum よりも計算量が多いにも関わらず，解の精度が悪化してしまう場合がある．そこで本節では，このような MS-Stable の効果が期待できない問題において，MS-Stable が選択され続けないようにした場合にどの程度の効果得られるかを評価するために，大域的な評価値の情報を容易に得られるという前提のもとで，MS-Stable で大域的な評価値が改善しない場合には Max-Sum に切り替え，Max-Sum で解が悪化するならば MS-Stable に切り替える実験的な手法 Global を示す．具体的な切り替え方法を Algorithm2 に示す．ここで $GlobalValuation$ は全ての制約の評価値の和である．また $GlobalValuation'$ は前サイクルにおける評価値の和である．前サイクルにおける評価値から，現時点で取得した評価値が改善されていないなら評価関数を切り替え，λ サイクル間使用する．λ サイクル後，また同様の判定を行い，評価関数を切り替えるかどうかを判定する．大域的な評価値を用いて評価関数を切り替えれば，4.1 節で指摘した MS-Stable を使っても効果がない問題において MS-Stable を使い続けることを防ぐことができると考えられる．また 4.2 節で指摘した MS-Stable から Max-Sum に切り替えても解が悪化しない問題についても，改善することができると考えられる．ただし，大域的な評価値を得ることは実際には比較的大きな通信遅延を必要とする．このため，Max-Sum

Algorithm 2 大域的な違反数に基づく評価関数の切り替え

if use MS-Stable as $U_m(x_m)$ **then**
  if $GlobalValuation' < GlobalValuation$ **then**
    use MS-Stable as $U_m(x_m)$ for λ cycles
  else
    use Max-Sum as $U_m(x_m)$ for λ cycles
  end if
else
  if $GlobalValuation' < GlobalValuation$ **then**
    use Max-Sum as $U_m(x_m)$ for λ cycles
  else
    use MS-Stable as $U_m(x_m)$ for λ cycles
  end if
end if

Algorithm 3 部分的な違反数に基づく評価関数の切り替え

if use MS-Stable as $U_m(x_m)$ **then**
  if $LocalValuation'_m < LocalValuation_m$ **then**
    use MS-Stable as $U_m(x_m)$ for λ cycles
  else
    use Max-Sum as $U_m(x_m)$ for λ cycles
  end if
else
  if $LocalValuation'_m < LocalValuation_m$ **then**
    use Max-Sum as $U_m(x_m)$ for λ cycles
  else
    use MS-Stable as $U_m(x_m)$ for λ cycles
  end if
end if

のような局所的な情報の伝搬に基づく分散アルゴリズムの改良手法としては，単に大域的な評価値を集計することは本来，整合しない．ここでは，Max-Sum の評価関数を変更する指標として，大域的な評価値が利用できる場合にその効果を評価するために用いた．次節に，局所的な評価値を用いて切り替える手法を示す．

4·4 限られた範囲の違反数に基づく手法

大域的な違反数を用いて評価関数を切り替える手法は，実際にはその時点での正確な評価値を取得することは難しい．そこでエージェントに隣接するエージェントとの制約と，隣接するエージェント間の制約のみを評価し，切り替える手法 Local を示す．この場合，各エージェントで評価値を計算する際に用いるのは隣接するエージェントの変数値のみとなるため，比較的容易に計算可能である．具体的な切り替え方法は，algorithm3 に示す．ここで $LocalValuation_m$ はエージェント A_m に隣接するエージェントとの制約と，隣接するエージェント間との制約の評価値の和である．$LocalValuation'_m$ は前サイクルにおける $LocalValuation_m$ である．この切り替え

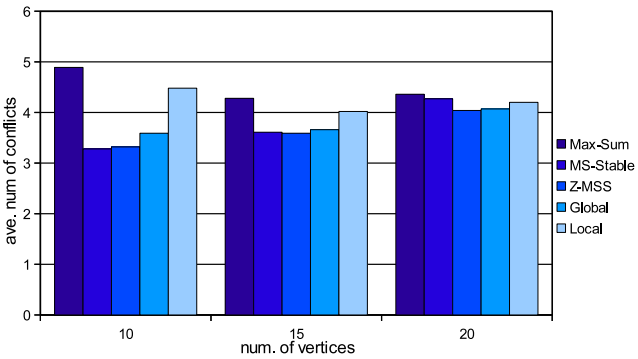


図 3 彩色問題における各手法の平均違反数

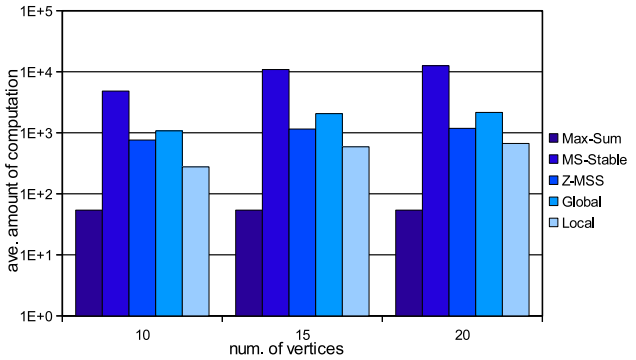


図 4 彩色における各種法の平均計算量

手法の場合には，大域的な違反数を用いる手法と比べると，部分的な解の収束や悪化により評価関数を切り替えるため，切り替えのレスポンスは良くなる．一方で，全体ではまだ収束していないにも関わらず評価関数を切り替える可能性が大きくなるため，解が安定しない可能性があると考えられる．

5. 評　　価

5·1 頂点彩色問題での評価

ここでは，大域的な違反数に基づく手法と部分的な違反数に基づく手法の 2 つを評価する．評価諸元は，4.1 節で述べたものと同様に 3 色の頂点彩色問題を用いた．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．各手法における平均の違反数と計算量を比較した．Z-MSS において，均衡を表す閾値 $\delta = 0.4$, 切り替えサイクル数 $\lambda = 3$ 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．Global と Local においては，切り替えサイクル数 $\lambda = 2$ とした．平均違反数は図 3 となり，平均計算量は図 4 となった．平均計算量は対数表記である．

頂点数 10 の場合では，MS-Stable,Z-MSS,Global,Local, Max-Sum の順に違反数が小さくなった．このような制約の密度が大きい問題の場合では，図 2(a) で示した通

り，ほぼ全ての問題において MS-Stable が効果的な問題であるので，その効果が違反数に現れていると考えられる．Global については，ある程度解が良くなるまで MS-Stable が用いられ，その後一部 Max-Sum と MS-Stable が切り替えて使われるため，違反数では MS-Stable を使い続けるよりも悪化したが，計算量においては MS-Stable の 1/4 程度に抑えられている．これは，図 2(b) で示した MS-Stable から Max-Sum に切り替えても違反数が増大しないような問題に効果があったと考えられる．Local については，違反数が Max-Sum より少し低い程度になった．Local では隣接するエージェントの状態と評価値から，評価関数の切り替えを判断しているが，あまり効果が出なかった．Z-MSS は，違反数が MS-Stable とほぼ同程度で，計算量は MS-Stable の 16%程度に抑えられた．周辺関数の評価値の差がない部分を優先的に探索する方法がうまく働いていると考えられる．

頂点数 15 の場合では，Global,Z-MSS の違反数が MS-Stable とほぼ同程度となった．これは MS-Stable から Max-Sum に切り替えても違反数が増大しないような問題に効果があったことに加え，元々MS-Stable の効果がない問題での MS-Stable の使用比率が減り，効果的に評価関数が選択されたと考えられる．

頂点数 20 の場合では，Global,Local,Z-MSS の違反数が MS-Stable より小さくなった．また計算量は MS-Stable に比べ，Z-MSS が 10%,Global が 17%, Local が 5%ほどに抑えられている．頂点数 20 の問題では，制約の密度がある程度小さく，MS-Stable を使うことで弊害が出る問題の割り合いが多いため，これらの手法が効果的に働いていると考えられる．

5·2 制約の重みを変化させた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ひとつであるが，ここでは二項制約の重みを変化させた場合において各手法を比較する．問題は，全ての制約の評価値の合計値を最小化する問題とする．実験諸元は，4.1 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 間の制約の重みを $f_{i,j}$ とし，$f_{i,j}$ には整数 1 から 9 のランダムな重みを割り当てた．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．その場合の各手法おける平均の評価値の和を図 5 に，平均の計算量を図 6 に示す． ここで opt は，各例題における最適解の平均である．図 5 では彩色問題での評価図 3 と同様の傾向が見られ，Z-MSS および Global,Local の 3 つの手法においてほぼ MS-Stable と同じ評価値となり，opt に近い値を示した．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，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，ランダムに重みを設定した場合には，わずかながら MS-Stable が良い結果となった．これは制約の重みの変化が，Z-MSS の均衡を表すパラメーター 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 を制約の重みに連動させるなどの改良の余地がある．図 6 では，彩色問題での計算量図 4 と同様の傾向が見られた．

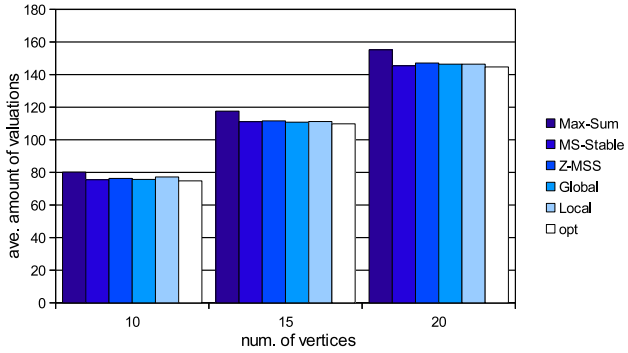


図 5 制約の重みを変えた場合の平均の評価値の和

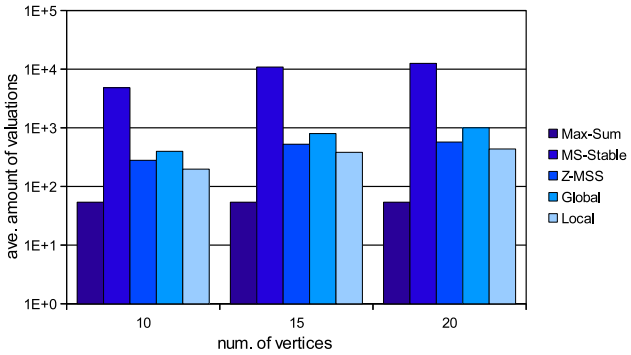


図 6 制約の重みを変えた場合の平均計算量

6. Z-MSS との比較と考察

5. 節の結果から，制約の密度がある程度小さい問題においては，Global,Local,Z-MSS が効果的であることがわかった．そこで本節では，Max-Sum,MS-Stable の評価関数の切り替えによる効果の限界について議論する．4.1 節の図 2(a) にあるように，制約の密度により Max-Sum が有効である問題と MS-Stable が有効である問題が存在する．そこで，あらかじめそれらの問題が判断できると仮定した場合，それぞれに適した評価関数を設定し実行することで，最適な評価関数を選んだ際の理論値が計算できる．その理論値の平均違反数は図 7 の左棒グラフに，平均計算量は図 8 の左棒グラフに示す．また，Z-MSS と Max-Sum において，図 2(a) と同様の比較をしたところ，頂点数 15,20 の場合にそれぞれ 8,21 例題で Max-Sum の違反数が小さくなった．そこで Z-MSS と Max-Sum で理論値を計算した結果を，図 7 の中央グラフに，平均計算量は図 8 の中央のグラフに示す．また参考として Z-MSS の結果を右の棒グラフに示す．

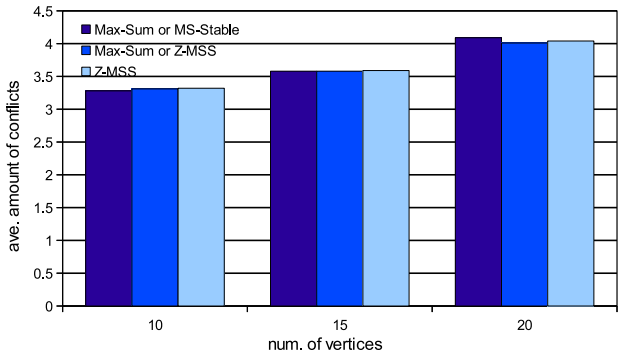


図 7 理想的な切り替えを行った際の平均違反数

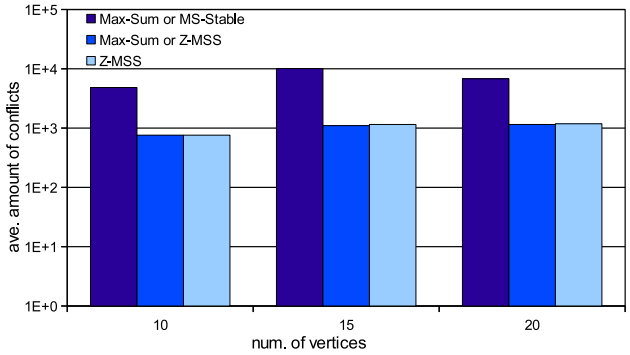


図 8 理想的な切り替えを行った際の平均計算量

静的に解析してあらかじめ最適に Max-Sum と MS-Stable を設定したもの，Max-Sum と Z-MSS を設定したもの，および Z-MSS でほぼ同程度の違反数となった．また計算量においては，最適に Max-Sum と Z-MSS を設定したもの，および Z-MSS で大幅に小さくなった．また違反数と計算量において Z-MSS とその理論値はほぼ変わらない値を示した．この結果から，Z-MSS において周辺関数が均衡している部分を優先的に調べることによって，MS-Stable が有効な問題と Max-Sum が有効な問題において，適切に評価関数が切り替えられていると考えられる．さらに，静的に設定した Max-Sum と MS-Stable の理論値よりも計算量が抑えられている点から，図 2(b) で示した MS-Stable から Max-Sum に切り替えても解が悪化しない問題においても，適切に評価関数が切り替えられ，計算量が抑えられていると考えられる．

7. お わ り に

本研究では，分散制約最適化問題の解法 Max-Sum および MS-Stable の解析を行い，それぞれの解法において問題による得手不得手があることを示した．また解析結果から，実験的な手法 Global,Local の 2 つの手法を提案し，従来手法 Max-Sum,MS-Stable,Z-MSS と比較を行った．その結果，制約の密度がある程度小さい問題において，Z-MSS,Global,Local の手法が MS-Stable より違反数を抑えることができた．特に，Z-MSS では平均違反数を維持しつつ，平均計算量を大幅に抑えることができた．また平均計算量が大幅に抑えることができた理由について，解析結果と比較し，問題による得手不得手をある程度考慮した評価関数の切り替えがされていることを考察した．しかしながら，制約の密度の小さい問題では，少ないながらも Z-MSS より Max-Sum が良くなる場合があるため，そのような場合にはできるだけ MS-Stable が選択されないようにする工夫の余地がある．

頂点彩色問題における実験において，頂点数 15,20 では Max-Sum と提案手法の違反数の差が小さかったが，問題のグラフの構造にサイクルを多く含めることによって，違反数の差は大きくなると考えられる．このような問題を含む，より広範囲な問題についての評価と，統計的な検定は今後の課題である．また，動的な問題への適用が挙げられる．元々Max-Sum は実行時に現在の最適な変数値を推定し，その値をとり続けるためある程度動的な問題に対応できる．しかしながら，問題の制約網が変化した時に，一時的に解が悪化することが予想される．そのような場合には，本研究で提案した評価関数の動的な切り替えを応用することで，少ない計算量において解を速やかに収束させることができると考えられる．

◇ 参 考 文 献 ◇

[Farinelli 08] Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R.: 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)

[Farinelli 09] Farinelli, A., Rogers, A., and Jennings, N.: Bounded Approximate Decentralised Coordination using the Max-Sum Algorithm, in *IJCAI-09 Workshop on Distributed Constraint Reasoning*, pp. 46–59 (2009)

[Kawahigashi 10] Kawahigashi, Y., Matsui, T., and Matsuo, H.: 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, 合同エージェントワークショップ&シンポジウム JAWS2010 (2010)

[Kim 10] Kim, Y., Krainin, M., and Lesser, V.: 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)

[Macarthur 10] Macarthur, K. S., Farinelli, A., Ramchurn, S. D., and Jennings, N. R.: Efficient, Superstabilizing Decentralised Optimisation for Dynamic Task Allocation Environments, in *3rd Int. Workshop on Optimisation in Multiagent Systems at the 9th Joint Conf. on Autonomous and Multiagent Systems*, pp. 25–21 (2010)

[Modi 03] Modi, P. J., Shen, W., Tambe, M., and Yokoo, M.: An Asynchronous Complete Method for Distributed Constraint Optimization, in *Proc. of 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, pp. 161–168 (2003)

[Petcu 05] Petcu, A. and Faltings, B.: DPOP: A scalable method for multiagent constraint optimization, *Proc. of 19th Int. Joint Conf. on Artificial Intelligence*, pp. 266–271 (2005)

[Ramchurn 09] Ramchurn, S., Farinelli, A., Macarthur, K., Polukarov, M., and Jennings, N.: Decentralised Coordination in RoboCup Rescue (2009)

[Zhang 02] Zhang, W., Wang, O., and Wittenburg, L.: Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance, in *Workshop on Probabilistic Approaches in Search AAAI2002*, pp. 53–59 (2002)