平成23年度 修士論文

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の提案

指導教員
松尾 啓志 教授
松井 俊浩 准教授

名古屋工業大学大学院工学研究科
創成シミュレーション工学専攻
平成22年度入学 22413528番

川東 勇輝

目 次

第1章

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 [1] や，自律的に動作す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]．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 [6].

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，解の精度と計算量とを調整可能である．そのパラメータは，厳密に設定しなくても，ある程度の Z-MSS の効果が得ることができるが，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二項制約の最適化問題と頂点彩色問題において評価する．さらに提案手法については，複数のグラフの構造や問題の種類を用いて実験し，適切なパラメータについて調査する．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，MS-Stable 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 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，Max-Sum，MS-Stable の解析と評価関数の切り替えの効果について議論する．6 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．7 章においては特徴的なグラフの構造と問題の種類に対して実験し，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．8 章においてこれらをまとめる．

第2章

分散制約最適化問題と関連研究

本章では，分散制約最適化問題 (Distributed Constraint Optimization Problems, DCOP) の基本的な形式化，および現在研究されている DCOP の解法について説明する．

2.1 分散制約最適化問題 (DCOP)

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．

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．

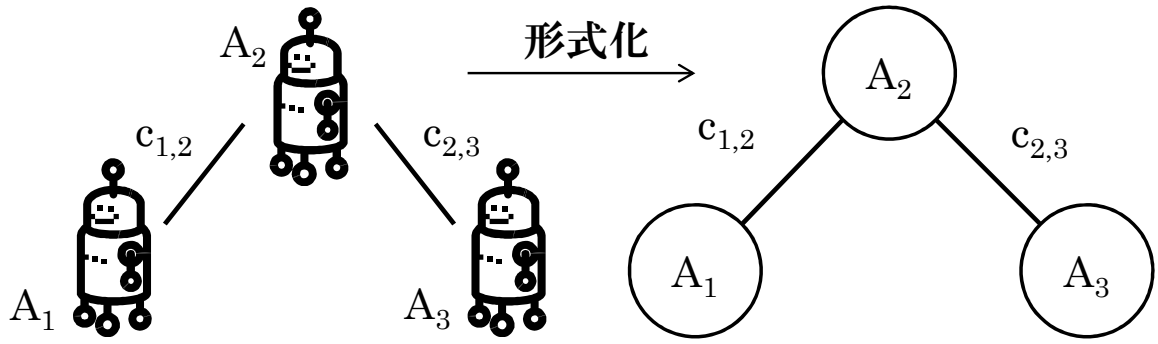


図 2.1: マルチエージェントシステムの DCOP への形式化

2.2 DCOP の関連研究

DCOP の解法の関連研究について説明する．DCOP の解法は必ず最適解が得られる厳密解法と，近似解が得られる非厳密解法の大きく 2 つに分類される．

2.2.1 厳密解法

DCOP の厳密解法として，ADOPT[4](Asynchronous Distributed Constraint Optimization)，DPOP[5](Dynamic Programming Optimisation Protocol) などが提案されている．ADOPT と DPOP は，制約網に対して，前処理として，深さ優先木などの生成木および，それにもとづく擬似木 (pseudo-tree) を生成する．擬似木により定義される変数の半順序関係に従った，メッセージ交換型の探索アルゴリズムにより最適解を求める．これらのアルゴリズムは最適となる解を確実に求めることができるが，変数や制約密度などの問題の規模に対して，計算/空間複雑度・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問題が挙げられる．DPOP では，前処理で作成した疑似木の幅 (induced-width) にメッセージのサイズが依存しており，与えられる問題によってはメッセージサイズ・メモリ使用量などが指数的に増加するためエージェントに用いられるデバイスの性能に制限がある場合には，計算量・メモリ使用量などにおいて問題となる．

2.2.2 非厳密解法

最適解が求まるとは限らないが，比較的少ない計算で近似解を求める解法として，DSA(Distributed Stochastic Algorithm)，Max-Sum アルゴリズムが挙げられる．DSA では，各エージェントは自身の持つ制約に関係する近傍エージェントの状態に基づいて，確率的に状態を更新する．確率的に状態を更新する事によって，連続して制約違反となる状態を取る事を避けている．この手法では，エージェントの状態に関する情報のみをメッセージとして送受信するので，通信コストを低く抑えることが出来る．そのため比較的大規模なシステムに適しているといえる．しかし，各エージェントは近傍エージェントの状態のみに基づいて自身の状態を決定しているため，局所的な最適解に収束しやすく，エージェント数や制約が多い複雑なグラフになると解の精度が低下してしまう．Max-Sum アルゴリズムでは，近傍エージェントから隣接する変数がどのような状態を取るべきかというメッセージを，周囲の制約を考慮して送信する．そのメッセージを用いて，各エージェントは周辺関数を計算し，全体として最適である自身の状態を取るため，より解の精度が向上すると考えられる．そこで本研究では，Max-Sum アルゴリズムとその評価関数を拡張した MS-Stable について注目する．

第3章

従来手法 – Max-Sumと DCOPへの適用

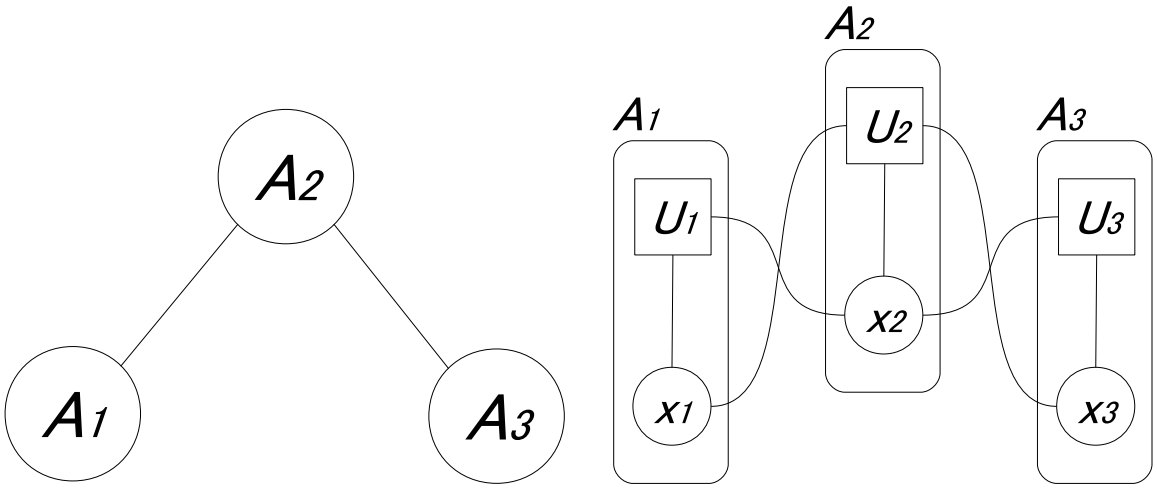
本章では，Max-Sum とその拡張である MS-Stable について述べる．Max-Sum に基づく手法で用いる，問題のグラフの表現，メッセージの計算，周辺関数の計算，計算量とアルゴリズムの特性について述べる．また頂点彩色問題やコスト最小化問題などの DCOP への適用方法について述べる．

3.1 Max-Sum アルゴリズム

3.1.1 Max-Sum におけるグラフ表現

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 [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 3.1(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 3.1(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 3.1: 問題のグラフ表現

3.1.2　全体の動作

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．Max-Sum では，各エージェントはこれらの動作を非同期に行うことができる．すなわち，各エージェントは，送信するメッセージの計算時に保持している最新のメッセージを用いて計算することで，最新の情報に基づいたメッセージの計算および評価値の伝搬を行うことができる．

3.1.3　変数ノードから関数ノードへのメッセージ

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．

$$Q_{n\to m}(x_n) = \alpha_{nm} + \sum_{m'\in M(n)\backslash m} R_{m'\to n}(x_n) \tag{3.1}$$

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．

$$\sum_{x_n} Q_{n\to m}(x_n) = 0 \tag{3.2}$$

3.1.4 関数ノードから変数ノードへのメッセージ

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．

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m\backslash n}\left(U_m(\mathbf{x}_m) + \sum_{n'\in N(m)\backslash n} Q_{n'\to m}(x_{n'})\right) \tag{3.3}$$

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 (3.3) 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．

3.1.5 周辺関数

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．

$$Z_n(x_n) = \sum_{m\in M(n)} R_{m\to n}(x_n) \tag{3.4}$$

式 (3.4) 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 (3.4) で計算される周辺関数の値は近似値となり [9]，

$$Z_n(x_n) \approx \max_{\mathbf{x}_m\backslash n} \sum_{m=1}^{M} U_m(\mathbf{x}_m) \tag{3.5}$$

を表す．Max-Sum に基づく手法では，周辺関数が最大となる変数値を選択することによって，各エージェントは準最適解となる変数値を選択することができる．

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．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) + \sum_{i\in N(m)\backslash m} f_{m,i}(x_m, x_i) \tag{3.6}$$

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．また，Max-Sum は評価関数を最大化する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．その場合において，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．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m) \setminus m} x_m \otimes x_i \tag{3.7}$$

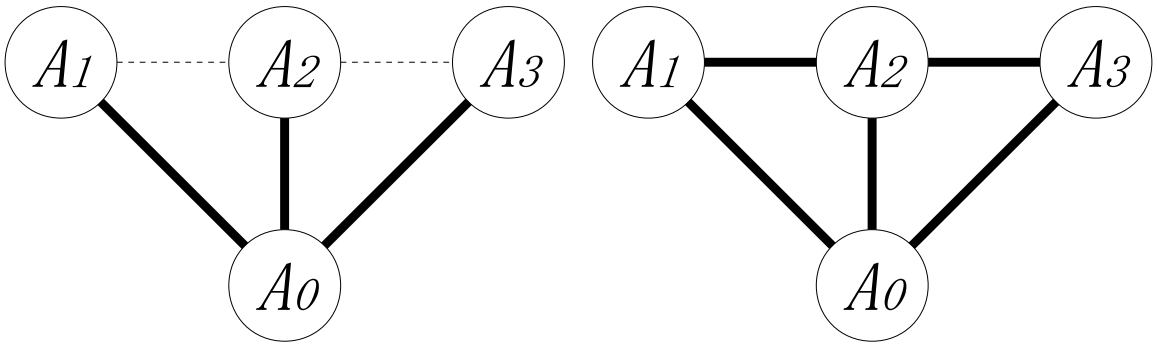
このとき

$$x_i \otimes x_j = \begin{cases} 1 & (x_i = x_j) \\ 0 & (x_i \neq x_j) \end{cases} \tag{3.8}$$

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 (3.6) もしくは式 (3.7) 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[6] 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



(a) Max-Sum により評価される制約　　(b) MS-Stable により評価される制約

図 3.2: 評価関数と評価される制約

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．

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) + \sum_{i \in N(m)} \sum_{j \in C(i,m)} f_{i,j}(x_i, x_j) \tag{3.9}$$

特に，彩色問題へ適用する場合には

$$U_m(\mathbf{x}_m) = \gamma_m(x_m) - \sum_{i \in N(m)} \sum_{j \in C(i,m)} x_i \otimes x_j \tag{3.10}$$

となる．ここで，$C(i,m)$ は次式のように表される[1]．

$$C(i,m) = \{l \in N(m) | l > i \wedge (i \in N(l) \vee l \in N(i))\} \tag{3.11}$$

具体的な例を示すと，エージェント A_0 が Max-Sum を用いた場合に評価される制約は図 3.2(a) となり，A_0 が MS-Stable を用いた場合に評価される制約は図 3.2(b) となる．評価される制約を太線で示す．図 3.2(a) の Max-Sum を用いる場合の例では，エージェントは自身のエージェントと制約で関係するエージェント間の制約のみを評価するため，$A_0 - A_1$，$A_0 - A_2$，$A_0 - A_3$ 間の制約が評価される．一方，図 3.2(b) の MS-Stable を用いる場合の例では，エージェントは Max-Sum を用いた場合に評価する図 3.2(a) の制約に加えて，隣接するエージェント間の $A_1 - A_2$，$A_2 - A_3$ の制約も評価に用いる．

[1]式 (3.11) 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 [6] に基づいている．

このように，MS-Stable では Max-Sum で用いられていない制約も評価することによって，より正確な評価値を計算することができるため，基本的には Max-Sum よりも高い解の精度が得られる．

3.5 Max-Sum と MS-Stable の計算量

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 (3.3) 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 (3.3) の計算に必要な変数値の組み合わせ数は

$$\sum_{i\in N(m)\setminus m} |D_m|\cdot|D_i| \tag{3.12}$$

である. ただし式 (3.12) は，次のように冗長な計算を省いた Max-Sum を使用した時の組み合わせ数である．まず，式 (3.3) に式 (3.6) を代入し変形すると次式となる．

$$\begin{aligned} R_{m\to n}(x_n) = \max_{\mathbf{x}_m\setminus n}\Bigg(& Q_{m\to m}(x_m)+\gamma_m(x_m) \\ & + \sum_{i\in N(m)\setminus m,n} \big(f_{m,i}(x_m,x_i)+Q_{i\to m}(x_i)\big) \\ & + f_{m,n}(x_m,x_n)\Bigg) \end{aligned} \tag{3.13}$$

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 (3.13) は

$$R_{m\to n}(x_n) = \max_{x_m}\Bigg(Q_{m\to m}(x_m) + \gamma_m(x_m) + \sum_{i\in N(m)\setminus m,n} \big(\max_{x_i}(f_{m,i}(x_m,x_i) + Q_{i\to m}(x_i))\big) + f_{m,n}(x_m,x_n)\Bigg) \tag{3.14}$$

となり，式 (3.12) 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

$$\prod_{i\in N(m)} |D_i| \tag{3.15}$$

まで増加する．

第4章

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するために，実行途中に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

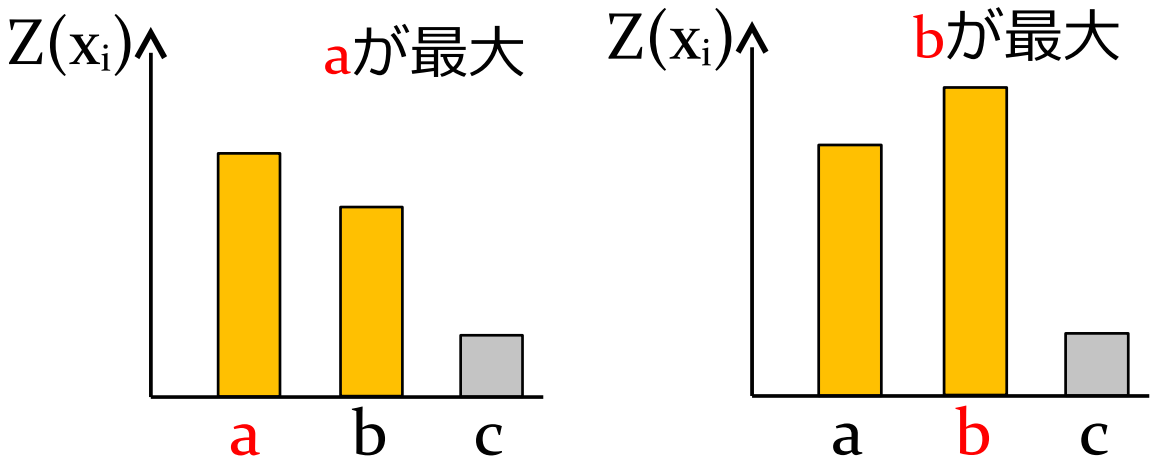
Max-Sum は比較的小さい計算量において高い精度の解が得られる手法であるが，複雑な制約網の問題に適用した場合に解の精度が低下する問題がある．その解の精度の低下は，隣接するエージェント間の制約を評価しないことが原因のひとつとして挙げられる．解の精度の低下を抑制する方法として，隣接するエージェント間の制約も評価する MS-Stable を評価関数として用いる方法がある．しかし，3.5 節で述べたように，解法における最も計算量が必要となるメッセージ R での計算において，隣接するエージェント間の制約を含めることが大幅な計算量の増加の原因となる．

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，分散環境を想定した場合には，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，大域的な情報収集のための通信遅延などの観点から難しいと考えられる．そ

のため分散環境を想定するアルゴリズムとしては，局所的な情報に基づいて評価関数の変更の判断を行うことが望ましい．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 (3.5) 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と定義する．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，大域的な解が低下する場合としては，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．例えば，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 4.1(a) に示す．図 4.1(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 4.1(b) となったとする．図 4.1(b) では，Max-Sum では評価しない制約により，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 4.1(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

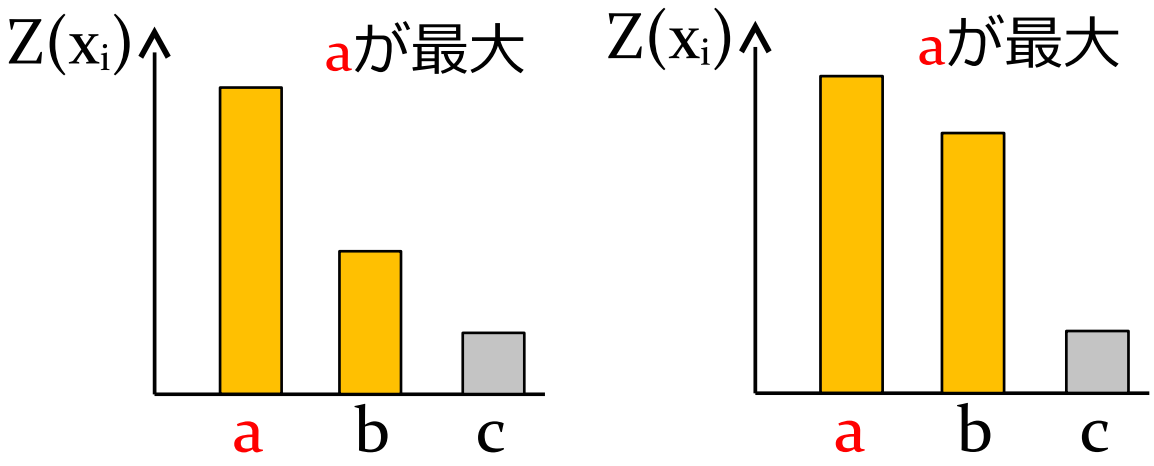


(a) Max-Sum で評価した場合の $Z_i(x_i)$　　(b) MS-Stable で評価した場合の $Z_i(x_i)$

図 4.1: 評価関数が最大となる変数値が異なる例 (Max-Sum で $Z_i(a)$ と $Z_i(b)$ の差が小さい場合は，両者の $Z_i(a)$ と $Z_i(b)$ の大小が異なる可能性が高いと考えられる)

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．例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 4.2(a) となったとする．図 4.2(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 4.2(b) となるとする．図 4.2(b) の例では，図 4.2(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



(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 4.2: いずれの評価関数でも周辺関数の最大値が変化しない例 (Max-Sum で $Z_i(a)$ と $Z_i(b)$ の差が大きい場合は，両者の $Z_i(a)$ と $Z_i(b)$ の大小が異なる可能性は比較的低いと考えられる)

れるわけではない．何故ならば，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り替えでは，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有望な部分に対して MS-Stable を割り当てることはできる．

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 の和を取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．

$$Z_n(x_n) = \sum_{m \in M(n)} R_{m \to n}(x_n) \approx \max_{\mathbf{x}_m \backslash n} \sum_{m=1}^{M} U_m(\mathbf{x}_m) \tag{4.1}$$

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，前述の通り MS-Stable を用いる計算コストは Max-Sum に比べて極めて大きいため，周辺関数の均衡が見られるエージェント全てにおいて，常に MS-Stable を用いることは効率的ではない．そこで，出来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，常に 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．常に MS-Stable を用いる手法では，常に 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，常に 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．均衡の検出は次の式に従って計算する．

$$Z_i(x_i^{max}) - Z_i(x_i^{max'}) < \delta \tag{4.2}$$

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．周辺関数の均衡を検出し，評価関数を切り替える処理を次の Algorithm1 に示す．ここで keepFuncCycle とは，Max-Sum から MS-Stable に切り替わった時，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．

Algorithm 1 周辺関数に基づく評価関数の切り替え手法 Z-MSS

if $Z_i(x_i^{max}) < Z_i(x_i^{max'}) + \delta$ **then**
  use MS-Stable as $U_i(x_i)$
  keepFuncCycle $\leftarrow \lambda$
else if keepFuncCycle ≤ 0 **then**
  use Max-Sum as $U_i(x_i)$
else
  keepFuncCycle $\leftarrow$ keepFuncCycle -1
end if

4.5 Z-MSS の全体の動作

Z-MSS は，周辺関数の計算をする毎に周辺関数の均衡を検出し，評価関数を MS-Stable に切り替える．そのため，Z-MSS を実行している間は，各エージェントが MS-Stable に切り替える割合が変化する．その MS-Stable に切り替えているエージェントの割合は，アルゴリズムの開始からの時間，すなわち解法が開始してから準最適解が得られるまでのどの時点であるのかや評価関数 Max-Sum に切り替えた際に解の悪化がみられる問題であるかなどに影響される．一例として，Z-MSS の実行開始からの全体の利得と MS-Stable に切り替えているエージェントの割合を表した図 4.3 を示す．

実行開始時点 P1

図 4.3 の P1 はアルゴリズムの開始時点であり，Z-MSS の各エージェントはまだ周囲のエージェントとメッセージの交換および周辺関数の計算を行なっていない状況である．アルゴリズム開始時点は全体の利得が小さく，また周辺関数の均衡が起きやすい状況ではあるが，メッセージの交換が不十分なことによる情報不足により周辺関数の正確な値が計算できないことから，Z-MSS での初期の評価関数からの切り替えが起きない．そのために，P1 の時点では各エージェントは Max-Sum を選択している割合が多い．

Z-MSS のアルゴリズムの開始時点において，Max-Sum ではなく MS-Stable から開始するという方法もある．その場合，アルゴリズムの開始時点から高い精度の解が得られることが期待できる．しかしながら，前述のとおり，MS-Stable の計算コストは

非常に大きい．そのため，アルゴリズムの開始時点から MS-Stable の割り当てをすると，不必要なエージェントにまで MS-Stable が割り当てられることになる．MS-Stable の割り当てが不必要なエージェントが多くの制約で結ばれている場合には，MS-Stable の割り当てが一時的であっても大幅に計算コストが増大する．そのため，Z-MSS では開始時点において Max-Sum を用いている．

開始から利得が上昇するまでの時点 P2

図 4.3 の P2 の時点では，アルゴリズムが開始され，各エージェントにおいてメッセージの交換および周辺関数の計算が行われた時点である．エージェントによるメッセージの交換により，周囲のエージェントの制約を反映した評価値を得ることができ，その評価値を用いて周辺関数の計算および周辺関数の均衡の検出が行われる．アルゴリズムの初期時点では，全体の利得は低く，周辺関数の均衡が起きやすい状況にある．そのため，周辺関数の均衡の検出によって MS-Stable に切り替えられるエージェントの割合が最も多くなる時点である．

準最適解に到達する時点 P3

図 4.3 の P3 の時点は，各エージェントのメッセージ交換によって，評価値が全体に行き渡っている状況である．そのため，各エージェントは正確な周辺関数の計算に基づいて，最も利得が高くなる変数値をとるができ，準最適解となる変数値を選択している．このとき，評価値が全体に行き渡り，解が安定していることから，多くのエージェントでは解の均衡は起こりにくい状況である．そのため，エージェントが MS-Stable を選択する割合は比較的小さい．

評価関数の切り替えにより評価値が減少する時点 P4

図 4.3 の P4 の時点では，P3 の時点において準最適解となり，各エージェントが MS-Stable から Max-Sum に切り替えたために，利得が低下している時点である．P4 の状況では，Max-Sum によって評価されたメッセージが伝搬し，周辺関数の計算にその

メッセージが使われることによって，再び周辺関数の均衡があらわれる．Z-MSS では，その周辺関数の均衡を検出するため，再び MS-Stable の割合が増加し始める．

なお，図 4.3 の例では，MS-Stable から Max-Sum に切り替えた場合に全体の利得が低下する例となっているが，一部の問題では MS-Stable から Max-Sum に切り替えた場合に利得が低下しない．そのような問題においては，Z-MSS では MS-Stable の割合が少ない状態で維持されるため，通常の問題に比べて解の精度を維持したまま，大幅に計算コストを削減できる．

再び MS-Stable が増加する時点 P5

図 4.3 の P5 の時点では，P4 で MS-Stable に切り替わったエージェントのメッセージが行き渡っておらず，全体の利得が低下している．このように，MS-Stable に切り替えて計算したメッセージがグラフ全体のエージェントに対して十分に行き渡るまでにはある程度時間がかかる．つまり，MS-Stable により計算されたメッセージが行き渡るまでは，エージェント全体における MS-Stable の割合は増加し続ける．

MS-Stable が減少しはじめる時点 P6

図 4.3 の P6 の時点では，P4，P5 の時点で MS-Stable で計算されたメッセージが十分にグラフ全体のエージェントに行き渡ったために全体の利得が向上し，周辺関数の均衡が解消するため，再び MS-Stable の割合は減少しはじめる．

4.6 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．パラメータ δ の概念図を図 4.4 に示す．図 4.4 の例では，x_i^{max} である a から δ の範囲内に，$x_i^{max'}$ である b が存在しないため，評価関数の切り替えは起きない．例えば，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 MS-Stable 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．

パラメータ λ の概念図を図 4.5 に示す．図 4.5 では，エージェント A_i が Max-Sum から MS-Stable に切り替え，再び Max-Sum に切り替えた例である．この図 4.5 における MS-Stable に切り替えている期間を λ とする．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 に戻るまで時間を要するので，結果的に評価関数を MS-Stable 切り替えているエージェントが多くなり，解の精度が向上し，計算コストが大きくなる．

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するためには，これらの値をうまく調整する必要がある．例えば，MS-Stable を切り替えている時間を表す λ は，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，問題に合わせた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．

4.7 アルゴリズムの正しさ

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 とでは，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．しかし，Max-Sum に基づく解法では，各エージェントで式 (3.2) 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は，評価値の近似値として扱われる．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．

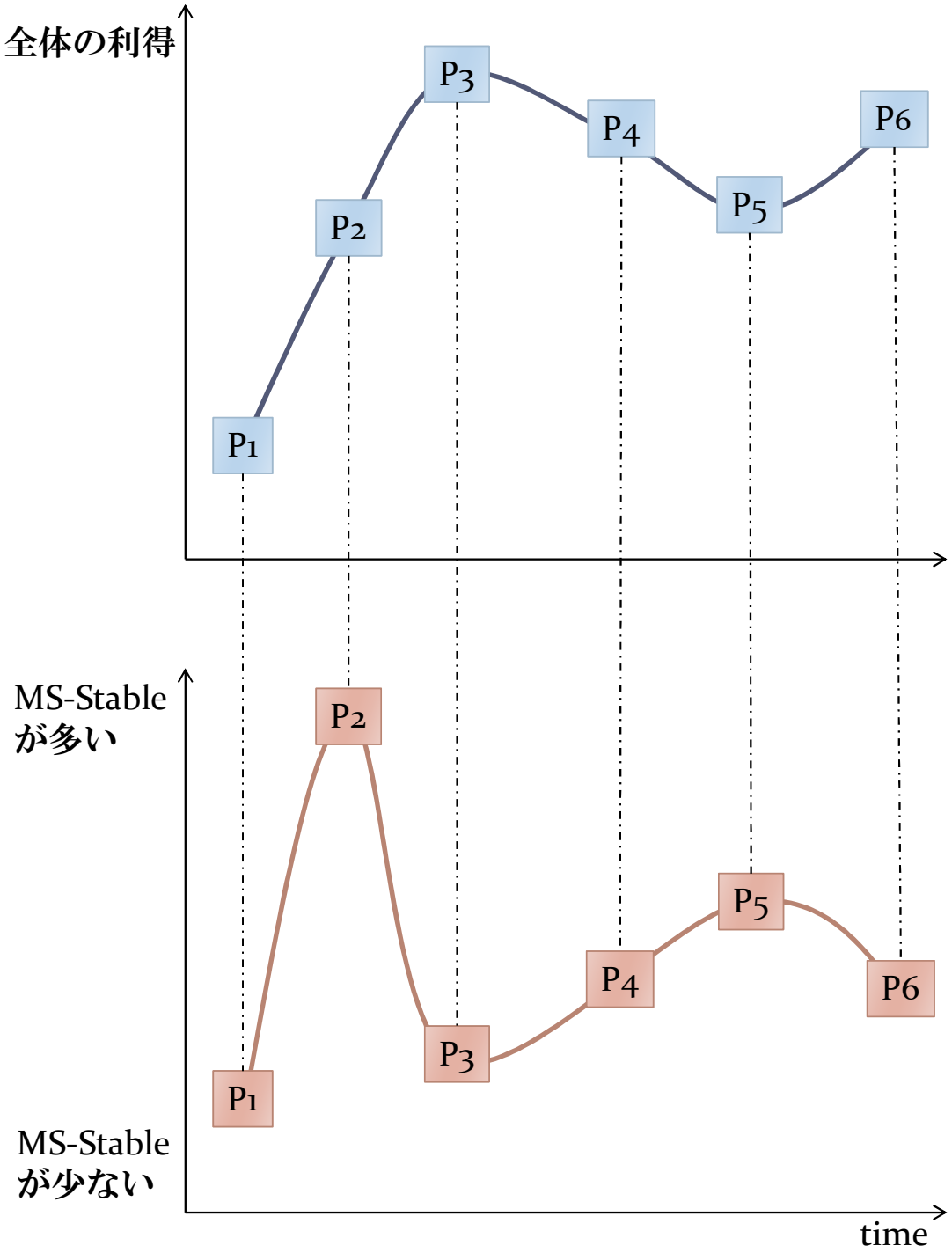


図 4.3: 全体の利得とエージェントが MS-Stable を選択する割合

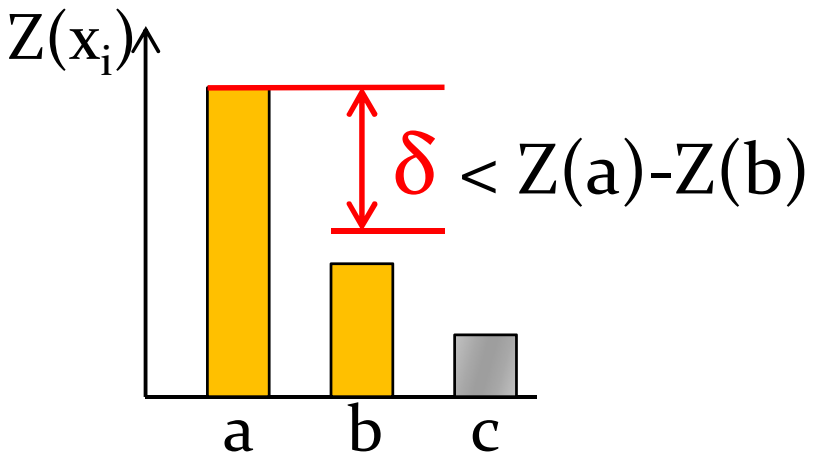


図 4.4: 周辺関数と Z-MSS のパラメータ δ(図は均衡が起きていない例)

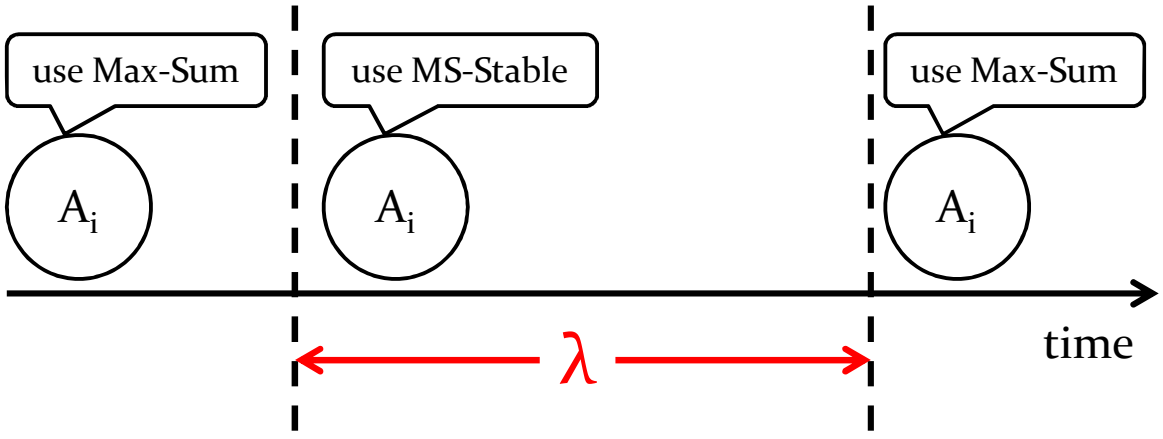


図 4.5: MS-Stable への切り替え期間 λ

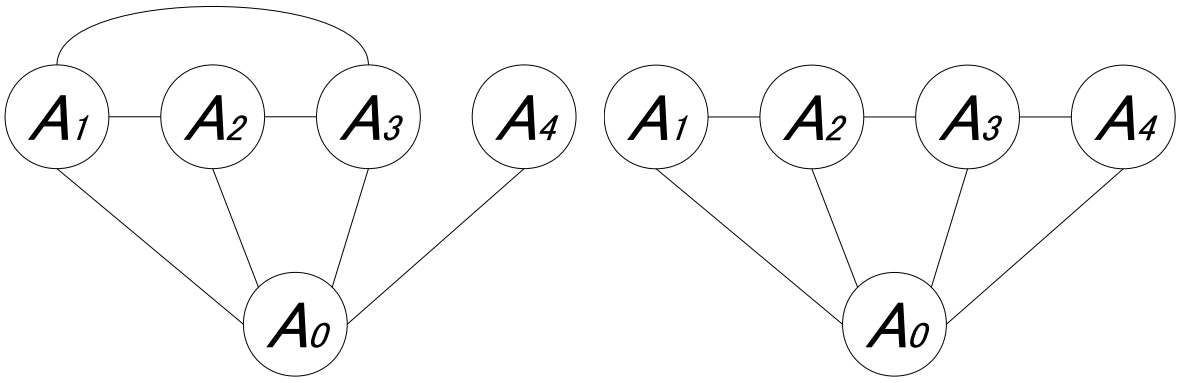
第5章

Max-SumとMS-Stableの解析と切り替えの効果

本章では，Max-Sum と MS-Stable の評価関数の特性および評価関数を切り替えた場合の効果について予備実験を行い，考察する．

5.1 特定のグラフ構造における評価

3 章で述べたように，複雑な制約網に対して Max-Sum を適用した場合には解の精度が低下し，MS-Stable を適用した場合にはその解の精度低下を抑えることができることは文献 [6] で示されている．しかしながら文献 [6] ではランダムな制約網にしか適用されておらず，具体的にどのような制約網において Max-Sum の解の精度が低下するかは不明であった．そこで本研究では，経験的に解の精度の悪化が見られた 4clique に注目した．4clique とは 4 頂点からなる完全部分グラフであり，図 5.1(a) の A_0，A_1，A_2，A_3 のように構成される．予備実験 1 では，エージェント数および制約数が同じ図 5.1(a)，図 5.1(b) の制約網に対して Max-Sum と MS-Stable の解の精度を比較することで，4clique が Max-Sum と MS-Stable に与える影響について考察する．問題は頂点彩色問題を用いた．解の精度として，制約の両端が同じ変数値である場合に違反とし，平均の違反数を比較した．各制約網に対して，Max-Sum，MS-Stable をそれぞれ 100 回試行し，平均を取った．図 5.1(a) の結果を表 5.1 に，図 5.1(b) の結果を表 5.2 に示す．表 5.1 では，Max-Sum と MS-Stable との違反数を比較すると，Max-Sum は約



(a) 4clique を含む場合　　(b) 4clique を含まない場合

図 5.1: 予備実験 1 における頂点彩色問題の制約網

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

表 5.1: 図 5.1(a) での平均違反数

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

表 5.2: 図 5.1(b) での平均違反数

1.6 倍の違反数となっている．一方，表 5.2 では，Max-Sum と MS-Stable の違反数はほぼ同じとなり，違いが見られなかった．これらの結果から，制約網における 4clique の有無が，評価関数 Max-Sum と MS-Stable の解の精度の差を生み出す要因のひとつとなっていると考えられる．

5.2　ランダムに生成した問題における比較

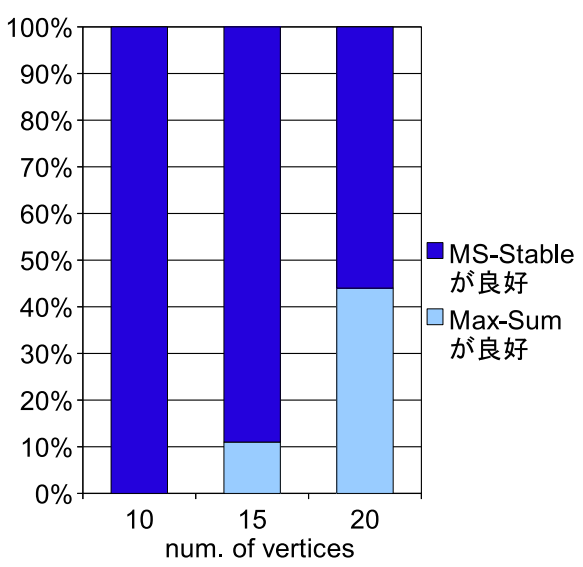
前述のとおり，拡張された評価関数 MS-Stable は，複雑な制約網での解の精度，特に 4clique を含む問題における解の精度を高めることができる．しかし，複雑でない制約網の問題に適用した場合には Max-Sum よりも大幅に計算量が増大するが，5.1 節で示したとおり，解の精度も同程度にしかならないと予想される．そこで，問題をランダムに生成した場合に，どの程度の割合の問題において MS-Stable が有効である問題があるかを確かめるための予備実験 2 を行った．予備実験 2 では，3 色の頂点彩色問題

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合いは図 5.2(a) のようになった．

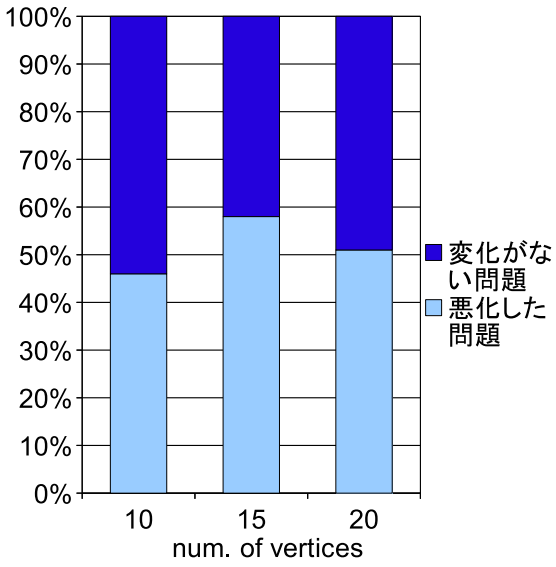
頂点数 10 の場合，全ての問題において Max-Sum よりも MS-Stable が高い解の精度を示した．頂点数 15 の場合では 11 例の問題で Max-Sum が MS-Stable よりも高い解の精度を示し，頂点数 20 の場合では 44 例の問題で Max-Sum が高い解の精度を示した．このような結果となった理由として，制約の密度が挙げられる．予備実験 2 では，頂点数の 3 倍の制約をランダムに設定しているため，頂点数が多くなるにつれ，制約の引き得る組み合わせに対して，制約の密度が小さくなる．そのことから，各変数が持つ制約の数に偏りが生じる．すなわち，頂点数 20 の例題において MS-Stable の効果が小さかったことから，MS-Stable が高い解の精度を示す問題は，制約の密度や 4clique 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．しかしながら，自身のエージェントが保持している情報のみから，Max-Sum が高い解の精度示す問題と MS-Stable が高い解の精度を示す問題を，アルゴリズム実行開始前において判断することは難しい．また，周囲のエージェントの制約を収集して用いる方法も考えられるが，追加される処理のための通信と計算が新たに生じる上に，収集した情報から正確に MS-Stable の効果を推定することは難しい．その点において，Z-MSS は新たに通信を必要とすることなく，自身の周辺関数の状況から判断して MS-Stable の効果が小さい問題に対しての MS-Stable の割り当てを減らすことが期待できるため，効果が大きいと考えられる．

5.3 評価関数の切り替えによる効果

アルゴリズムの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，ある時間で factor グラフのネットワーク上を伝搬して



(a) Max-Sum，MS-Stable が高い解の精度を示す問題の割合



(b) 評価関数の切り替えにより解が変化した問題の割合

図 5.2: 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

いき，その後に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解の精度に影響を与えるのかを予備実験 3 により確認する．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 3 を行った．予備実験 3 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 5.2 節の予備実験 2 と同様である．また評価関数の切り替えは，10 サイクル目で行い，全てのエージェントが MS-Stable から Max-Sum に切り替える設定とした．

100 例題のうち，解が悪化した問題と変化がなかった問題の割り合いを図 5.2(b) に示す．頂点数 10 の場合では 46 例題，頂点数 15 では 58 例題，頂点数 20 では 51 例題において MS-Stable から Max-Sum に切り替えた場合に解の悪化が見られた．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．Z-MSS では周辺関数に基づいて評価関数の切り替えを動的に行うため，解が収束した後に，MS-Stable を使い続けることはない．そのため，解の悪化が見られない問題に対しては，解の精度を低下させずに計算コストを削減することができ，効果的であると考えられる．また予備実験 2 と異なり，頂点数による変化は見られなかった．このことから，頂点数や周囲の制約だけから，MS-Stable の効果を推定することは難しいと考えられる．

第6章

従来手法と提案手法の比較

従来手法 Max-Sum，MS-Stable と提案手法 Z-MSS を，頂点彩色問題と二項制約のコスト最小化問題において比較する．また Z-MSS と理想値を比較することで，Z-MSS において評価関数の切り替えが適切に行われているかについて評価する．

6.1 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．制約数は頂点数の 3 倍の数をランダムに設定した．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み合わせの数を用いた．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，均衡を表すパラメータ $\delta = 0.4$，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．平均の違反数は図 6.1(a) に，平均計算量は図 6.1(b) に示す．

図 6.1(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，ほぼ全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%改善し 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り替えによって，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 6.1(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 6.1(a) をみると Z-MSS の違反数が MS-Stable より小さくなった．また図 6.1(b) をみると，計算量は MS-Stable に比べ，Z-MSS は約 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挙げられる．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．それに加えて，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．その結果，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．

6.2 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ひとつであるが，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全ての制約のコストの合計値を最小化する問題とする．

実験諸元は，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．その場合の各手法おける平均の評価値の和を図 6.2(a) に，平均の計算量を図 6.2(b) に示す．図 6.2(a) では彩色問題で

の評価図 6.1(a) と異なり，全ての頂点数において Max-Sum と MS-Stable の差が小さくなり，解の改善幅は小さいものとなったが，Z-MSS と MS-Stable はほぼ同等の評価値となった．また図 6.2(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 6.1(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 6.2(a) 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 が良い結果となった．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 6.2(b) では，彩色問題での計算量図 6.1(b) 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，全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 で導きだ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．

6.3 Z-MSS と理想値との比較

ここでは，Max-Sum が高い解の精度を示す問題，MS-Stable が高い解の精度を示す問題，MS-Stable から Max-Sum に評価関数を切り替えた際に解が悪化しない問題に対して，Z-MSS がどの程度適切な評価関数の切り替えを行っているかについて評価する．評価には，5.2 章，5.3 章と同様の 100 例題の問題を用いて行う．すなわち，頂点数 10 の問題は全て MS-Stable が高い解の精度を示し，頂点数 15 の問題では 11 例の問題で MS-Stable が高い解の精度を示し，頂点数 20 の問題は 44 例の問題で MS-Stable が高い解の精度を示す問題である．また，各頂点数の問題のうち，約半数は MS-Stable から Max-Sum に評価関数を切り替えても解の精度に変化がない問題である．これらの問題に対して，次の 2 つの適用方法により得られた解の精度と計算量を理想値として定義する．

Max-Sum or MS-Stable

事前に各問題に対して Max-Sum，MS-Stable を適用して問題を解き，Max-Sum の方が解の精度が高かった問題に対しては Max-Sum を割り当て，MS-Stable の方が解の精度が高かった問題に対しては MS-Stable を割り当てる．これにより，評価関数の動的な切り替えを行わなかった場合における，最高の解の精度の場合での計算量を求めることができる．

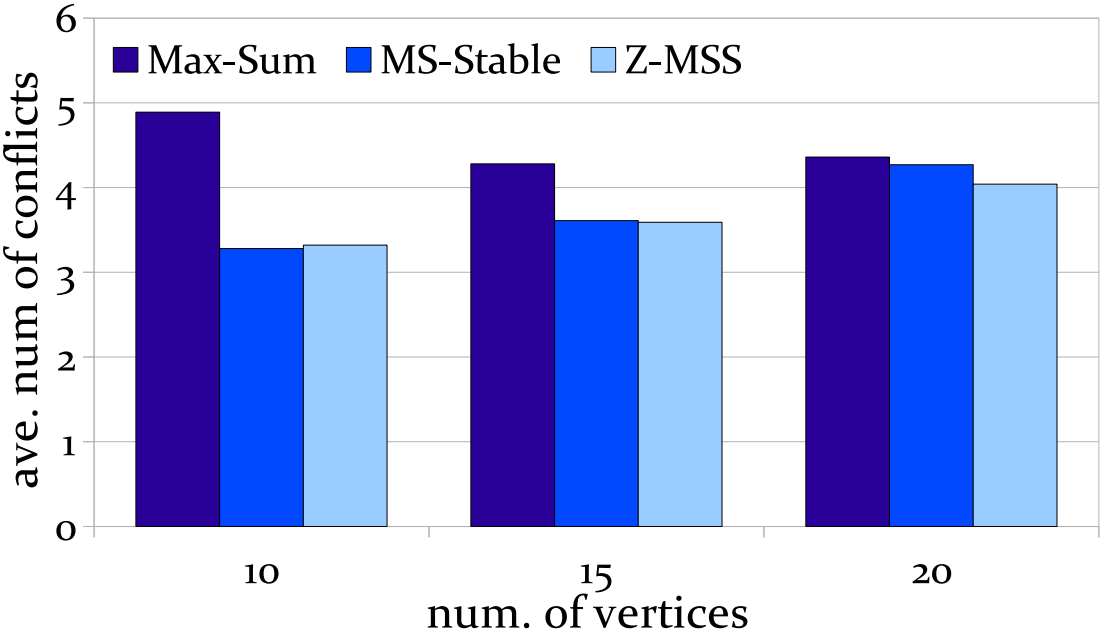
Max-Sum or Z-MSS

Max-Sum or MS-Stable と同様に，事前に行った評価を用いて，Max-Sum の方が解の精度が高かった問題に対しては Max-Sum を割り当て，Z-MSS の方が解の精度が高かった問題に対しては Z-MSS を割り当てる．これにより，Z-MSS を使う必要のなかった問題では Max-Sum しか実行されないため，Z-MSS より解の精度が高く，また計算量も小さい理想値が得られる．

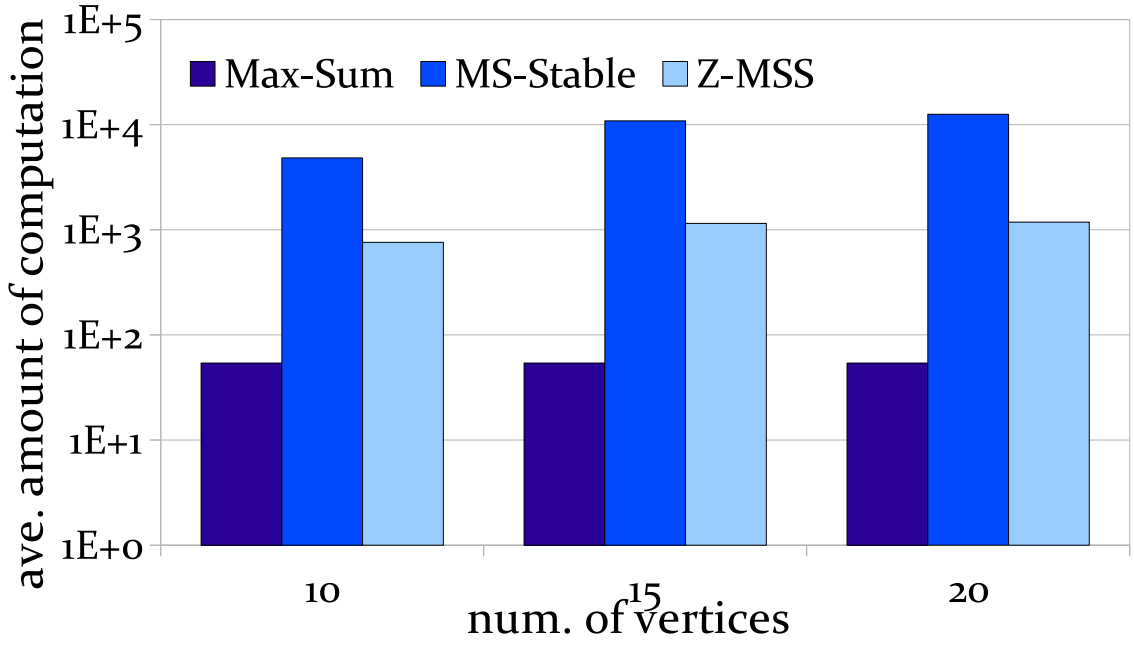
これら 2 つの手法と，Z-MSS を適用した場合の平均違反数を図 6.3(a) に，計算量を図 6.3(b) に示す．図 6.3(a) を見ると，頂点数 10 の場合では Max-Sum or MS-Stable, Max-Sum or Z-MSS，Z-MSS の順にわずかに解の精度が高くなった．頂点数 15 の場合には，Max-Sum or MS-Stable，Max-Sum or Z-MSS，Z-MSS はほぼ同じ違反数となった．頂点数 20 の場合には，Max-Sum or Z-MSS，Z-MSS，Max-Sum or MS-Stable の順に解の精度が高くなった．頂点数 10 の問題では，図 5.2(a) に示した通り，全ての問題において MS-Stable が高い解の精度を示す問題であり，その場合にはわずかではあるが Max-Sum or Z-MSS，Z-MSS よりも高い解の精度が得られる．頂点数 15 の場合では，約 10%ほどの問題のみ Max-Sum が高い解の精度となる問題が含まれているが，理想値をとった場合でもほぼ Z-MSS と変わらない解の精度となった．頂点数 20 の場合では，約 45%の問題が Max-Sum で高い解の精度を示す問題である．さらに，そのうち約半数の問題では，MS-Stable から Max-Sum に切り替えても解の精度が悪化しない．そのため，Z-MSS の動的に評価関数を切り替えて用いる手法が有効に働いたために，Max-Sum or MS-Stable よりも Max-Sum or Z-MSS，Z-MSS が高い解の精度を示

したと考えられる．以上より，各頂点数においての解の精度の理想値と Z-MSS を比較しても大幅な差は見られない．そのことから，Z-MSS の周辺関数の均衡による評価関数の切り替えによって，問題の特性に応じた評価関数が選択されていると考えられる．

また 6.3(b) の計算量においては，頂点数による変化はみられず，Max-Sum or MS-Stable が極端に大きく，Max-Sum or Z-MSS と Z-MSS がほぼ同じ値となったが，頂点数 15 で比較するとわずかに Max-Sum or Z-MSS が小さい値となった．Max-Sum or MS-Stable は，問題の約半数ある MS-Stable から Max-Sum に切り替えても解の精度が悪化しない問題においても，MS-Stable を使用し続けるため計算量が大きくなったと考えられる．Max-Sum or Z-MSS，Z-MSS ではこれらの問題において MS-Stable を使い続けることがないため，計算量が削減できていると考えられる．また Max-Sum or Z-MSS と Z-MSS の計算量に大きな差がないことから，Z-MSS は Max-Sum が高い解の精度を示す問題に対しても MS-Stable を使い続けることはなく，問題に特性に応じて適切な切り替えが行われていると考えられる．

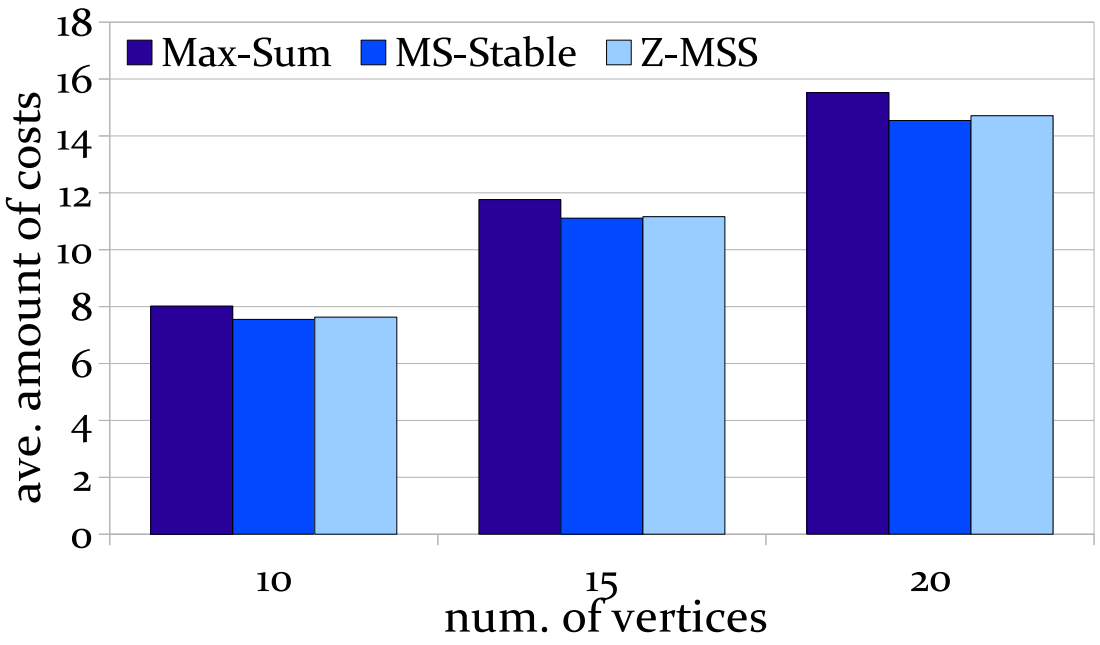


(a) 平均違反数

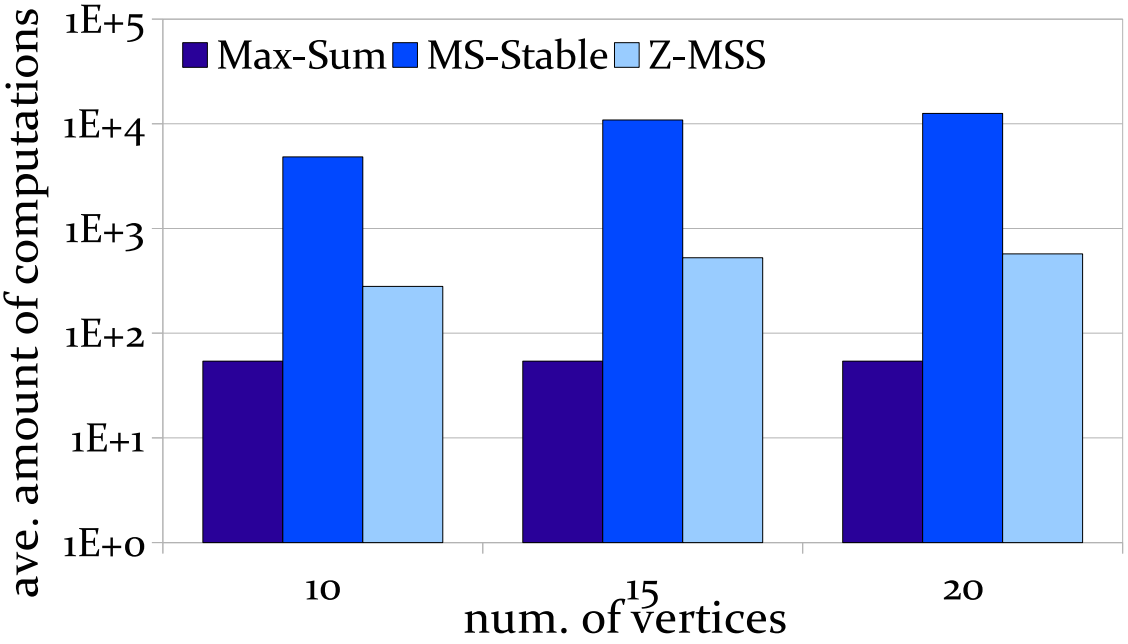


(b) 平均計算量

図 6.1: 彩色問題における評価

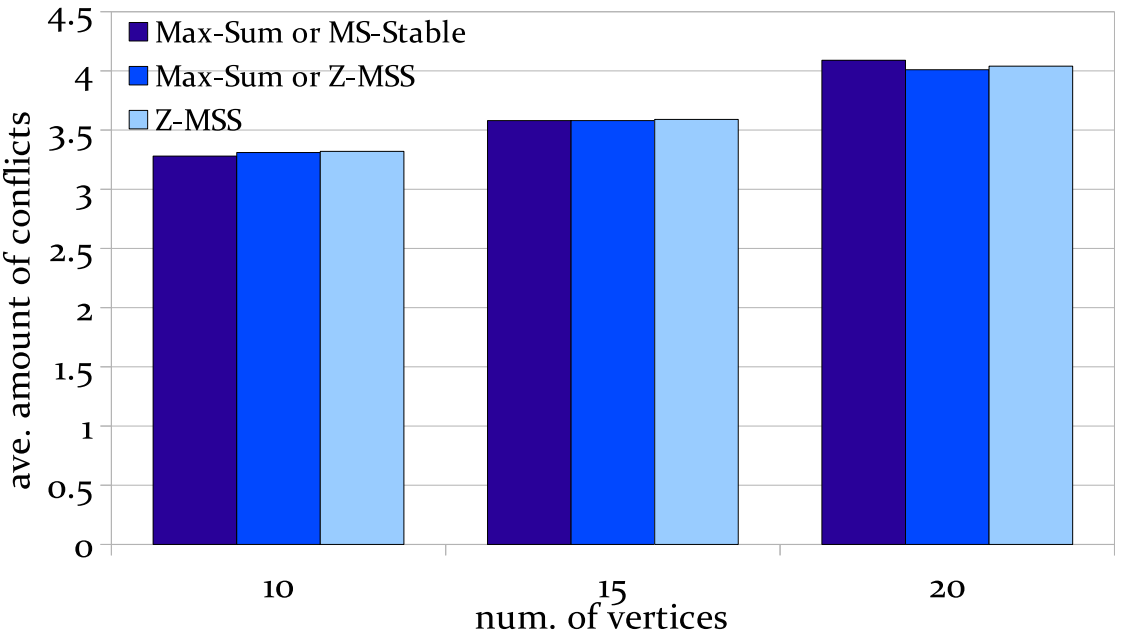


(a) 平均違反数

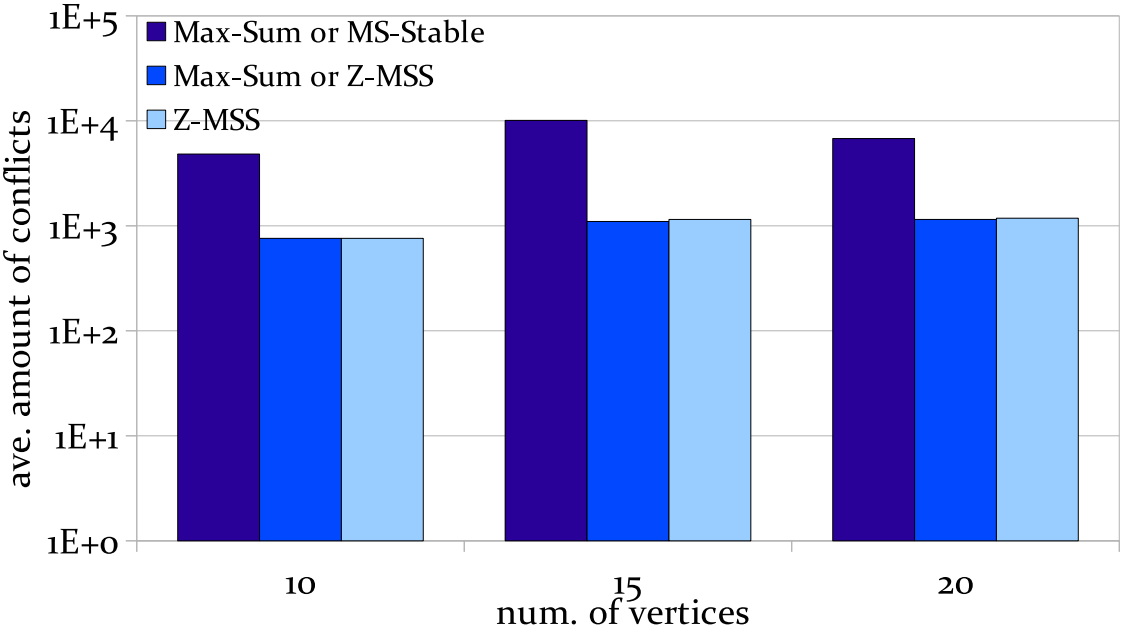


(b) 平均計算量

図 6.2: 二項制約の最小化問題における評価



(a) 平均違反数



(b) 平均計算量

図 6.3: 理想値と Z-MSS との比較

第7章

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り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．ただし本論文では，問題による変化が大きいと考えられるパラメータ δ のみを変化させ，変更サイクル数 λ は固定した．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，全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 は図 7.1 に示すグラフで，4 頂点からなる完全グラフ複数個を線形に連結した，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．制約数は complite が 45，4clique が 34，4clique-random は 60 とした．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 節と同様である．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

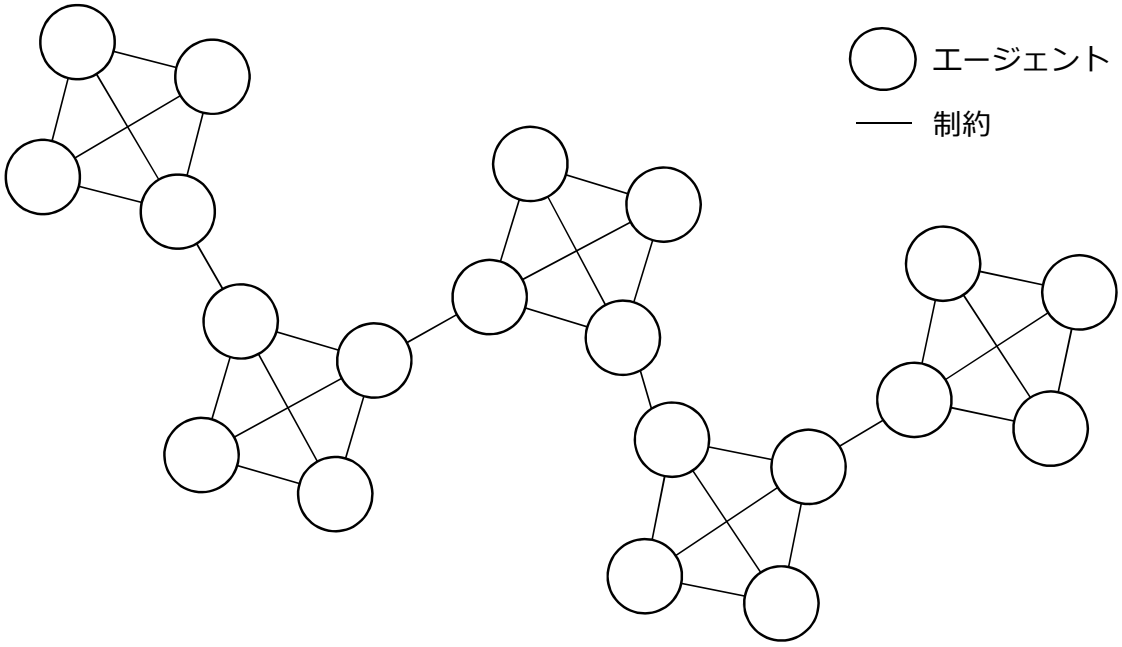


図 7.1: 4clique グラフ

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 7.1 に，計算量を表 7.2 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 7.2 に示す．

問題による違いの傾向としては，表 7.1 の problem 毎に 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た．この理由としては，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難いことが考えられるが，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 $\delta = 0.01 \sim 0.1$ 付近で，重み付き彩色問題では $\delta = 0.1 \sim 1$

problem	structure	Max-Sum	Z-MSS $\delta = 0.001$	Z-MSS $\delta = 0.01$	Z-MSS $\delta = 0.1$	Z-MSS $\delta = 1$	MS-Stable
graph-coloring	complete	17.21	16.61	15.79	12.14	12.14	12.00
	4clique	7.85	7.81	5.06	5.06	5.06	5.09
	4clique-random	8.69	6.88	6.12	6.02	5.99	6.05
weighted graph-coloring	complete	3.13	2.99	2.79	2.28	2.04	2.00
	4clique	1.56	1.44	1.08	0.71	0.62	0.63
	4clique-random	1.26	1.03	0.97	0.95	0.96	0.96
minimization	complete	14.05	13.66	13.48	13.17	12.73	12.51
	4clique	10.15	9.42	9.25	9.10	8.67	8.65
	4clique-random	16.15	15.58	15.18	15.11	15.11	15.02

表 7.1: グラフの構造と問題を変化させた場合のコストの合計

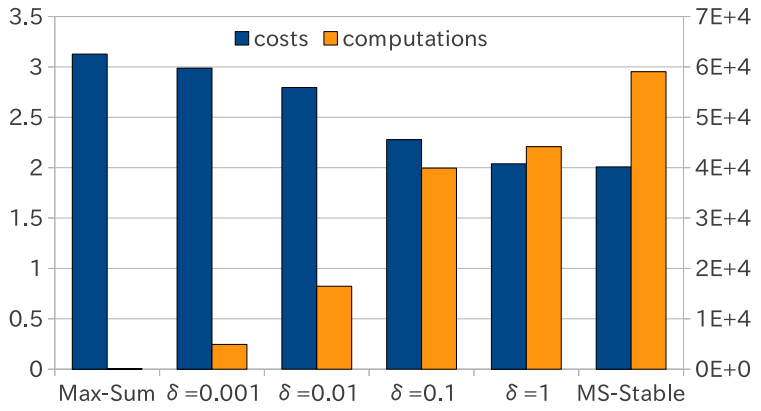
problem	structure	Max-Sum	Z-MSS $\delta = 0.001$	Z-MSS $\delta = 0.01$	Z-MSS $\delta = 0.1$	Z-MSS $\delta = 1$	MS-Stable
graph-coloring	complete	81	4892	15448	57869	57869	59049
	4clique	31	42	141	143	143	146
	4clique-random	54	525	1009	1239	1495	10326
weighted graph-coloring	complete	81	4916	16462	39919	44177	59049
	4clique	31	41	63	86	106	146
	4clique-random	54	487	909	1175	3127	10326
minimization	complete	81	2439	5647	14268	21981	59049
	4clique	31	33	40	49	70	146
	4clique-random	54	159	370	521	1084	10326

表 7.2: グラフの構造と問題を変化させた場合の計算量

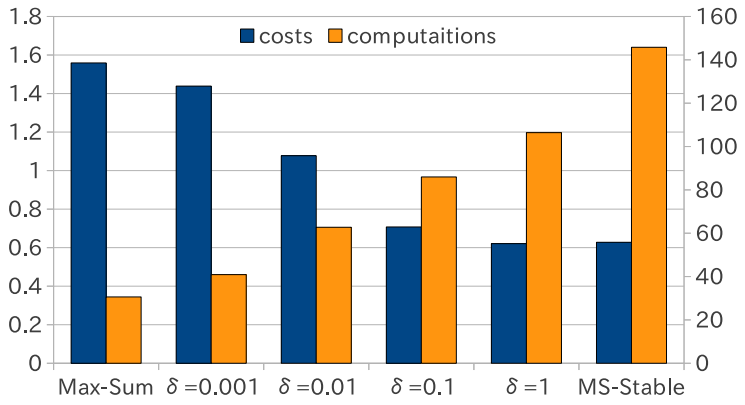
付近，コスト最小化問題では $\delta = 1$ 付近となった．

グラフの構造による違いとしては，表 7.1 の structure 毎に比較してみると，完全グラフの場合は，MS-Stable の割合が増えるにつれ解の改善が見られた．また表 7.2 で完全グラフについて比較すると，解の改善に伴い計算量も大幅に増加した．重み付き彩色問題では図 7.2(a) をみると，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%解の精度が改善したが，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，全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 7.1 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，MS-Stable の割合が増加するほど解が改善したが，隣接エージェント数が

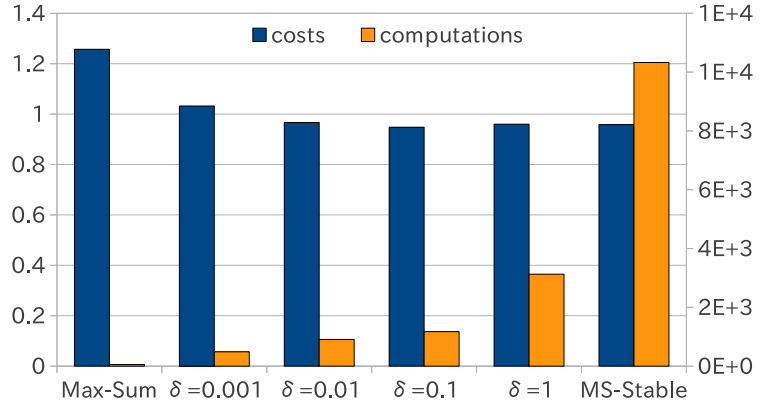
小さいため，表 7.2 の 4clique をみると Max-Sum と MS-Stable の計算量の差が小さい．重み付き彩色問題では図 7.2(b) をみると，Z-MSS は $\delta = 1$ のとき MS-Stable に近い解の精度となり，計算量は MS-Stale に比べ 27%改善したが，Max-Sum と MS-Stable の計算量の差が小さいために，計算量の削減効果は小さいといえる．4clique のような問題では，あらかじめ MS-Stable を用いるか，δ の値を大きく設定することが有利となると考えられる．最後に表 7.1 の 4clique-random について，Max-Sum から MS-Stable までを比較すると，他の問題と同様に MS-Stable の割合が増えるにつれ解が改善したが，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 7.2(c) をみると，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%改善し，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は，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り替えが行われたからと考えられる．



(a) complite の場合



(b) 4clique の場合



(c) 4clique-random の場合

図 7.2: 重み付き彩色問題における解の精度と計算量

第8章

おわりに

本論文では，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り替えおよび，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 がパラメーター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ーを確認できた．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．

Z-MSS のパラメーター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が今後の課題である．

謝辞

本研究のために，多大なご尽力を頂き，日頃から熱心なご指導を賜った名古屋工業大学の松尾啓志教授，松井俊浩准教授，津邑公暁准教授，齋藤彰一准教授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，齋藤研究室の皆様に深く感謝致します．

本研究に関する発表

1. 川東 勇輝, 松井 俊浩, 松尾 啓志: ”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”, 合同エージェントワークショップ&シンポジウム (JAWS2011) ロング発表論文 (Oct. 2011)

2. 川東 勇輝, 松井 俊浩, 松尾 啓志: ”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) ショート発表論文 (Oct. 2010)

参考文献

[1] 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).

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

[3] Macarthur, K. S., Farinelli, A., Ramchurn, S. D. and Jennings, N. R.: Efficient, Superstabilizing Decentralised Optimisation for Dynamic Task Allocation Environments, pp. 25–21 (2010).

[4] Modi, P. J., Shen, W., Tambe, M. and Yokoo, M.: An Asynchronous Complete Method for Distributed Constraint Optimization, pp. 161–168 (2003).

[5] 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).

[6] 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).

[7] Farinelli, A., Rogers, A. and Jennings, N.: Bounded Approximate Decentralised Coordination using the Max-Sum Algorithm, pp. 46–59 (2009).

[8] Zhang, W., Wang, O. and Wittenburg, L.: Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance, pp. 53–59 (2002).

[9] D.J.C.MacKay: *Information theory, inference, and learning algorithms*, Cambridge University Press (2003).