RF-002

連続変数の分散制約最適化問題のための解法
Solution Methods for Distributed Constraint Optimization Problems on Continuous Variables

羽佐田智之 † Tomoyuki Hasada　**松井俊浩** † Toshihiro Matsui　**松尾啓志** † Hiroshi Matsuo

1. はじめに

複数の，自律的に振る舞う要素である「エージェント」が協調して仕事を達成したり問題を解決するシステムをマルチエージェントシステムと呼ぶ．マルチエージェントシステムはセンサネットワーク，コラボレーション支援，スマートグリッドにおける電力資源割り当てなどへの応用が期待される．このようなシステムではエージェントが分散して存在しているため，各エージェントは他のエージェントがどの程度の能力を持ち，どのような情報を保持しているかが分からない．そのような状況のもとで問題を解決するためには，分散アルゴリズムも考慮しつつ問題解決手法を検討する必要がある．

マルチエージェントシステムにおける協調問題を定式化する，分散制約最適化問題 (Distributed Constraint Optimization Problem: DCOP)[1] が研究されている．DCOP は，全てのエージェントについての評価関数値の総和が最大または最小となるような変数値の割り当てを求める最適化問題である．この問題の解法の一つに確率的な山登り法に基づく DSA[2] が提案されている．DCOP では一般的に離散値を扱う．しかし，実際のシステムでは連続値を扱う必要がある場合が多い．したがって，連続変数の最適化を伴うシステムを定式化出来るように DCOP の枠組みを拡張することは重要である．

その一方で，連続変数を扱う最適化問題として，Network Utility Maximization (NUM) Problem[3][4] が研究されている．NUM Problem は，ネットワーク上に送信ノードであるソースが複数存在し，各リンクの容量制約を満たしつつ各ソースの効用の和を最大化する問題である．この問題の解法の 1 つとして分散ニュートン法 [4] が研究されている．

本論文では連続変数に拡張された DCOP のうち，NUM Problem に帰着できる問題に注目し，NUM Problem のための分散ニュートン法に基づく解法と，DSA を拡張した解法を提案する．NUM Problem はネットワーク上のリソースを配分する問題であるため，これと対応する DCOP においては，隣接ノード間での資源配分を調整する問題のある範囲を表現すると解釈できる．

本論文の構成を以下に示す．まず，2 章で本研究の背景として DCOP と NUM Problem について述べ，それぞれの問題に対する解法として DSA および分散ニュートン法の概略を示す．3 章では，上記の連続変数の DCOP を分散ニュートン法に基づいた解法で解く方法と，同様の問題を解くために DSA を拡張する方法について説明する．4 章では，2 つの提案手法を実験により比較し，評価する．最後に 5 章で本研究のまとめと将来的な展望を示す．

†名古屋工業大学, Nagoya Institute of Technology

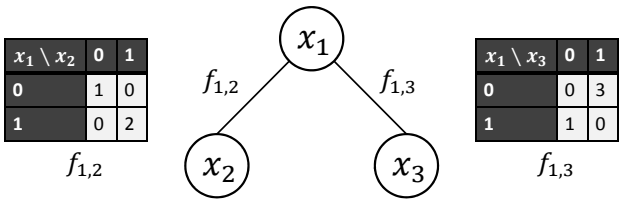

図 1: 分散制約最適化問題

2. 研究の背景

2.1. 分散制約最適化問題

分散制約最適化問題 (Distributed Constraint Optimization Problem: DCOP)[1] は，問題の構成要素が複数のエージェントに分散して配置された，離散組み合わせ最適化問題である．DCOP は，エージェントの集合 A，変数の集合 X，各変数の値域 D，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．各エージェント A_i は変数 $x_i \in X$ と値域 $D_i \in D$ を持ち，変数は各エージェントの状態を表す．エージェント間の関係は制約と評価関数により表される．各制約 $c_{i,j} \in C$ について，評価関数 $f_{i,j}(x_i, x_j) \in F$ により変数値の割り当てに対する評価値が定義される．DCOP の解は，全ての制約についての評価値の和を最大化するような変数値の割り当てであり，次の式で表される．

$$\text{maximize} \quad \sum_{f_{i,j} \in F} f_{i,j}(x_i, x_j) \tag{1}$$

DCOP の例を図 1 に示す．例では 3 つの変数と 2 つの制約から問題が構成されており，$(x_1, x_2, x_3) = (0, 0, 1)$ のとき評価値の和が 4 となり最大となる．

DCOP では，エージェントが分散して配置されているため，各エージェントはメッセージ交換を伴う分散アルゴリズムを用いて最適解を求める．DCOP の解法には，厳密解法 [1][5][6] と非厳密解法 [2][7][8] がある．厳密解法では必ず最適解を求めることが出来るが，計算量やメッセージサイズが指数的に増加するため，大規模な問題に対して適用すると現実的な時間で終了しない可能性がある．一方，非厳密解法では最適解を求められるとは限らないが，厳密解法に比べて計算量やメッセージサイズが少ないため，大規模な問題に対しても適用出来る．

一般に，DCOP は離散最適化問題を扱うが，実際的な分散システムでは連続値を扱うことも少なくない．そ

Algorithm 1 DSA における各エージェントの動作

1: ランダムに変数値を決定する
2: **while** 終了条件を満たしていない **do**
3: **if** 変数値が変更された **then**
4: 隣接ノードに変更後の変数値を送信する
5: **end if**
6: 隣接ノードから新しい変数値の情報を受け取る
7: 次の変数値を決定する
8: **end while**

のため，DCOP のような分散最適化の枠組みで連続値を扱うことは有用である．しかし，そのような問題と解法の研究は多くは無いため，連続値を扱う場合の解法に対する研究は重要である．

2.2.DSA

本研究では，確率的な山登り法に基づく DCOP の解法 DSA[2] に注目する．DSA は評価関数から求められる局所的な評価値と改善量 Δ を基に各エージェントが変数の値を変更する確率的アルゴリズムである．各エージェントは自身が保持する情報と隣接エージェントの状態に関する情報のみを利用しているため，局所最適解に収束しやすいが，通信コストを低く抑えることが出来るため，比較的大規模なシステムに適している．

DSA の動作を Algorithm 1 に示す．

DSA では始めに各エージェントがランダムに変数値を決定する．その後，終了条件を満たすまで次の 2 つのステップを繰り返す．

〈**ステップ 1**〉

各エージェントは自身の変数値を変更した場合，変更後の変数値を隣接ノードに対して送信する．(3~5 行目)

〈**ステップ 2**〉

隣接ノードに変数値を変更したノードがいる場合，変更後の変数値を受け取り，その情報を用いて自身が関係する評価関数の評価値の和の最大値とその時の変数値 v を求める．このとき，1 つ前の評価値の和との差を改善量 Δ とし，$\Delta > 0$ のときは確率 p で変数値を v に変更する．(6~7 行目)

DSA では各エージェントが取り得る値を列挙し，その中から次の変数値を決定する．しかし，連続変数を扱う場合，取り得る値は無限に存在するため，取り得る値を列挙する方法では次の変数値を決定することは困難である．そのため連続値への対応が課題となる．DSA はごく簡単なアルゴリズムであるため，連続変数への拡張を検討するためには適当であると考えられる．3.2 節では，NUM Problem に帰着可能な DCOP のための拡張された DSA を示す．

2.3.Network Utility Maximization Problem

Network Utility Maximization (NUM) Problem[3][4] は，ネットワーク上に存在する送信ノードが情報を送信する量を調整する問題である．

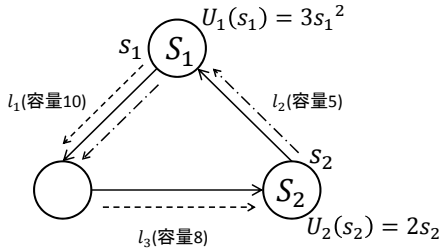

図 2: Network Utility Maximization Problem

NUM Problem では，ノードの集合 $\mathcal{N} = \{1, ..., N\}$ と有向リンクの集合 $\mathcal{L} = \{1, ..., L\}$ によってネットワークを表す．このネットワークは強連結グラフであり，各リンクは有限容量 $\boldsymbol{c} = (c_1, ..., c_L)$ を持つ．ネットワーク内のノードには送信ノード（以下，ソースと呼ぶ）も存在し，ソースの集合は $\mathcal{S} = \{1, ..., S\}$ で表される．各ソースはあらかじめ決められた分岐のない経路に沿って情報を送信する．各ソースが送信している情報の量はベクトル $\boldsymbol{s} = (s_1, ..., s_S)$ により表される．s_i に対する効用を表す関数として，効用関数 $U_i(s_i) : \mathbb{R}_+ \to \mathbb{R}$ が与えられる．各ソースが情報を送信する経路は経路行列 $\boldsymbol{R}$ により表され，各要素は以下のように与えられる．

$$R_{ij} = \begin{cases} 1 & : \text{リンク } i \text{ がソース } j \text{ の経路上にある} \\ 0 & : \text{その他} \end{cases} \tag{2}$$

各リンク l について，リンク l を使用しているソースの集合を $S(l)$ とし，各ソース i について，i が使用しているリンクの集合を $L(i)$ とする．このとき，各リンクには容量があるため，各リンク l は $\sum_{i \in S(l)} s_i \leq c_l$ を満たさなければならない．この式を全てのリンクについてまとめると $\boldsymbol{Rs} \leq \boldsymbol{c}$ となる．

また，全てのソースは情報を送信しているため，全てのソース i について，$s_i \geq 0$ を満たさなければならない．このことを $\boldsymbol{s} \geq 0$ と表す．ただし，$s_i = 0$ のときは，ソース i は情報を送信していないのと同義である．

NUM Problem では上に示した 2 つの制約を満たしつつ，各ソースが持つ効用関数の和を最大化することが目的となる．NUM Problem を定式化すると以下のようになる．

$$\text{maximize} \quad \sum_{i=1}^{S} U_i(s_i) \tag{3}$$

$$\text{subject to} \quad \boldsymbol{Rs} \leq \boldsymbol{c}, \tag{4}$$

$$\boldsymbol{s} \geq 0. \tag{5}$$

NUM Problem の例を図 2 に示す．例では，S_1 は $l_1 \to l_3$，S_2 は $l_2 \to l_1$ を経由して情報を送信している．このとき，どちらのソースも l_1 を経由しているため，容量制約から，$s_1 + s_2 \leq 10$ を満たさなければならない．同様に，l_2, l_3 についても，$s_2 \leq 5, s_1 \leq 8$ を満たさなければならない．これらの条件を満たしつつ $U_1(s_1) + U_2(s_2)$ が最大となるように s_1, s_2 の値を決定する．

一般に制約付き最適化問題では制約条件が不等式である問題よりも等式である問題のほうが容易に問題を解くことが出来る．文献 [4] に従えば，制約 (4),(5) は次のように等式制約に変形される．

$$\text{minimize} \quad -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i) \quad (6)$$

$$\text{subject to} \quad A\boldsymbol{x} = \boldsymbol{c} \quad (7)$$

ここで，$\boldsymbol{x} = (s_1, ..., s_S, y_1, ..., y_L)$ は新しい決定変数であり $\boldsymbol{y} = (y_1, ..., y_L)$ はスラック変数である．また，$A = [\boldsymbol{R} \ I(L)]$ であり，μ は 1 以上の定係数である．

以降では，

$$f(\boldsymbol{x}) = -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i) \quad (8)$$

とする．

2.4. 分散ニュートン法

2.4.1. 効用関数が満たすべき条件

NUM Problem では目的関数を最小化することが目的となる．しかし，ニュートン法ではヘッセ行列が正定値であることが保証されていない場合，解が下降方向に移動しない可能性がある．そこで，ヘッセ行列の正定値性を保証するために，効用関数に (a) 狭義凹関数，(b) 単調増加，(c)2 階微分可能の三つの条件を設定する．

また，ニュートン法の特徴である二次収束性を得るために，効用関数に (d)*self-concordant* であるという条件を設定する．ここで，ある実数の凹関数 $f : \mathbb{R} \to \mathbb{R}$ が，定義域内の全ての x について $|f'''(x)| \leq 2f''(x)^{\frac{3}{2}}$ を満たすとき f は *self-concordant* であるという．

以下，本論文中では特に断らない限り，効用関数は上の (a)∼(d) の条件を全て満たすものとする．

2.4.2. 初期解の設定

分散ニュートン法では現在の解を用いて次の解の計算を行うため，現在の解は実行可能でなければならない．そのため，初期解には実行可能な解を設定する．初期解 $\boldsymbol{x}^0$ の設定例を以下に示す．

例:

$$x_i^0 = \frac{\min_k\{c_k\}}{S+1} \quad \text{for } i = 1, 2, ..., S, \quad (9)$$

$$x_{i+S}^0 = c_i - \sum_{j=1}^{S} R_{ij} \frac{\min_k\{c_k\}}{S+1} \quad \text{for } i = 1, 2, ..., L \quad (10)$$

以下，k ステップ後の解を $\boldsymbol{x}^k$ と表す．

2.4.3. 解の更新

初期解を設定したら，以下の更新式を用いて解の更新を行う．

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + s^k \Delta \boldsymbol{x}^k \quad (11)$$

ここで，s^k はステップサイズ，$\Delta \boldsymbol{x}^k$ はニュートン方向である．$\Delta \boldsymbol{x}^k$ は以下の式を解くことで求められる．

$$\begin{pmatrix} \nabla^2 f(\boldsymbol{x}^k) & A^{\mathrm{T}} \\ A & O \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{x}^k \\ \boldsymbol{w}^k \end{pmatrix} = \begin{pmatrix} \nabla f(\boldsymbol{x}^k) \\ O \end{pmatrix} \quad (12)$$

以下，$H_k = \nabla^2 f(\boldsymbol{x}^k)$ とする．また，ベクトル $\boldsymbol{w}^k = (w_1^k, ..., w_L^k)$ は容量制約に対する双対変数である．式 (12) を $\Delta \boldsymbol{x}^k$ と $\boldsymbol{w}^k$ について解くと，

$$\Delta \boldsymbol{x}^k = -H_k^{-1}(\nabla f(\boldsymbol{x}^k) + A' \boldsymbol{w}^k) \quad (13)$$

$$(AH_k^{-1}A^{\mathrm{T}})\boldsymbol{w}^k = -AH_k^{-1}\nabla f(\boldsymbol{x}^k) \quad (14)$$

となる．$f(\boldsymbol{x})$ は x_i で分割可能であるため，H_k^{-1} は対角行列となる．したがって，ベクトル $\boldsymbol{w}^k$ が与えられれば，ニュートン方向 Δx^k を局所情報のみで計算出来る．しかし，$(AH_k^{-1}A^{\mathrm{T}})$ の逆行列を求めるには大域的な情報が必要となるため，$\boldsymbol{w}^k$ を計算出来ない．そこで，イテラティブな方法に基づいた方法で $\boldsymbol{w}^k$ を求める．

・双対変数の計算

行列分割を用いることで双対変数の値を計算する [4]．D_k を対角行列とし，対角要素を $(D_k)_{ll} = (AH_k^{-1}A^{\mathrm{T}})_{ll}$ とする．また，$B_k = AH_k^{-1}A^{\mathrm{T}} - D_k$ とし，$\bar{B}_k$ を対角要素が $(\bar{B}_k)_{ii} = \sum_{j=1}^{L}(B_k)_{ij}$ であるような対角行列とすると，$AH_k^{-1}A^{\mathrm{T}}$ は $(D_k + \bar{B}_k) + (B_k - \bar{B}_k)$ に分割出来る．このとき，任意の初期ベクトルを $\boldsymbol{w}(0)$ とすると，

$$\begin{aligned} \boldsymbol{w}(t+1) &= (D_k + \bar{B}_k)^{-1}(-AH_k^{-1}\nabla f(\boldsymbol{x}^k)) \\ &\quad -(D_k + \bar{B}_k)^{-1}(B_k - \bar{B}_k)\boldsymbol{w}(t) \end{aligned} \quad (15)$$

は $t \to \infty$ のとき，式 (14) の解に収束する．そこで，双対変数の近似値を式 (15) を用いて計算し，その値を用いる．

双対変数を計算する時の情報交換手続きについて述べる．任意の双対変数のベクトル $\boldsymbol{w}(0)$ が与えられたとき，$\boldsymbol{w}(t)$ は次の 2 つのステップにより求められる．

〈 **ステップ 1**〉

各ソース i は使用しているリンク $l \in L(i)$ に対して勾配 $\nabla_i f(x^k)$，ヘッセ行列 H_{ii}^k，経路情報 $(A_{ij})_{j=i...L}$，双対変数のベクトル $(w_l(t))_{l \in L(i)}$ を送信する．

〈 **ステップ 2**〉

各リンク l は $(D_k)_{ll}$，B_k の l 行目，$(\bar{B}_k)_{ll}$ を計算し，式 (15) を用いて $w_l(t)$ を更新する．その後，リンク l を使用しているソースに対して，更新した双対変数 $w_l(t+1)$ を送信する．

・ニュートン方向の計算

双対変数の値が得られたら，式 (13) を用いてニュートン方向 $\Delta \boldsymbol{x}^k$ を計算する．しかし，双対変数の値が正確ではないため，制約 (7) を満たさない可能性がある．そこで，実行可能性を維持するために，$\Delta \boldsymbol{x}^k$ の代わりに $\Delta \tilde{\boldsymbol{x}}^k$ を次の 2 つのステップに分けて計算する．

〈**ステップ 1**〉

式 (15) で得られた $\boldsymbol{w}^k$ を用いて，$\Delta\tilde{\boldsymbol{x}}^k$ の最初の S 個の要素を式 (13) を用いて計算する．

〈**ステップ 2**〉

スラック変数に当たる $\Delta\tilde{\boldsymbol{x}}^k$ の最後の L 個の要素を，$A\Delta\tilde{\boldsymbol{x}}^k = 0$ を満たすように計算する．

$\Delta\tilde{\boldsymbol{x}}^k$ の計算を行ったら式 (11) の代わりに次の式を用いて解の更新を行う．

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + s^k\Delta\tilde{\boldsymbol{x}}^k \tag{16}$$

2.4.4. ステップサイズの計算

ニュートン減少値 $\tilde{\lambda}(\boldsymbol{x}^k)$ を次のように定義する．

$$\tilde{\lambda}(\boldsymbol{x}^k) = \sqrt{(\Delta\tilde{\boldsymbol{x}}^k)^{\mathrm{T}}\nabla^2 f(\boldsymbol{x}^k)\Delta\tilde{\boldsymbol{x}}^k} \tag{17}$$

$\tilde{\lambda}(\boldsymbol{x}^k)$ を求めるには大域的な情報が必要であるため，計算出来ない．そこで，合意に基づいた反復平均化アルゴリズム [9] により $\tilde{\lambda}(\boldsymbol{x}^k)$ の計算を行う．平均化アルゴリズムにより得られた $\tilde{\lambda}(\boldsymbol{x}^k)$ の値を θ^k とすると，ステップサイズ s^k は以下の式で与えられる．

$$s^k = \begin{cases} \frac{c}{\theta^k+1} & \text{if} \quad \theta^k \geq \frac{1}{4} \\ 1 & \text{otherwise} \end{cases} \tag{18}$$

ここで，c は $\dfrac{5}{6} < c < 1$ を満たす正の数である．

3. 提案手法

本研究では，連続変数の DCOP のうち，NUM Problem に帰着できる問題を解くための手法を二つ提案する．第一の手法では，連続変数の DCOP を NUM Problem に帰着し，その問題を分散ニュートン法で解く．第二の手法では，DSA を改良し，同様に連続変数の DCOP を解く．

3.1. 連続変数の DCOP から NUM Problem への帰着

NUM Problem はネットワーク上のリソースを配分する問題である．したがって，これと対応する DCOP においては，隣接ノード間での資源配分を調整する問題のある範囲を表現すると解釈できる．問題の帰着を行うにはいくつかの条件を満たしている必要がある．この条件について説明した後，問題の帰着について説明する．

3.1.1. 問題を帰着するために必要な条件

問題の帰着を行うために必要な条件は以下の 3 つである．

1. 各制約辺が計算能力を持っている
2. 各評価関数が単項関数に分割可能，すなわち $f_{i,j}(x_i, x_j) = g_{i,j}(x_i) + g_{j,i}(x_j)$ であり、各単項関数が 2.4.1 で示した効用関数の条件を満たしている
3. 各制約は $x_i + x_j \leq c_{i,j}$ の形で表すことが出来る。ただし、$c_{i,j}$ は正の定数である

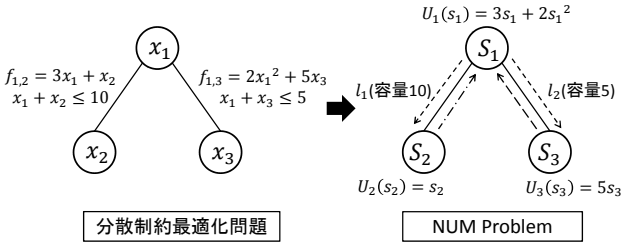

図 3: DCOP から NUM Problem への問題の帰着

1 つ目の条件は，NUM Problem を分散ニュートン法で解く際に各リンクも計算を行うため必要となる．2 つ目の条件は，NUM Problem では単項関数を扱うため必要となる．3 つ目の条件は，各制約を NUM Problem における容量制約に合わせるために必要である．

なお，本論文では記述を簡潔にするために，必要に応じて $c_{i,j}$ を容量，DCOP における制約を容量制約と表記する．

3.1.2. 問題を帰着する際のパラメータの対応関係

上で示した条件を満たしているとき，次のように問題を帰着する．

- DCOP におけるエージェント A と変数 X をソース $\mathcal{S}$ と情報量 $\boldsymbol{s}$ に，制約辺を双方向のリンク $\mathcal{L}$ に置き換える
- 各ソース i の効用関数は $\sum_{j \in Nbr(i)} g_{i,j}(x_i)$ とする
  ただし，$Nbr(i)$ はソース i の隣接ノードの集合を表す．
- 各ソースは全ての隣接ノードに情報を送信する
- 各リンクに容量 $c_{i,j}$ を持たせる

問題の帰着例を図 3 に示す．例では 3 つの変数と 2 つの制約で構成された DCOP を NUM Problem に帰着する様子を示している．DCOP における各ノード x_i は NUM Problem ではソース S_i となり，隣接するノードに対して情報を送信する．また，変数 x_i は情報量 s_i に置き換え，制約 $f_{i,j}$ は効用関数 $U_i(s_i)$ に置き換える．このとき，ソース S_i の効用関数 $U_i(s_i)$ は制約 $f_{i,j}$ から x_i を含む項を全て取り出し，足し合わせたものである．例えば，ソース S_1 の効用関数 $U_1(s_1)$ に注目してみると，制約 $f_{1,2}$ の $3x_1$ の項と制約 $f_{1,3}$ の $2x_1^2$ の項を足し合わせたものとなるので，$U_1(s_1) = 3s_1 + 2s_1^2$ となる．各制約は右辺の定数項の値がそのままリンクの容量となる．そのため，リンク l_1, l_2 の容量はそれぞれ 10,5 となる．

3.1.3. 分散ニュートン法をそのまま適用する場合

問題の帰着を行ったら，その問題に対して分散ニュートン法を実行することで各ソース S_i の情報量 s_i を求めることができる．s_i を求めたら，その値をそのまま

DCOP の変数 x_i に割り当てる．このときの変数値の割り当てが DCOP の解となる．

ここで，DCOP を帰着して生成された NUM Problem は，一般的な NUM Problem と違い，各リンクを使用するソースが必ず 2 つになるという性質がある．この性質を利用することで，既存の分散ニュートン法に比べて初期解の値を大きくでき，解が収束するまでの時間を短縮できる．

また，提案手法を適用するためにさまざまな条件を設定したため，提案手法を適用できる問題の範囲が狭くなっている．そこで，分散ニュートン法における計算を改良することで，使用できる関数を拡張する．

3.1.4. 初期解の改善

本手法により問題を帰着した場合，各リンクは必ず 2 つのソースにのみ使用されることになる．この性質を利用して初期解を改善することで，解が収束するまでの時間の短縮を図る．一般的な問題を解く場合は，どのリンクも全てのソースに使用される可能性があったため，初期解は式 (9),(10) のようにする必要があった．しかし，今回の場合は必ず 2 つのソースにのみ使用されるため，初期解を以下のように変更出来る．

$$x_i^0 = \frac{\min_{k\in L(i)}\{c_k\}}{2} - \varepsilon \quad \text{for } i = 1, 2, ..., S, \tag{19}$$

$$x_{i+S}^0 = c_i - \sum_{j=1}^{S} R_{ij}\, x_j^0 \quad \text{for } i = 1, 2, ..., L \tag{20}$$

ここで，ε は十分小さい正の定数である．ε を付けない場合，スラック変数の値が 0 になり実行不可能な状態になる可能性があるため，ε の項を追加している．

この変更により初期解が向上するため，解の収束までの時間が短縮される．

3.1.5. 使用可能な関数の拡張

分散ニュートン法では 2.4.1 で示した条件を満たした関数しか扱うことが出来ない．ニュートン法では，ヘッセ行列が正定値でない場合に探索方向が必ずしも降下方向になるとは限らないため，このような条件が付けられている．これに対し，ヘッセ行列が正定値でない場合にヘッセ行列を正定値になるように修正する，修正ニュートン法が存在する．修正ニュートン法では次のようにヘッセ行列を修正する．

$$B_k = H_k + \gamma I \tag{21}$$

ここで，I は単位行列であり，$\gamma \geq 0$ である．γ が十分大きな値を取ると，B_k は正定値となるので，H_k の代わりに B_k 用いてニュートン法を行うことで，探索方向が下降方向となる．

本研究では，修正ニュートン法のアイデアを用いることで，2.4.1 で示した条件を緩和する．分散ニュートン法では $f(\boldsymbol{x})$ は x_i で分割可能であるため，H_k は対角行列となっている．したがって，H_k の対角要素の内，最も小さい値を d とすると，γ の値は $\gamma > -d$ となる．しかし，γ を求めるには大域的な情報が必要となり，通信コストが増加する．そこで，次のようにヘッセ行列を修正する．

$$\begin{pmatrix} -5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

H_k B_k

図 4: ヘッセ行列の修正

$$(B_k)_{ii} = \begin{cases} (H_k)_{ii} & (H_k)_{ii} > 0 \\ -(H_k)_{ii} & (H_k)_{ii} < 0 \\ 1 & (H_k)_{ii} = 0 \end{cases} \tag{22}$$

ヘッセ行列の修正例を図 4 に示す．

このように修正することで，関数が狭義凹関数でなくても B_k は正定値となるため，分散ニュートン法を実行出来る．また，局所情報のみを用いているため，通信コストは変わらない．しかし，修正ニュートン方と違い，各要素毎に異なる γ の値を用いているため，解の精度が低くなる可能性がある．また，2.4.1 で示した条件の内，*self-concordant* であるという条件を満たさない関数も扱うようになるため，収束速度も低下する可能性がある．

3.2. 連続変数の DCOP を解くための DSA の改良

DSA で連続変数を扱う場合，各エージェントの変数が取り得る値が無限に存在するため，従来の離散値を列挙する方法では，どの変数値を選択するかを容易に決められない．ここでは前述の NUM Problem に帰着できる DCOP を対象として，この問題のために拡張された DSA を示す．

3.2.1.DSA を動作させるために DCOP が満たすべき条件

DCOP に設定する条件はと同じである．ただし，評価関数を単項関数に分割した際の単項関数が満たすべき条件のみ異なり，単調増加であるだけでよい．

このような問題では，どのエージェントも変数値を大きくするほど評価値が大きくなる．また，制約の形式から隣接するエージェントの変数値の和がある定数以下となるので，各エージェントが選ぶことのできる変数値の最大値は (辺の容量 – 隣接エージェントの変数値) となる．これら 2 つの性質から，各エージェントは自身が取り得る値の中から最も評価値の改善量が大きくなる値を求めることができるため，DSA を動作させることができる．

3.2.2.DSA が動作する領域

DSA では，隣接ノードから得られた情報を用いて，自身が関係する評価関数の和が最も大きくなるように確率的に変数値を変更する．このような動作をすると，場合によっては実行不可能な領域の値を選択するエージェントが存在する．特に本研究で扱う問題では各エージェントは可能な限り変数値を大きくしようとするため，隣接するエージェントが同時に変数値を変更したときに，大きく制約を違反し，著しく性能が低下する

恐れがある．その一方で，分散ニュートン法では実行可能な領域で動作するため，違反を起こすことがない．そこで，本研究では性能の低下を抑えるために，DSA が分散ニュートン法と同様に実行可能な領域で動作するようにした．

3.2.3. 初期値の設定

DSA では始めにランダムに初期値を決定する．しかし，完全にランダムに決定すると実行不可能な領域の値をとる可能性がある．そこで，初期値は次の方法で設定する．

1. 各エージェントは自身が関係する制約の中で最も小さい $c_{i,j}$ の値 $min(c_{i,j})$ を調べる
2. 各エージェントは $0 \sim \frac{min(c_{i,j})}{2}$ の範囲でランダムに値を選択し，その値を初期値として設定する

このように設定することで，各エージェントは実行可能な領域の範囲内でランダムに初期値を選択できる．

3.2.4. 変数値の変更規則

3.2.2 で述べたとおり，DSA では隣接ノードから得られた情報を用いて，自身が関係する評価関数の和が最も大きくなるように確率的に変数値を変更する．しかし，このような方法で変数値を決定すると隣接するノードが同時に変数値を変更したときに実行不可能となる．そこで，実行可能な状態を維持するために，変数値の変更規則を次のようにした．

1. 各ノードは自身に接続されている全ての制約辺について，もっとも小さい空き容量の値 d を調べる
2. 各ノードは $0 \sim \frac{d}{2}$ の範囲でランダムに値を選択し，その値の分だけ変数値を増やす

この変更規則では，全てのエージェントが毎回必ず変数値を変更するようにしている．しかし，全てのエージェントが同時に変数値を最大である $\frac{d}{2}$ 増やした場合でも，制約違反が起きないようになっている．そのため，上記のように変数値の変更規則を変更した DSA は，実行可能な領域で動作するようになる．

4. 評価

2 つの提案手法について，1 台の計算機上でシミュレーションによる実験を行い，2 つの手法の解の精度および収束するまでの解の更新回数を比較した．使用した問題はエージェント数を 50，制約数を 100，各制約辺の容量の範囲を [5,10] と設定し，グラフは各ノードを制約辺によりランダムに接続することにより作成した．各制約の評価関数は表 1 の中から選択し，n の値の範囲は [1,5],[5,10] とした．ここで，表中の a の値は [0.5,1.5]，b の値は [10,20] であり，表中の (a) は $n = 1$ のとき線形関数，$n > 2$ のとき [0,b] で狭義凸関数となっており，(b),(c) は [0,b] で狭義凹関数となっている．また，問題に含まれる評価関数のうち，狭義凹関数と狭義凸関数の比率を 90:10,50:50,10:90 とした．ただし，狭義凹関数は表 1 の (b)(c) からランダムに選択した．

表 1: 使用した関数

	関数
(a)	ax^n
(b)	$-ax^n + 4ab^2x^{n-2}$ $(n > 2$ かつ n が奇数$)$
(c)	$-ax^n + 2abx^{n-1}$ $(n > 2$ かつ n が偶数$)$

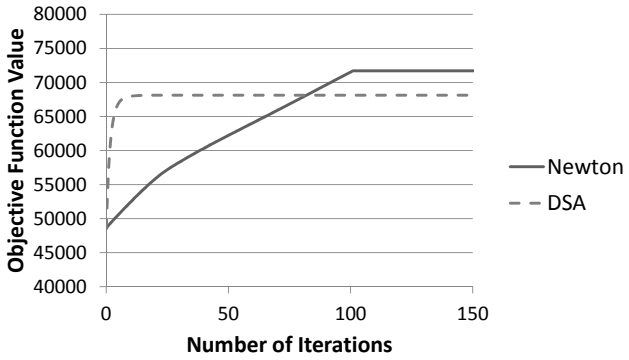

図 5: 比較的容易に解ける問題 (関数の次数 [1,5]，関数の比率 90:10)

本実験では DSA の初期値を分散ニュートン法と同じ値とした，また，分散ニュートン法は解の変化量が 10^{-8} 以下になったときに収束したものとし，動作を停止させた．DSA は各問題を 1000 回ずつ解き，得られた値の平均値を結果としている．それぞれの手法において，メッセージ交換における通信の遅延は考慮せず，通信中にメッセージの欠損はないものとした．

本実験ではイテレーション回数を解の更新回数として表し，全てのエージェントが解を更新する毎にイテレーション回数を 1 増加する．

4.1. 比較的容易に解ける問題

各評価関数の次数の範囲が [1,5] の問題に対してそれぞれの手法を動作させ，性能を比較した．この問題には比較的次数の低い関数のみが含まれている．ニュートン法では，評価関数を二次関数に近似して問題を解いてゆくため，二次関数に近い関数ほど収束が速くなる．この問題では，次数が 2 に近い関数が多いため，比較的速く解を求めることができる．

結果を図 5 ～ 7 に示す．各図の横軸はイテレーション回数，縦軸は評価値の総和を表している．

図 5,6 では DSA よりも分散ニュートン法のほうが評価値が高くなっており，図 7 では DSA のほうが評価値が高くなった．このことから，既存の分散ニュートン法では扱うことが出来なかった狭義凸関数や線形関数が含まれている場合，これらの関数の割合が多くなるほど分散ニュートン法の解の精度が悪くなると考えられる．よって，そのような関数が多い場合には分散ニュートン法よりも DSA を用いるほうが良いと考えられる．

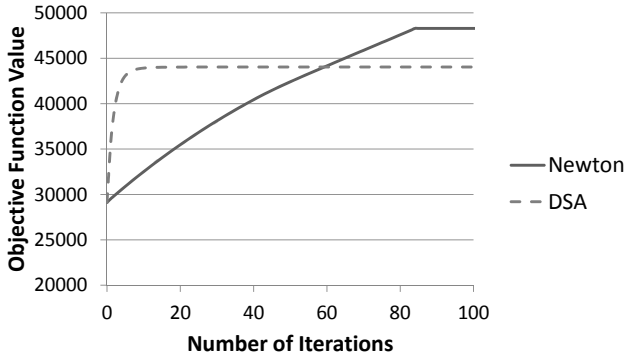

図 6: 比較的容易に解ける問題 (関数の次数 [1,5]，関数の比率 50:50)

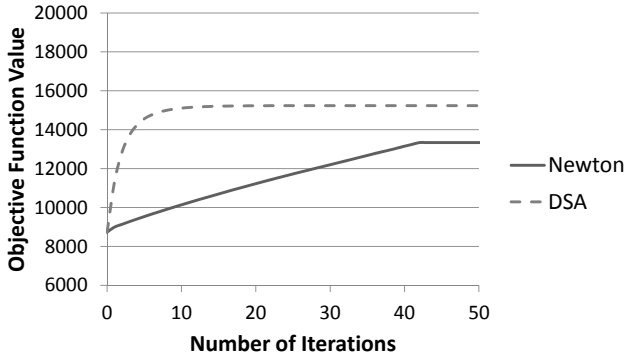

図 7: 比較的容易に解ける問題 (関数の次数 [1,5]，関数の比率 10:90)

また，分散ニュートン法では収束するまでのイテレーション回数が 50~100 回程度であり，DSA に比べると多いが，比較的少ない回数で収束した．

4.2. 収束までに時間を要する問題

各評価関数の次数の範囲が [5,10] の問題に対してそれぞれの手法を動作させ，性能を比較した．この問題では，次数の高い関数のみを扱うため，収束に時間がかかる．

結果を図 8 ~ 10 に示す．

図 8 と 10 では分散ニュートン法より DSA のほうが評価値が高くなっており，図 9 では分散ニュートン法のほうが評価値が高くなっている．分散ニュートン法は局所最適解を求めるアルゴリズムであるため，問題によって良い解に収束する場合や悪い解に収束する場合がある．そのため，結果にばらつきが生じたと考えられる．実際に，それぞれの問題について 10 個の異なる問題を用意し 2 つの手法を動作させたところ，それぞれ 5~8 個の問題において分散ニュートン法の評価値が高くなり，問題によって結果にばらつきが生じていることが分かる．しかし，全体的に分散ニュートン法のほうが評価値が高くなる割合が高いため，分散ニュートン法のほうが解の精度が良いと考えられる．ただし，次数の範囲が [1,5] のときと比べてイテレーション回数が増加しているため，収束時間を許容できない場合には DSA を用いると良い．

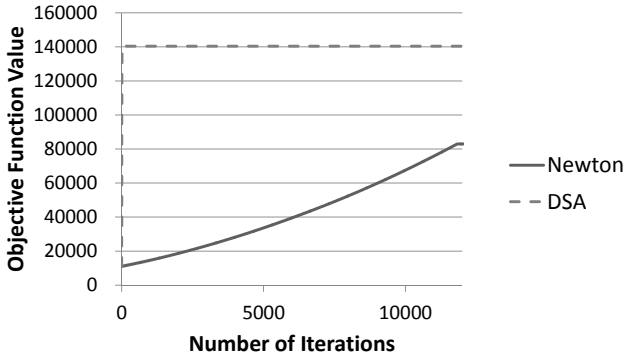

図 8: 収束までに時間を要する問題 (関数の次数 [5,10]，関数の比率 90:10)

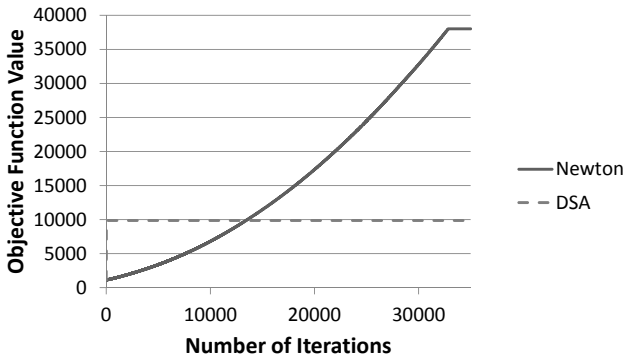

図 9: 収束までに時間を要する問題 (関数の次数 [5,10]，関数の比率 50:50)

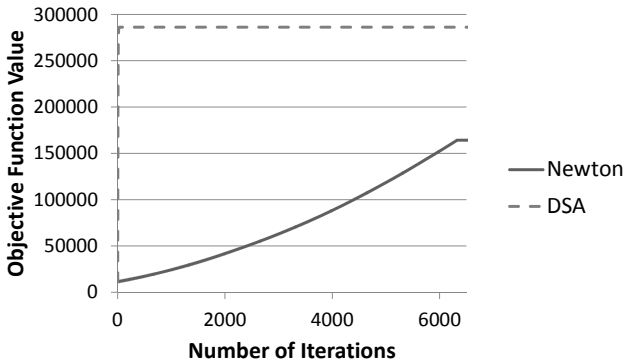

図 10: 収束までに時間を要する問題 (関数の次数 [5,10]，関数の比率 10:90)

5. まとめ

本研究では，連続変数に拡張された分散制約最適化問題のうち，Network Utility Maximization Problem に帰着できる問題を解く手法を提案した．まず，連続変数の分散制約最適化問題を Network Utility Maximization Problem に帰着し，分散ニュートン法で解く手法を示した．この際，分散制約最適化問題に (a) 各制約辺が計算能力を持っている，(b) 各評価関数が単項関数に分割可能，(c) 各制約は $x_i + x_j \leq c_{i,j}$ の形で表すことが出来る，という 3 つの制約を加えた．また，既存の分散ニュートン法では扱えなかった一部の問題に対しても，ヘッセ行列を修正することで分散ニュートン法を動作させる手法を提案した．さらに，同様の問題を解くために DSA を拡張する方法を提案した．実験により 2 つの手法を比較することで，問題に応じてどちらの手法が有効であるかを示した．

今後の研究課題として，適用可能な問題を拡充するために，より適切な修正ニュートン法や，他の降下法の適用が挙げられる．異なる性質のネットワーク上での効果の評価も課題として挙げられる．また，既存手法 [10] との比較や DCOP の解法と数値計算を併用する手法の検討も今後の課題である．

謝辞

本研究の一部は，科研費基盤研究 (C)25330257 および平成 23 年度人工知能研究振興財団研究助成による．

参考文献

[1] Adrian Petcu and Boi Faltings. Dpop: A scalable method for multiagent constraint optimization. In *IJCAI 05*, pp. 266–271, Edinburgh, Scotland, Aug 2005.

[2] Weixiong Zhang, Ong Wang, and Lars Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *in PAS*, pp. 53–59, 2002.

[3] F. P. Kelley, A. Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of Operational Research Society*, No. 49, pp. 237–252, 1998.

[4] Ermin Wei, Asuman E. Ozdaglar, and Ali Jadbabaie. A distributed newton method for network utility maximization. In *CDC*, pp. 1816–1821, 2010.

[5] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.*, Vol. 161, No. 1-2, pp. 149–180, January 2005.

[6] Roger Mailler and Victor Lesser. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pp. 438–445. IEEE Computer Society, 2004.

[7] Rogers Alex Petcu Adrian Farinelli, Alessandro and N. R. Jennings. ecentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm. In *n, Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, 2008.

[8] H. Katagishi and J. P. Pearce. Distributed DCOP Algorithm for Arbitrary k-Optima with Monotonically Increasing Utility. *Ninth DCR Workshop (CP-07)*, 2007.

[9] Ali Jadbabaie, Jie Lin, and A. Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automat. Contr.*, Vol. 48, No. 6, pp. 988–1001, 2003.

[10] Thomas Voice, Ruben Stranders, Alex Rogers, and Nicholas R. Jennings. A hybrid continuous max-sum algorithm for decentralised coordination. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pp. 61–66, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.