情報処理学会論文誌 Vol. 50 No. 12 3136–3149 (Dec. 2009)

分散制約最適化問題近似解法の多重実行の効果

飯塚泰樹[†1] 竹内郁雄[†1]

分散制約最適化問題は，マルチエージェント環境における分散問題解法として注目を集めている．しかし提案されている解法は，最適解を保証するために非常に長い処理時間を要するか，あるいは確率的解法により短い時間で粗い近似解を得るのが限界であった．現実世界の複雑な問題への適用を考えた場合，短い時間で品質の高い近似解が得られる効率的手法が求められる．本論文では，分散制約最適化問題のための分散非決定性近似アルゴリズムの効率化を目指した多重化手法を提案する．分散アルゴリズムの計算時間の多くはメッセージ通信に費やされること，よって個々のノードの計算量やメッセージ長の少しの増加はアルゴリズム全体の効率に直接大きな影響を及ぼさないという前提のもとでは，分散近似アルゴリズムを多重実行することで効率を上げることが可能である．本論文ではその効果について極値分布理論を用いた理論的解析を行うとともに，シンプルなアルゴリズムの多重化実験により検証を行った．その結果，計算時間の大幅な短縮や解の品質の向上とともに，計算結果の分散を小さくできる効果を確認した．

A Multiplexed Method for Distributed Constraint Optimization Problems

Yasuki Iizuka[†1] and Ikuo Takeuchi[†1]

Distributed constraint optimization problems (DCOP) have been attracting attention as one of the effective approaches for modeling distributed reasoning tasks in distributed autonomous systems. However, most of existing approaches require very long computation time to guarantee the optimal solution, or can obtain only a roughly approximate solution quickly by a probabilistic method. For real world problems, however, we need a more efficient algorithm that can obtain high quality near-optimal solutions within reasonably short time. The authors propose a multiplexed method that improves the efficiency of distributed approximation algorithms under the assumption that most of the computation time of distributed algorithms are spent by message communication, and therefore small increase of the computation at individual nodes and the message lengths do not matter much to the efficiency of the whole computation. In this paper, we present a theoretical analysis to verify the effectiveness of the proposed method by using the theory of extreme value distribution. Then, by multiplexed experimentations based on a simple algorithm, we show the outstanding improvement of the solution quality and computation speed as well as the decrease of the variance of the obtained solutions.

1. はじめに

分散制約最適化問題（DCOP: Distributed Constraint Optimization Problem）は分散人工知能の基本的枠組みの 1 つとして，近年注目を集めている[1)–3)]．DCOP の解法として，いくつかの完全アルゴリズム[4)–6)]，あるいは近似アルゴリズム[7)–9)] が提案されている．しかし実世界の問題を扱うためには，効率の高いアルゴリズムが求められている[10)]．本研究は，分散近似アルゴリズムの効率を上げることを目指すものである．分散アルゴリズムの処理時間の多くは通信により費される．本論文では，個々のノードの計算量や通信メッセージ長の多少の増加はアルゴリズムの効率に直接大きな影響を及ぼさないという前提のもと，1 種類の分散近似アルゴリズムをアルゴリズムポートフォリオ[11)] 同様に多重実行することで，アルゴリズムの計算時間短縮と解の高品質化が可能であることを示す．

以下，本論文では 2 章で DCOP について概観する．3 章では DCOP のための多重化解法を提案するとともに，期待できる効果について極値分布理論を用いた理論的解析を行う．そして 4 章において例題を通した実験により提案した多重化解法の効果を検証し，5 章において考察を行う．

2. 分散制約最適化問題（DCOP）

分散制約最適化問題（Distributed Constraint Optimization Problem: DCOP）は次のように定義されている[4),6)]．変数の集合 $x_1, x_2, \ldots, x_n$，変数のそれぞれが値をとる有限で離散的な領域 $D_1, D_2, \ldots, D_n$ が与えられていて，それぞれの変数はエージェント $a_1, a_2, \ldots, a_m$ に割り当てられているものとする．一部の変数間，x_i と x_j の間には制約 $c_{ij} : D_i \times D_j \to \{true, false\}$ が与えられていて，各制約にはコスト関数 $g_{ij} : D_i \times D_j \to \mathbb{R}^+$ が対応している．

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

$$g_{ij}(x_i, x_j) = \begin{cases} v_{ij}^t, & \text{if } c_{ij}(x_i, x_j) = true \\ v_{ij}^f, & \text{otherwise} \end{cases}$$

ただし $v_{ij}^t < v_{ij}^f$ とする．エージェント a_k は自分が持つ変数 x_k，x_k に関係する制約 c_{ij}，およびコスト関数 g_{k*} に関する情報だけを持つ．このとき，DCOP の目的はコスト関数の総和 $G(\mathcal{A}) = \sum g_{ij}(\mathcal{A})$ を最小化する変数の割当て $\mathcal{A}$ を求めることである．

g_{ij} は変数間に制約 c_{ij} が定義されていないときは，$g_{ij} \to 0$ の恒等写像として扱う．本論文では各エージェントはただ 1 つの変数を持つことを仮定し，$n = m$ として扱う．

DCOP において，すべての割当て $\mathcal{A}$ の中で最小の $G(\mathcal{A}_o)$ を与える割当て $\mathcal{A}_o$ を最適解と呼ぶ．本論文では，ある割当て $\mathcal{A}_p$ について $l = G(\mathcal{A}_p) - G(\mathcal{A}_o)$ であるとき，$\mathcal{A}_p$ を最適解から距離 l の解と呼ぶことにする．距離 l は解の品質を判断する尺度の 1 つであり，本論文では l が小さい解のことを品質が高い解と呼ぶことにする．

DCOP では，変数が制約で結ばれたエージェント間で，変数の値をメッセージ通信で交換しながら問題を解く．

DCOP を解くための代表的なアルゴリズムとして，ADOPT[4]，DPOP[5]，OptAPO[6]，分散 Stochastic Search（DSA）[7]，DiSTaS-Anne[12] などがある．ADOPT は事前に制約グラフを深さ優先探索木（Depth First Search 木：DFS 木）に変換してから処理を進める完全アルゴリズムであるが，最適解からの最大距離 *bound* を指定して近似アルゴリズムとして動作させることも可能であるという特徴を持つ．ADOPT はエージェント数が増加すると処理時間は指数関数的に増加する傾向にあり[12]，処理が終了するまでに交換されるメッセージ数もこれにともなって増加する．DPOP も DFS 木を使った完全アルゴリズムである．メッセージ交換回数は DFS 木の深さ分しか必要ないため非常に少ないが，最大メッセージサイズは induced width[5] の指数オーダになり，これを処理するためにエージェントが必要とするメモリ量も大きくなる．OptAPO はメディエータの概念を用いる完全アルゴリズムであり，メディエータ役のエージェントが周辺のエージェントの制約違反を調整する．OptAPO ではメディエータ役のエージェントに処理が集中するという課題を持つ．最悪条件では全エージェントの調整を 1 つのメディエータエージェントが行うことになる．

DCOP を現実の問題，特にロボットやセンサネットワークの問題解決に使う場合，問題を分散環境の少ない計算資源で短時間のうちに解く必要がある[7),13)]．複雑な問題を表現するためには，多くの変数を必要とする．このように変数の数や制約の数が多い場合，完全なアルゴリズムによる最適解の探索は必ずしも最良の方法ではなく，高速で効率の良い近似解法が求められる．

分散 Stochastic Search アルゴリズム（DSA）は反復改善型の近似アルゴリズムである．アルゴリズムは確率的アプローチを採用しており，シンプルで，高並列性，低通信コストなどの特徴を持つ．また，処理時間に対して目的関数の評価値がおおよそ単調減少することから，任意時間アルゴリズム（Anytime Algorithm）[14] と分類できるが，アルゴリズムは停止機構を持たない．DSA のような近似アルゴリズムは，非常に短い実行時間で粗い近似解を得られるが，高い品質の解を得るためにはある程度長い実行時間をかける必要がある．

DiSTaS-Anne は同じく確率的アプローチと Tabu サーチの要素を採用した分散近似アルゴリズムであり，高い品質の解を短い時間で得ることを特徴とする．

本研究では，DSA や DiSTaS-Anne のような分散近似アルゴリズムの効率をさらに上げることを目指すものである．

3. 多重化手法の提案

本章では，分散近似アルゴリズムの多重化について提案をする．まず最初に本論文では，分散アルゴリズムとその多重化について以下の 3 つの前提をおくことにする．

［前提 1］多重化を行うときに，計算資源（CPU の数）を増やさない．

本研究における多重化は，計算資源は同じまま，いくつかの探索を同時に実行するものとして次節で提案する．分散制約最適化問題はもともと問題が分散されているため，複数のエージェントで計算を行うが，多重化は各エージェントごとの計算資源を増やすことを前提としない．探索空間を分割し，処理の高速化を目的として複数の計算資源で処理を行う並列処理との混乱を避けるため，本論文では “多重化” という用語を用いる．

［前提 2］分散近似アルゴリズムの計算成績には，ばらつきが発生する．

本論文では，計算にかかった時間や得られた解の品質などを計算成績と呼ぶことにする．乱数を用いた非決定性分散アルゴリズムの計算成績には，一般的にばらつきが発生する．ただし分散アルゴリズムの場合，アルゴリズムに乱数を用いていなくても，ノード間の通信遅延のゆらぎなどにより計算時間にはばらつきが発生する[15]．

［前提 3］分散アルゴリズムの計算時間の多くはメッセージ通信に費やされる[16]．

このため，分散アルゴリズムの評価尺度には一般的に，理想時間複雑度と通信複雑度が用いられている[17]．理想時間複雑度とは，各エージェントが同期的にメッセージ交換と処理を行い，アルゴリズムが停止した時点までの全体のメッセージ交換ラウンド数のことをいう．またメッセージ長が大きくない場合は，通信複雑度として，アルゴリズム終了までに交

```
1: Algorithm S
2: while(TerminationConditionIsNotMet)do
3:    mc := 0; /* Number of Messages */
4:    while(mc < #Neighbors) do
5:       neighborsStatus[mc] := Receive; mc++;
6:    endwhile
7:    x := NewValue(neighborsStatus[ ])
8:    Send(x)
9: endwhile
```

図 1　単純な分散反復改善アルゴリズム $\mathcal{S}$
Fig. 1　Simple distributed iterative repair algorithm $\mathcal{S}$.

換されるメッセージ総数を意味するメッセージ複雑度が適当とされている．これらの尺度を採用するという前提のもとでは，メッセージ数が増えなければ，各ノードの処理時間，および通信メッセージ長の少しの増加は分散アルゴリズムの実行時間に直接大きな影響を与えないといえる．本論文では，アルゴリズムの計算時間として理想時間複雑度を用いる．

3.1　分散近似アルゴリズムの多重化

ある試行 S の結果が何らかの確率分布に従うとき，この試行を繰り返し行い，複数試行の最小値（あるいは最大値）を選択した場合を試行 M とすると，試行 M は試行 S とは違う確率分布に従う．たとえばサイコロを投げて出た目を得点とする場合，この試行の確率分布は $X = \{1, 2, 3, 4, 5, 6\}$ が 1/6 ずつの一様分布になり，期待値 μ_s は 3.5 になる．サイコロを m 回，あるいは m 個のサイコロを同時に投げて，その出た目の最小値を得点とする試行 M は，一様分布にはならず，m が大きいほどその期待値 μ_m は 1 に近づく．

この原理を分散近似アルゴリズムに応用すれば，アルゴリズムの効率改善に役立つことが期待できる．すなわち，分散近似アルゴリズムを m 回実行してその最大値や最小値を結果として選択する，あるいは同時に m 重実行してその最大値や最小値を結果として選択すれば，より良い計算成績が得られるものと考えられる．

［前提 3］のもと，メッセージ数を増やすことなく，分散アルゴリズムを実行する各ノードでの計算とメッセージを多重化することを検討する．

図 1 は分散制約最適化問題のための単純な反復改善型分散近似アルゴリズム $\mathcal{S}$ の例である．アルゴリズムを実行するエージェントは近傍のエージェントからメッセージを受信し，自らの値 x を決定した後，これを近傍のエージェントに通知する．手続き NewValue( ) には非決定的要素を含むものと仮定する．本論文では，このような通常のアルゴリズムの実行の形態を 1 つの平面と呼ぶことにする．この平面を複数同時に実行することを，本論文では多重化と呼ぶ．図 1 のアルゴリズム $\mathcal{S}$ を多重化したアルゴリズム $Multiplexed\text{-}\mathcal{S}$ を**図 2** に示す．エージェントは m 個の平面で独立した値 $x[i]$ を持ち，各平面は独立に探索を行う．多重化の概念図を**図 3** に示す．図 3 のノード x_0 から x_3 はエージェントであり，エッジは制約で結ばれていることを示している．多重化したとき，エージェントは平面ごとに値 $x[i]$ を持つが，近傍のエージェントと交換するメッセージは多重化された値を 1 つのメッセージにして交換するため，メッセージ長は増加するがメッセージ数は増加しない．

```
11: Algorithm Multiplexed-S
12: while(TerminationConditionIsNotMet)do
13:    mc := 0; /* Number of Messages */
14:    while(mc < #Neighbors) do
15:       neighborsStatus[mc][ ] := Receive; mc++;
16:    endwhile
17:    forall i ∈ m do /* for each plane */
18:       x[i] := NewValue(neighborsStatus[ ][i])
19:       message[i] := x[i]
20:    end
21:    Send(message[ ])
22: endwhile
```

図 2　多重化した分散反復改善アルゴリズム $Multiplexed\text{-}\mathcal{S}$
Fig. 2　Multiplexed distributed iterative repair algorithm $Multiplexed\text{-}\mathcal{S}$.

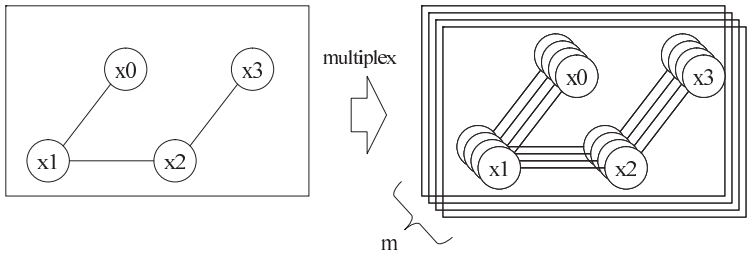

図 3　アルゴリズム多重化の概念図
Fig. 3　Concept of multiplexed algorithm.

多重化について本論文では，同時に実行する平面の数を多重度と呼び，m で表現する．

3.2　多重化により期待される効果の解析

本節では，分散近似アルゴリズムの計算成績としてアルゴリズムが停止するまでの計算時間（理想時間複雑度）が何らかの確率分布に従うものと仮定し，多重化により得られる効果について検討する．

ある分散近似アルゴリズム $\mathcal{S}$ の計算時間が分布関数 $F_s(w)$，確率密度関数 $f_s(w)$ の確率分布に従うものと仮定する．ただしアルゴリズムは有限時間内に必ず停止するものとする

($f_s(w) = 0, w \geq^{\exists} T$)．このアルゴリズム $\mathcal{S}$ を m 個の平面で多重実行して，平面の 1 つで計算が終了した時に全体も終了することにする．m 個の平面の実行は独立であると仮定する．このとき，多重度 m により計算時間 y が得られる確率は，極値分布の理論によると元の確率分布の “最小値の分布” に従い，その確率分布関数 $F_m(y)$，確率密度関数 $f_m(y)$ は次式で与えられる[18),19)]．

$$F_m(y) = 1 - (1 - F_s(y))^m \tag{1}$$

$$f_m(y) = m(1 - F_s(y))^{(m-1)} f_s(y) \tag{2}$$

この確率分布の期待値 μ_m と分散 $\sigma_m^2(m)$ は m の関数として表現可能だが，厳密解は非常に複雑になる．そこでこのような最小値の極値分布の解析は一般的に，元の確率分布には下限値が存在することを仮定した第 3 型漸近最小値分布が用いられる．元の確率分布の下限値を ϵ と仮定し，第 3 型漸近最小値分布を用いると，期待値 μ_m は次のように表せる[19)]．

$$\mu_m(m) = \epsilon + \frac{\delta}{m^{(1/k)}} \Gamma\left(1 + \frac{1}{k}\right) \tag{3}$$

同様に分散 $\sigma_m^2(m)$ は次のように表せる．

$$\sigma_m^2(m) = \left(\frac{\delta}{m^{1/k}}\right)^2 \left(\Gamma\left(1 + \frac{2}{k}\right) - \Gamma\left(1 + \frac{1}{k}\right)^2\right) \tag{4}$$

ただし $\Gamma(\ )$ はガンマ関数であり，δ，k は元の確率分布の形状に依存するパラメータである．式 (3) 導出の詳細と δ，k の説明は付録に記述する．元の確率分布の期待値を μ_s とすると $\mu_m(1) = \mu_s$ であるから，本論文では式 (3) を単純化し，$h = 1/k$ とすることで次のように扱うことにする．

$$\mu_m(m) = \epsilon + m^{-h}(\mu_s - \epsilon) \tag{5}$$

式 (5) の導出は付録に記述する．分散についても $\sigma_m^2(1) = \sigma_s^2$ を使うことで

$$\sigma_m^2(m) = m^{-2h} \sigma_s^2 \tag{6}$$

と記述できる．

h は多重化の効果の指標であり，h が大きいほど多重化の効果が大きいことに相当する．近似式 (5) より，多重化の効果は多重度 m の累乗 m^{-h} の形になり，その分散は近似式 (6) より m^{-2h} の形になる．多重化の効果の指標 h は，k の逆数であるから，元の確率分布の形状に依存することになる．多重度が $m \to \infty$ のときに期待値 μ_m は ϵ に漸近する．つまり，μ_m は ϵ より小さくなることはなく元の確率分布のレンジ（範囲）に収まる．式 (5) より，多重度 m を大きくしても多重化で得られる効果はだんだん小さくなることが分かる．

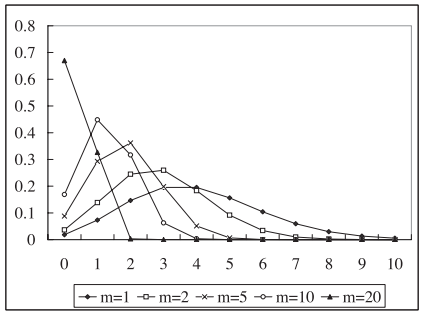

図 4　元の確率分布がポアソン分布の場合の多重化によるモードの移動
Fig. 4　The Effect of multiplexed method when F_s is poisson distribution (1).

多重化後の計算成績の確率分布が元の確率分布に対して，どのような形になるかの概念図を，**図 4** に示す．これは元の確率分布として母数 $\lambda = 4$ のポアソン分布を仮定したときに，$m = 1$，2，5，10，20（$m = 1$ は元の確率分布）で多重化した場合の確率分布を数値計算した結果である．このように m を増加させると確率分布のモード（峯）が左（小さい方向）に移動するとともに，分散が小さくなっているが，これは元の確率分布の左側を持ち上げたような形になっている．このようなモードの移動は元の確率分布がポアソン分布以外の場合であっても，あるいは計算成績として計算時間でも解の品質でも，同様に発生する．

m を増加させたとき，多重化後の確率分布の期待値 μ_m について，上記 $\lambda = 4$ のポアソン分布の場合に数値計算した結果を**図 5** に示す．図には最小二乗法で求めた近似曲線 $\mu_m = m^{-0.51}\lambda$ を重ねて示してある．このような期待値の減少についても，元の確率分布がポアソン分布の場合に限定するものではない．我々の数値計算の結果では，元の確率分布が正規分布のような山形の確率分布の場合，近似式 (5) の h はおおよそ $0 < h < 1$ の範囲に入るが，元の確率分布が幾何分布や指数分布のようにモードが最小値側に偏っていて右の裾野が長い場合，$h > 1$ もあり得る．$h > 1$ の場合，多重化の効果はスーパリニアになる．また元の確率分布の分散が大きいほど，h が大きくなる傾向がある．元の確率分布が一様分布の場合，多重化後の分布は指数分布になり，$h = 1$ になる．

3.3　多重化が効果を持つ条件

前節の議論は，計算成績のうち理想時間複雑度に注目した場合の効果だが，理想時間複雑

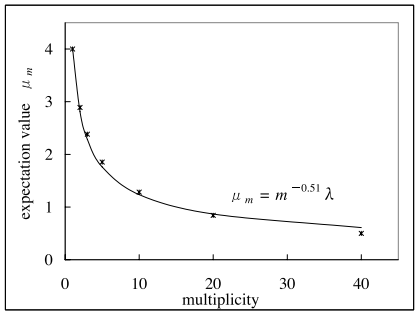

図 5　元の確率分布がポアソン分布の場合の多重化による期待値減少の効果
Fig. 5　The Effect of multiplexed method when F_s is poisson distribution (2).

度だけによる議論はあまり現実的ではない．図 2 で示したアルゴリズムの 17 行目から 20 行目ように，多重化すると複数の平面の計算が逐次的に行われるため CPU 時間は増加するし，多重度 m を非常に大きくした場合には，メッセージ長の増加による通信時間の増加も無視できない．よって多重化に効果があったとしても，多重度を上げると効果が減じることが予想される．

多重度 m の場合に期待される理想時間複雑度を $\mu_m(m)$ とした場合，多重化による増加分を考慮した計算時間 T_c は

$$T_c(m) = \mu_m(m) \times (\alpha m + \beta) \tag{7}$$

と表現できる．α，β はそれぞれ，多重化することで m に比例して増える計算時間と多重化しても増加しない計算時間の割合であり，$\alpha + \beta = 1$ とする．$\alpha : \beta$ はアルゴリズムや計算資源，通信環境によって変化する．［前提 3］は α が小さいことを仮定したものであり，これは分散環境における計算の特徴でもある．しかし現実には $\alpha > 0$ である．**図 6** に多重度 m と $\alpha : \beta$ を変化させた場合に，図 5 の効果がどのように変化するかを $m = 1$ の計算時間を 1 とした場合の比で示す．この図の条件では α が小さい場合は多重化の効果があるが，$\alpha : \beta = 0.5 : 0.5$ では効果がない．$\alpha : \beta$ がその中間の比率のとき，最適な m が存在することが予想される．

最適な m は次のように計算できる．式 (5) で $\epsilon = 0$ と仮定して，$\mu_m(m)$ が $\mu_m(m) = m^{-h}\mu_s$ と近似できる場合，多重化の効果 $e_{ff}(m)$ は $T_c(m)$ と $T_c(1)$ の比率で

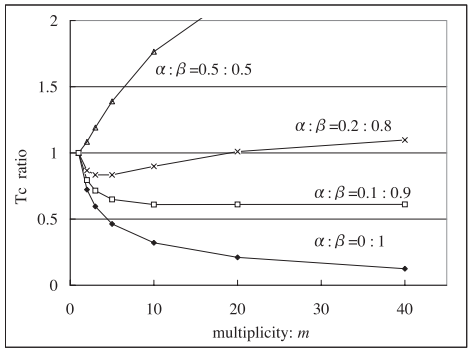

図 6　時間増加を加味した場合の多重化の効果
Fig. 6　The Effect of multiplexed method (considering a time increase).

$$\begin{aligned} e_{ff}(m) &= \frac{T_c(m)}{T_c(1)} \\ &= \frac{m^{-h}\mu_s(\alpha m + \beta)}{\mu_s} \\ &= m^{-h}(\alpha m + \beta) \end{aligned} \tag{8}$$

と表せる．これを m で微分して $= 0$ とおくと，最適な m_{opt} が計算できる．

$$\begin{aligned} &\frac{d}{dm}e_{ff}(m) = \frac{d}{dm}\left(m^{-h}(\alpha m + \beta)\right) = 0 \\ &\therefore\ m_{opt} = \frac{h\beta}{(1-h)\alpha} \end{aligned} \tag{9}$$

式 (9) を使うと，たとえば $h = 0.7$，$\alpha = 0.2$，$\beta = 0.8$ のとき，最適な多重度は $m_{opt} \approx 9$ と求めることができる．また，そのときの実計算時間は式 (8) より，元の計算時間の 0.56 倍と計算できる．

多重化が効果を持つ条件（h，α，β の関係）は式 (8) を $e_{ff}(2) < 1$ とおくと

$$\alpha < 2^h - 1 \tag{10}$$

と計算できる．定義より $0 \le \alpha \le 1$ であるから，$h > 1$ では任意の α で多重化が意味を持つことになる．前節で述べたとおり，元の確率分布が指数分布や幾何分布のように左に偏っている場合は $h > 1$ になることがある．

以上のように，多重化の効果は元のアルゴリズムの計算時間の分布，多重度，アルゴリズムや通信環境における α と β の比などにより大きく異なることになる．それでも α よりも β が大きい通信環境ほど，あるいは α よりも β が大きくなるアルゴリズムほど多重化の効果は大きいといえる．つまり，ノード間が遠く離れていて通信遅延が大きい場合や，一部のセンサネットワークのように通信遅延が大きく設定されている場合，あるいはシンプルで計算量が少ないアルゴリズムなどはその効果が現れやすいことになる．

4. 多重化効果の実験による検証

本章では分散制約最適化問題（DCOP）用近似アルゴリズムの多重化による計算時間短縮効果を実験により検証する．実験では，DCOP のための単純な分散近似アルゴリズムを用意し，これを多重化することで計算終了までの時間にどのような効果（式 (5) の h の大きさ）が出るかを調べる．実験は，文献 4)，7)，9) にならい，分散グラフ彩色問題をシミュレータを用いて解き，そのメッセージ交換ラウンド数（理想時間複雑度）と導出された解の品質で評価を行うことにする．分散グラフ彩色問題は，分散制約充足問題のうち充足解がない過制約な問題を準備し，制約の充足状態を $\{1, 0\}$ のコスト関数として表現することで分散制約最適化問題として扱う．ただし今回はシミュレータを用いるため，$\alpha : \beta$ の測定は行わない．

4.1 実験に用いたアルゴリズム DSt

多重化分散近似アルゴリズムは，2 つのフェーズから構成されることを前提として考える．第 1 フェーズでは各平面が探索を行い，平面の 1 つ以上が探索を停止した場合，ただちに第 2 フェーズに入る．第 2 フェーズでは実行が終了した平面のうち，最も良い値を得た平面の結果で全エージェントの合意を形成する．本実験では第 1 フェーズ終了までの比較をすることにした．

多重化によるアルゴリズム停止までの時間変化を測定するためには，停止機構を持つ単純なアルゴリズムが適切であると考えられる．そこで実験には，分散 Stochastic Search アルゴリズム（DSA）[7] に停止機構を取り込んだアルゴリズム DSt を新規に開発して用いた．アルゴリズム DSt の第 1 フェーズの詳細を**図 7** に示す．

DSt は複数種類ある DSA のうちの DSA-B に相当し，DSA-B で使われている 2 つの確率を p_1 と p_2 に分離したものである．アルゴリズムは周辺のエージェントと値を交換しながら（34，40 行目），反復改善型の探索を行う．もし自分が d^*_{new} へ値を変更するとコスト関数の値を改善できる場合，確率 p_1 で値を d^*_{new} へ変更する（46 行目）．そうではない場合

```
30:Algorithm DSt phase-1
  const   D          : set :          /* domain of x */
          m          : integer :      /* multiplicity */
    var   x[ ]       : x ∈ D :        /* agent's variable */
          my_tc[]    : integer :      /* termination counter */
          r          : integer :      /* execution rounds */
31:SetInitialValue;
32:Send(x, [all 0]);
33:while(∀j : my_tc[j] < termination_cond)do
34:     ReceiveNeighborsMessages;
35:     r := r + 1
36:     for all j ∈ m do /* for each plane */
37:       my_tc[j] :=Calctc(j) /* termination counter */
38:       x[j] :=NewValue(j, r);
39:     end
        /* send multiplexed message */
40:     Send(x[ ], my_tc[ ]);
41:endwhile

42:function NewValue(j, r)
43:if(Satisfied(j,r))then
44:     return x[j]; /* nothing to do */
45:else if(∃d*_new : Σ g^k(d*_new, j) ≤ Σ g^k(x, j))then
        /* I can improve conflicts */
46:     return (d*_new with p1| x[j] with (1 − p1))
47:else
        /* otherwise */
48:     return (d_new with p2| x[j] with (1 − p2))
49:endif
```

図 7　実験用アルゴリズム DSt の詳細
Fig. 7　Details of algorithm DSt for experiments.

でも，確率 p_2 で値を任意の d_{new} へ変更する（48 行目）．r は繰返し回数である．$g^k(x, j)$ はエージェントの j 平面での変数値が x だったとき，このエージェントが持つ k 番目のコスト関数の値を表している．Send は変数の値，停止のためのカウンタ（後述）を送信する手続きである．

アルゴリズム第 1 フェーズは，多重化された平面のうちの 1 つで変数 $my\_tc[j]$ が定数 $termination\_cond$ 以上になった場合に終了する（33 行目）．第 1 フェーズ終了後，ただちに第 2 フェーズに入る．$my\_tc[j]$ は第 1 フェーズ停止のためのカウンタであり，エージェントから距離 my_tc 以内のエージェントが j 平面において探索停止条件（後述の Satisfied(j, r)）に達したことを示す[20]．$my\_tc[j]$ は以下に示す関数 Calctc(j) により計算される．エージェントは近傍のエージェントの $my\_tc[j]$ を受け取り，これを集合 $\{my\_tc\_recieved[j]\}$ として保持しているものとする．Calctc(j) は探索停止条件 Satisfied(j, r) が満たされていない間は 0 を返し，そうでなければ $my\_tc[j]$ と $\{my\_tc\_recieved[j]\}$ のうちの最小値を $+1$ し

て返す．

$$\mathrm{Calctc}(j) = \begin{cases} 0, & \text{if not Satisfied}(j,r) \\ \min(my\_tc[j], \{my\_tc\_recieved[j]\}) + 1, & \text{otherwise} \end{cases} \tag{11}$$

Satisfied(j,r) はすべての制約 k について，制約が満たされるか，あるいはこれまでの実行ラウンドにおける制約違反の平均値が閾値を上回ったときに真になる関数である．

$$\mathrm{Satisfied}(j,r) = \begin{cases} true, & \text{if } \forall k : \left(g^k(x,j) = 0 \text{ or } \dfrac{\left(\sum_r g^k(x,j)\right)}{r} > Th(r)\right) \\ false, & \text{otherwise} \end{cases} \tag{12}$$

すなわち，Satisfied(j,r) により，制約違反しやすい（制約違反回数が多い）制約は，探索停止条件として制約違反を許すことにする．これは，制約違反を繰り返すということはこの制約の周辺は過制約状態であり，最適解においても制約違反が残る部分であるという前提に基づくものである．閾値 Th(r) は以下の式に示すように，初期値が th_{init} で，単調減少する関数である．t_s は探索停止条件に到達するまでの時間を調整するためのパラメータであり，t_{min} は Satisfied() の平均を計算するために最低繰返し回数として設定されるパラメータである．

$$\mathrm{Th}(r) = \begin{cases} \infty, & \text{if } r < t_{min} \\ th_{init} - r/t_s, & \text{otherwise} \end{cases} \tag{13}$$

閾値 Th(r) が緩和され，最終的には 0 に近づくため，エージェントは必ず探索停止条件に到達することになる．

DSt は多重化による計算速度向上を計測するためのアルゴリズムであるため，第 1 フェーズで平面の 1 つ以上が探索を停止した場合，ただちに第 2 フェーズに入る．第 2 フェーズでは，制約グラフの直径が既知であるという前提のもと，各エージェントのコスト関数の値を隣から隣へと交換，あるいは転送し，各エージェントが大域的な評価値を計算するものとする．そのうえで，拡張 Chang-Roberts アルゴリズム[21] によりどの平面の解を選択するかの合意を形成することにする．ただし今回の実験では，多重化の効果の計測を目的として第 1 フェーズのみの計算時間を測定することにし，1 つ以上の平面で探索条件に到達したことをシミュレータが検知した時点でアルゴリズムを停止させ，最も良い値を得た平面の解を選択することにした．これらの平面には終了の合意（$my\_tc[j] < termination\_cond$）に到達していない平面も含まれるが，今回は多重化による解の品質の改善効果を見るために，すべての平面の中から最良の結果を選択することにした．

表 1　実験用パラメータ設定
Table 1　Parameters for experiments.

settings	(A1)	(A2)	(B)
th_{init}	0.7	0.7	0.5
t_s	(variable)	500	2000
others	$p_1 = 0.5, p_2 = 0.02$		
	$t_{min} = 20$		
	$termination\_cond = 10$		
	$m = (variable)$		

なお本アルゴリズムの場合，1 つのラウンドにつきエージェントは近傍に対して 1 つずつメッセージを送信する．このためアルゴリズム停止までのラウンド数を r_{em}，アルゴリズム停止までに交換されたメッセージ総数を $msgs$ とすると，この 2 つの間には $msgs = C \times r_{em}$（C は定数）が成立する．

4.2　実 験 条 件

今回の実験で問題として用意したのは，ランダムグラフのノードをエージェント，エッジを $x_i \neq x_j$ 型の制約と見なして，制約で結ばれたエージェントを 3 つの色で塗りわける彩色問題である．コスト関数は，制約が充足された場合は 1 を，違反している場合は 0 を返す関数とし，このコスト関数の総和を最小化することを目的とする．問題としてエージェント数 100 の問題を 10 個用意した．制約数はエージェント数に対して 3 倍に設定し，完全アルゴリズムで解いて最適解のときの違反数を事前に調べておいた．分散アルゴリズムの動作は本来，分散環境で非同期に実行されるが，今回の実験では各エージェントが同期的にメッセージ交換と処理を行ったときのアルゴリズム停止までのメッセージ交換ラウンド数（理想時間複雑度）で時間評価を行う．同時に，導出した解について最適解からの距離で解の品質の評価を行う．

今回の実験に用いたアルゴリズムパラメータの設定を**表 1** に示す．アルゴリズム停止用閾値 Th(r) の設定として (A1)，(A2)，(B) の 3 つを用意し，多重度 m を変化させた．(B) は (A1)，(A2) に比べ，より停止に近い初期値（th_{init}）から始めて長い時間（t_s）をかけるもので，停止までの時間のばらつきが大きくなることを目指した設定である．各エージェントの変数の初期値はアルゴリズム開始時に平面ごとに乱数で決めた．アルゴリズムパラ

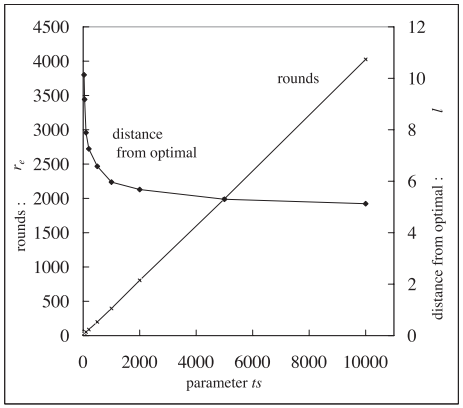

図 8　アルゴリズム DSt 単体の性質（$m = 1$ の場合）
Fig. 8　The character of algorithm DSt ($m = 1$).

メータの設定組合せ 1 つにつき，1 つの問題インスタンスについて 30 回ずつ（問題インスタンスは 10 個あるので合計 300 回）実行し，その平均を評価値として，パラメータ設定別に集計することを基本にする．

4.3　実 験 結 果

4.3.1　アルゴリズム DSt の性質の確認実験

まずアルゴリズム単体の性質を調べる．**図 8** は表 1 の条件 (A1) で停止時間をコントロールするパラメータ t_s を変化させながら実行した場合の終了までのラウンド数（理想時間複雑度）r_e と最適解からの距離 l である．図 8 からは，終了するまでのラウンド数 r_e は t_s に比例していることが分かる．t_s を増加させると（終了するまでのラウンド数が増加すると）解の品質（最適解からの距離 l）が改善されているが，この曲線はおおよそ t_s の累乗で近似できる．以下，このアルゴリズムを多重化（$m \geq 2$ に設定）した場合について実験を行う．

4.3.2　多重化の効果の測定実験

次に，表 1 の条件 (A2) で多重度 m だけを変更しながら，1 つの問題を 5,000 回ずつ解いてその終了までのラウンド数（理想時間複雑度）の頻度分布と最適解からの距離の頻度分布を計測した．結果を**図 9** と**図 10** に示す．これは 3.2 節図 4 のモードの移動を実際に観測したものである．実験に用いたアルゴリズム DSt のラウンド数と最適解からの距離はともに，多重化しない場合におおよそ山形の分布を示し，［前提 2］を満たす．そして多重化

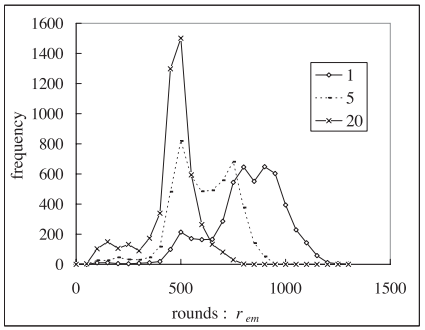

図 9　多重度 m を変化させた場合のラウンド数の頻度分布の変化
Fig. 9　The change in the frequency distribution of the number of rounds when multiplicity m is changed.

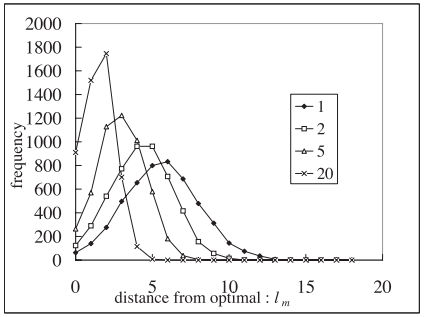

図 10　多重度 m を変化させた場合の最適解からの距離の頻度分布の変化
Fig. 10　The change in the frequency distribution of distance from optimal when multiplicity m is changed.

した場合，分布のモードが左に移動するとともに，ばらつきが小さくなっていることが分かるが，これは理論的な解析とおおよそ一致する．

同じく多重度を変更した場合に，停止するまでのラウンド数 r_{em} の平均と最適解からの距離 l_m の平均がどのように変化するかを調べた．表 1 の条件 (A2) と (B) を用いた結果の概要を**表 2** に，ラウンド数の変化を**図 11** に，最適解からの距離の変化を**図 12** に示す．いずれも 10 個の問題を 30 回ずつ解いて平均を求めている．ラウンド数の変化，最適解からの距離の変化それぞれについて多重化の効果があり，その効果を表す h は別である．表と 2

表 2　実験結果．平均値とその分散（抜粋）
Table 2　Part of experiments results when multiplicity is changed, mean value and variance.

settings			(A2)	(B)
	$m=1$	μ	805.80	810.04
		σ^2	(31216.00)	(246423.00)
round	$m=40$	μ	375.57	55.25
r_{em}		σ^2	(15479.40)	(425.64)
	h		≈ 0.21	≈ 0.78
	$m=1$	μ	5.68	7.62
distance		σ^2	(5.40)	(5.87)
from	$m=40$	μ	1.55	2.77
optimal		σ^2	(0.90)	(1.30)
l_m	h		≈ 0.35	≈ 0.27

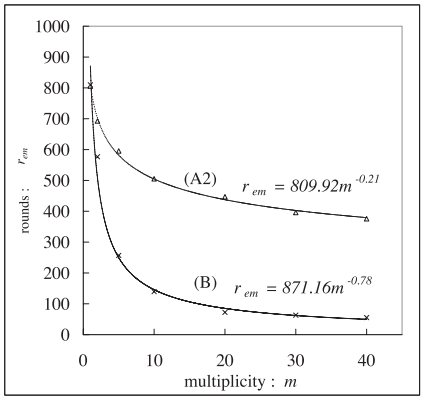

図 11　多重度を変えた場合の実行ラウンド数（理想時間複雑度）
Fig. 11　Number of execution rounds when multiplicity is changed.

つの図にはそれぞれ最小二乗法により求めた h と近似曲線も示す．これらの図は，3.2 節において期待できる効果として示した図 5 の平均値の減少を実際に観測したものであり，理論的解析で示した m の累乗の形をしている．表 2 と図 11 に示すとおり，条件 (A2) と (B) では $m=1$ のときのラウンド数がほぼ同じだが，$m=1$ のときの分散が大きい (B) の方がより大きな h になっていることが確認できる．

4.3.3　同一の品質の解を求める実験

今回の実験に用いたアルゴリズム DSt は，多重化により処理時間短縮と解の品質改善の

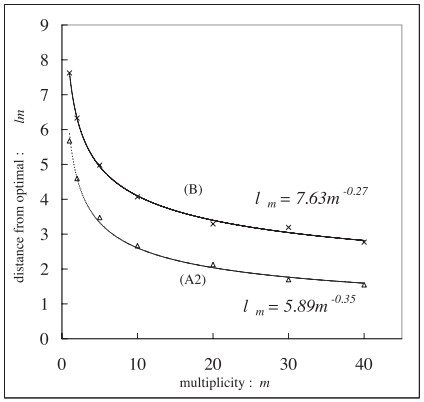

図 12　多重度を変えた場合の最適解からの距離
Fig. 12　Distance from optimal solution when multiplicity is changed.

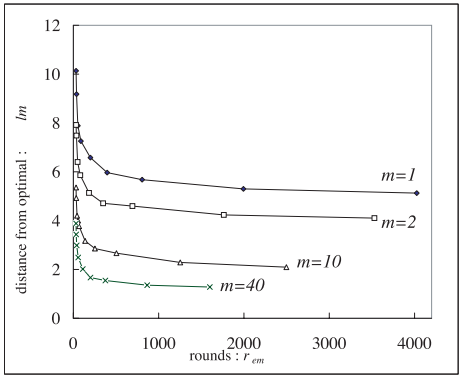

図 13　実行ラウンド数（理想時間複雑度）と解の品質の関係
Fig. 13　Relation between number of execution rounds and quality of solution.

両方を実現している．そこで実行時間と解の品質の関係が多重化によってどのように変化するかを調べた．表 1 の条件 (A1)，多重度 $m=1$，2，10，40 の場合について，t_s パラメータを 30 から 10000 の範囲で変化させながら停止するまでのラウンド数を変化させた場合の，ラウンド数と得られた解の品質の関係を**図 13** に示す．このラウンド数–品質曲線が左下にあるほど，短い時間で高い品質の解が得られる効率的なアルゴリズムということにな

表 3　同じ品質の解を得ようとした場合の多重度とラウンド数．平均値と分散
Table 3　Multiplicity and number of rounds when the same quality is obtained.

m	t_s	rounds : r_{em}		distance from optimal: l_m
1	—	—		—
2	15000	5376.65	(1830111)	3.93 (3.36)
5	400	128.22	(533.48)	4.07 (2.40)
10	150	54.97	(42.86)	4.00 (1.94)
20	50	33.22	(0.70)	4.14 (1.60)
30	30	32.00	(0.00)	3.98 (1.58)

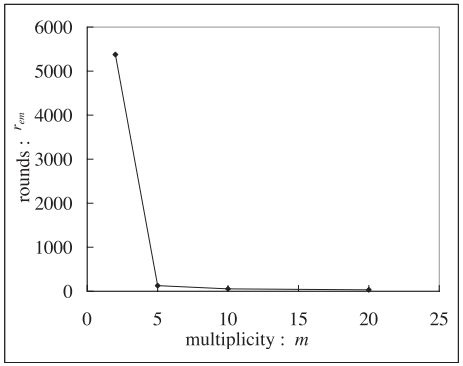

図 14　同じ品質の解を得ようとした場合の多重度とラウンド数の関係
Fig. 14　Multiplicity and number of rounds when the same quality is obtained.

る．実験に用いたアルゴリズム DSt については，多重度を変更した場合，多重度が大きいほど曲線が左下に位置していて効率が上昇していることが分かる．

今回の実験では，実行時間の短縮と解の品質改善が同時に実現されたわけだが，このような場合，アルゴリズムの効率の観点からは，アルゴリズムのパラメータ設定を変えずに多重度だけを変更したときの結果を単純に比較するべきではない．そこで表 1 の条件 (A1) を用いて，多重度を変えながら最適解からの距離 $l_m \approx 4$ が得られるようにパラメータ t_s を人手で調整したときに，停止までのラウンド数がどのように変化するかを測定した．結果を**表 3** に，ラウンド数を**図 14** に示す．$m = 1$ では現実的な時間内に $l_m = 4$ を得ることはできなかった．この結果からは，多重度を上げると解を得るためのラウンド数が劇的に減少していることが分かる．図 14 の減少率は式 (5) の $h > 2$ に相当する．このように実行時間と解の品質の改善が同時に達成される場合，多重化の効果は非常に大きいといえるだろう．

5. 考　　察

前章では，アルゴリズムの計算時間を理想時間複雑度（ラウンド数）で比較した．しかし 3.2 節で議論したとおり，現実には分散アルゴリズムの実計算時間は多重度が増すと式 (7) に従って増加するはずである．前章の理想時間複雑度は式 (7) において $\alpha = 0$，$\beta = 1$ の場合に相当した．一方，分散制約最適化問題のための計算量の指標に Non-concurrent Constraint Check（NCCC）[22] がある．NCCC は，並列に実行される分を含めて各エージェントの制約チェック回数を単純に合計するものなので，総計算量に相当する．実験に用いたアルゴリズム DSt は，停止するまでの間，各エージェントが毎回制約違反のチェックを行うため，NCCC は ($round$ 数) $\times m$ で近似できる．すなわち実験に用いたアルゴリズム DSt においては NCCC は，$\alpha = 1$，$\beta = 0$ に相当する．実際の $\alpha : \beta$ は 0 : 1（理想時間複雑度）と 1 : 0（NCCC）の間にあり，理想時間複雑度も NCCC も多重化の効果を測るための現実的想定とはいい難い．

前章の実験結果図 11，図 12，図 14，は $\alpha : \beta$ によって 3.2 節の図 6 同様に変化する．このとき，式 (10) より多重化が効果を持つ条件が計算できる．図 11 に示した条件 (A2) のラウンド数 r_{em} の実験結果では，式 (5) の h が 0.21 であるため，式 (10) より $\alpha < 0.16$ の範囲で多重化の効果が期待できる．同様に条件 (B) のラウンド数 r_{em} は $h = 0.78$ であるので，$\alpha < 0.72$ の範囲で多重化の効果が期待できる．

一方，同じ品質の解が得られるように調整した 4.3.3 項図 14 の条件では $h > 2$ であるため，任意の α で多重化が効果を持つことになる．図 14 は条件 (A1) を用いているが，多重化の効果は条件 (A2) とほぼ同じはずである．条件 (A2) のラウンド数と品質について，多重化の効果はそれぞれ表 2 に示すとおり $h = 0.21$ と $h = 0.35$ であったのに，合わせた図 14 の効果が $h > 2$ になっている．これは次のように説明できる．DSt は 4.3.1 項図 8 に見たとおり，$m = 1$ のときに最適解からの距離と終了までのラウンド数は累乗の関係にあった．l を最適解からの距離，r_e を終了までのラウンド数，a_1 と b を定数とすると，これらの関係は次式のように書ける．

$$l = a_1 r_e^{-b} \tag{14}$$

ここで多重度 m で多重化すると，終了までのラウンド数と最適解からの距離はそれぞれ独立に減少する．このそれぞれについて多重化の効果を h_1 と h_2 で表すと，式 (5) より終

了までのラウンド数は

$$r_{em} = m^{-h_1} r_e \tag{15}$$

最適解からの距離 l は

$$l_m = m^{-h_2} l \tag{16}$$

に変化する．これらを式 (14) に代入すると次式を得る．

$$m^{h_2} l_m = a_1 (m^{h_1} r_{em})^{-b} \tag{17}$$

この m をパラメータとした l_m と r_{em} の関係式は図 13 の各曲線に相当する．得られる解の品質 $l_m = 4$ を一定にするように調整するということは，図 13 に最適解からの距離=4 の直線を引き，この直線と各曲線との交点を測定したことに相当する．式 (17) に l_m を定数にするという条件を加えて r_{em} について解くと，m と r_{em} の関係として次式が導かれる．

$$r_{em} = a_2 m^{-(h_1 + h_2/b)} \tag{18}$$

a_2 は定数である．これが図 14 に相当する．すなわち，時間短縮の効果は h_1 と h_2 だけではなく，多重化していない場合の時間と品質の関係曲線の指数 b も関係してくる．b が小さくて h_2 が大きいほどこの効果は大きい．つまり，図 13 の曲線がより水平に近い形で，多重化により曲線が下方向に大きく動くとき，曲線と $l_m =$ 定数との交点が大きく動くことになる．実際今回の実験でも，$m = 1$ の曲線と $l_m = 4$ の直線との交点は無限遠だったほどである．このため，b と h_2 の比率によっては，$h_1 < 1$ かつ $h_2 < 1$ であったとしても，見た目の多重化の効果がスーパリニアになる可能性がある．

ところで，表 2 に示したように，多重度を大きくして計算した方が小さな多重度のときよりもラウンド数や最適解からの距離の分散が小さくなっていることが観測できた．分散が小さくなることは式 (4) でも理論的に示したが，アルゴリズムとしては期待される計算時間や解の品質の最悪値を想定しなければならない場合が多く，分散が小さいことはアルゴリズムとして扱いやすい（各種の見積りがしやすい）ことを意味する．これは多重化の重要な効果の 1 つといえるだろう．

本論文で提案した多重化方式は，複数の候補から最も良いものを選択するという意味では遺伝的アルゴリズムの概念に通じるものがあるといえる．しかし今回の実験に用いた DSt では，多重化された平面間で情報の共有（たとえば nogood 情報の共有など）や戦略の交換などをいっさい行っていないし，今回の実験では変数の初期値も平面ごとに独立に決めている．情報を共有することは，各平面での探索空間の範囲について相関性を増加させ，各平面の探索が独立試行ではなくなることを意味する．本実験で用いたアルゴリズムのように単純な確率的探索の場合，情報の共有は探索空間を間違った方向に狭めてしまい，多重化は計算時間を増加させるだけになってしまう危険性がある．ただし，たとえば ADOPT のように DSt よりも “賢い” アルゴリズムでは，平面間の情報共有は探索空間を効率的に絞ることに貢献して，多重化の効果のさらなる増進が期待できる．

この多重化方式は，多重化したときに各エージェントの計算時間やメッセージ数，メッセージ長に極端な増加がない場合に限り，分散制約最適化問題（DCOP）ばかりではなく分散環境における探索アルゴリズムに広く適用可能と考えられる．たとえば DBA [2] は分散制約充足問題（DisCSP）のための近似アルゴリズムであるが，多重化により停止までの時間の短縮が期待できる．

6. おわりに

本論文では分散制約最適化問題のための分散近似アルゴリズムの高速化手法として，多重化手法を提案した．そしてアルゴリズム多重化の効果について理論的に解析するとともに，実験により，計算時間短縮と解の品質の改善に大きな効果が期待できることを確認した．

関連する研究としてアルゴリズムポートフォリオの研究[11] では，1 種類以上の探索アルゴリズムを並列実行することで，探索効率が大きく改善できることを示している．アルゴリズムポートフォリオでは，複数の CPU を使う場合ばかりではなく，1 つの CPU で擬似並列実行する場合も効果があると述べられている．また，文献 23) では，計算時間の確率分布の裾野が非常に厚い場合（期待値が計算できないような場合）について，同じアルゴリズムを繰り返し適用した場合の解析が行われている．このアルゴリズムポートフォリオの研究以前にも，探索アルゴリズムは計算時間が膨大になることから，並列実行による高速化が数多く試みられてきた歴史がある．近似アルゴリズムを並列実行することにより，計算時間の短縮だけでなく解の品質も向上することについては，すでに文献 24) において述べられている．

これらの研究は基本的に非分散問題を扱った並列実行，あるいは擬似並列実行についての議論であった．一方，文献 15) では分散制約充足問題（DisCSP）の tree-based アルゴリズムを多重化する方式が述べられている．これはヒューリスティックス選択などのリスクを分散させるために多重化を導入し，実験によって評価している．

以上の既存研究は，理論的考察が行われてはいるものの，その効果については実験により

評価をしている．これに対し本論文では，分散制約最適化問題（DCOP）の非決定性近似アルゴリズムを多重化することの効果を極値分布理論を用いて理論的に解析した．その結果，通信時間に対してエージェントの計算時間が十分小さいという理想的な条件下では，計算時間や解の品質について，多重化の効果は多重度 m の累乗の形 m^{-h} で現れること，その分散は m^{-2h} の形になること，多重化の効果の指標となる指数 h は元の確率分布の形状に依存していることを示した．さらに分散環境特有の条件下では，多重化により計算資源を増やすことなく高速化が可能であり，多重度には最適値 m_{opt} が存在すること，および多重化が効果を持つ条件を示した．そして，高速化と解の品質の向上を同時に実現した場合，見かけ上，スーパリニアに高速化される例を実験により示すとともに，そのメカニズムを理論的に解析した．今回の実験では多重化による高速化と解の品質の向上を目的としたアルゴリズムを用いたが，これらの効果と同時に，計算結果の分散が小さくなることも確認できた．

今回の実験には単純なアルゴリズムを用いた．今後はアルゴリズム実行中に得られた情報の平面間での共有の方法や多重度の自律的調整法などについて検討を進めるとともに，多目的最適化など，最適解の位置付けについても研究を進める必要がある．

参 考 文 献

1) 飯塚泰樹，鈴木浩之，竹内郁雄：分散制約充足問題のための Multi-agent Tabu Search 手法の効果，信学論（D），Vol.J90-D, No.9, pp.2302–2313 (2007).
2) Yokoo, M. and Hirayama, K.: Algorithms for Distributed Constraint Satisfaction: A Review, *Autonomous Agents and Multi-Agent Systems*, Vol.3, No.2, pp.198–212 (2000).
3) Calisti, M. and Neagu, N.: Constraint satisfaction techniques and software agents, *Agents and Constraints Workshop at AIIA'04* (2004).
4) Modi, P.J., Shen, W.-M., Tambe, M. and Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees, *Artif. Intell.*, No.161, pp.149–180 (2005).
5) Petcu, A. and Faltings, B.: A Scalable Method for Multiagent Constraint Optimization, *Proc. International Joint Conference on Artificial Intelligence*, pp.266–271 (2005).
6) Mailler, R. and Lesser, V.: Solving Distributed Constraint Optimization Problems Using Cooperative Mediation, *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, IEEE Computer Society, pp.438–445 (2004).
7) Zhang, W., Wang, G., Xing, Z. and Wittenburg, L.: Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks, *Artif. Intell.*, No.161, pp.55–87 (2005).
8) Silaghi, M.C. and Yokoo, M.: Discussion on the Three Backjumping Schemes Existing in ADOPT-ng, *8th International Workshop on Distributed Constraint Reasoning, in 20th International Joint Conference on Artificial Intelligence*, pp.83–97 (2007).
9) Yeoh, W., Koenig, S. and Felner, A.: IDB-ADOPT: A Depth-First Search DCOP Algorithm, *8th International Workshop on Distributed Constraint Reasoning, in Twentieth International Joint Conference on Artificial Intelligence*, pp.56–70 (2007).
10) Junges, R. and Bazzan, A.L.C.: Evaluating the performance of DCOP algorithms in a real world, dynamic problem, *AAMAS '08: Proc. 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp.599–606 (2008).
11) Gomes, C.P. and Selman, B.: Algorithm portfolios, *Artif. Intell.*, Vol.126, pp.43–62 (2001).
12) 飯塚泰樹，竹内郁雄：分散制約最適化問題のための高速近似解法の提案と評価，コンピュータソフトウェア（ソフトウェア科学会論文誌）(2009).
13) Fitzpatrick, S. and Meertens, L.: An Experimental Assessment of a Stochastic, Anytime, Decentralized, Soft Colourer for Sparse Graphs, *1st Symposium on Stochastic Algorithms: Foundations and Applications*, pp.49–64 (2001).
14) 人工知能学会（編）：人工知能学事典，共立出版 (2005).
15) Ringwelski, G. and Hamadi, Y.: Boosting Distributed Constraint Satisfaction, *CP-2005*, pp.549–562 (2005).
16) Gray, J.: The cost of messages, *PODC '88: Proc. 7th Annual ACM Symposium on Principles of Distributed Computing*, ACM, pp.1–7 (1988).
17) 萩原兼一，増澤利光：分散アルゴリズム，情報処理，Vol.31, No.9, pp.1245–1256 (1990).
18) 蓑谷千凰彦：統計分布ハンドブック，朝倉書店 (2003).
19) 柴田明徳：確率的手法による構造安全性の解析，森北出版 (2005).
20) 横尾　真，平山勝敏：分散 breakout：反復改善型分散制約充足アルゴリズム，情報処理学会論文誌，Vol.39, No.6, pp.1889–1897 (1998).
21) Tel, G.: *Introduction to Distributed Algorithms Second Edition*, chapter 7, p.247, CAMBRIDGE University Press (2000).
22) Meisels, A., Kaplansky, E., Razgon, I. and Zivan, R.: Comparing Performance of Distributed Constraints Processing Algorithms, *Workshop on Distributed Constraint Reasoning, in 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp.86–93 (2002).
23) Gomes, C.P., Selman, B., Crato, N. and Kautz, H.A.: Heavy-Tailed Phenomena

in Satisfiability and Constraint Satisfaction Problems, *Journal of Automated Reasoning*, Vol.24, No.1/2, pp.67–100 (2000).
24) Falco, I.D., Balio, R.D., Tarantino, E. and Vaccaro, R.: Improving Search by Incorporating Evolution Principles in Parallel Tabu Search, *1994 IEEE Conference on Evolutionary Computation*, pp.823–828 (1994).
25) 杉山高一，杉浦成昭，国友直人，藤越康祝（編）：統計データ科学事典，朝倉書店 (2007).
26) 東京大学教養学部統計学教室（編）：統計学入門，東京大学出版会 (1991).
27) 腐食防食協会（編）：装置材料の寿命予測入門—極値統計の腐食への適用，丸善 (1984).

付　　録

A.1　3.2 節式 (3)，(5) の導出の補足説明

ここでは 3 章で示した式 (1)，(2) から式 (3)，(5) を導出する過程について説明する．以下の導出は文献 18)，19)，25)–27) による．

5 章で示した式 (1)，(2) は以下のものだった．

$$F_m(y) = 1 - (1 - F_s(y))^m \tag{19}$$

$$f_m(y) = m(1 - F_s(y))^{(m-1)} f_s(y) \tag{20}$$

これは最小値の分布の厳密解である．元の確率分布の最小値付近では $F_s(y)$ が十分小さい．このため $ln(1 - F_s(y)) \approx -F_s(y)$ と近似できる．これを利用すると，最小値の分布の近似的な漸近分布（asymptotic distribution）として以下を導くことができる．

$$\begin{aligned} F_m(y) &= 1 - (1 - F_s(y))^m \\ &= 1 - e^{ln(1-F_s(y))^m} \\ &= 1 - e^{m(ln(1-F_s(y)))} \\ &\approx 1 - e^{-mF_s(y)} \end{aligned} \tag{21}$$

最小値の分布（式 (21)）を特徴付けるのは，元の分布の最小値付近の裾野部の形である．

最大値の分布や最小値の分布といった極値分布は，Fisher-Tippett の定理により，3 つのタイプに限られる[25]．最大値の分布の場合では，裾野の形状は，上限値がなく，指数オーダの長い裾野を持つ第 1 型，同じく上限値がなく，累乗のオーダの厚い裾野を持つ第 2 型，上限値を持つ第 3 型である．最小値の分布の場合，解析対象の事象が下限値を持つことが多いため，第 3 型が解析に用いられることが多い．本論文でも，計算時間，解の品質（最適解からの距離）はともに下限値を持つため，以下では第 3 型漸近最小値分布を扱う．

第 3 型漸近最小値分布では元の分布が下限値 ϵ，上限値 τ を持ち，最小値付近の裾野の立ち上がりが累乗の形になる次のような分布を考える．

$$\begin{aligned} F_s(x) &= \left(\frac{x-\epsilon}{\tau-\epsilon}\right)^k \\ &= \left(\frac{x-\tau}{\delta}\right)^k, \quad (\epsilon \le x \le \tau) \end{aligned} \tag{22}$$

$$\begin{aligned} f_s(x) &= \frac{k}{\tau-\epsilon}\left(\frac{x-\epsilon}{\tau-\epsilon}\right)^{k-1} \\ &= \frac{k}{\delta}\left(\frac{x-\epsilon}{\delta}\right)^{k-1}, \quad (\epsilon \le x \le \tau) \end{aligned} \tag{23}$$

ただし $\delta = \tau - \epsilon$ である．これを式 (21) に代入することで，

$$\begin{aligned} F_m(y) &= 1 - e^{-mF_s(y)} \\ &= 1 - e^{-m((y-\epsilon)/\delta)^k} \end{aligned} \tag{24}$$

を得る．ここで，$\theta = \delta/m^{1/k}$ とおくと，確率分布関数，確率密度関数はそれぞれ以下のようになる．

$$F_m(y) = 1 - e^{-((y-\epsilon)/\theta)^k} \tag{25}$$

$$f_m(y) = \frac{k}{\theta}\left(\frac{y-\epsilon}{\theta}\right)^{k-1} e^{-((y-\epsilon)/\theta)^k} \tag{26}$$

これはワイブル分布[25),26)] の形である．この確率分布の期待値は次のように求めることができる．

$$\begin{aligned} E(Y_m) &= \int_\epsilon^\infty y f_m(y) dy \\ &= \int_\epsilon^\infty y \, \frac{k}{\theta}\left(\frac{y-\epsilon}{\theta}\right)^{k-1} e^{-((y-\epsilon)/\theta)^k} dy \end{aligned} \tag{27}$$

積分範囲は本来 ϵ〜τ だが，計算上 ϵ〜∞ とする．ここで以下の変数変換を考える．

$$z = \left(\frac{y-\epsilon}{\theta}\right)^k \tag{28}$$

$$y = \epsilon + \theta z^{1/k} \tag{29}$$

$$dz = \left(\frac{k}{\theta}\right)\left(\frac{y-\epsilon}{\theta}\right)^{k-1} dy \tag{30}$$

すると積分範囲は以下のようになる．

$$y = \infty \longrightarrow \quad z = \infty \tag{31}$$

$$y = \ \epsilon \ \longrightarrow \quad z = 0 \tag{32}$$

よって平均は以下のようになる．

$$\begin{aligned} E(Y_m) &= \int_0^\infty \left(\epsilon + \theta z^{1-1/k}\right) \frac{k}{\theta} e^{-z} \left(\frac{\theta}{k}\right) z^{-1+1/k} dz \\ &= \int_0^\infty \left(\epsilon + \theta z^{1/k}\right) e^{-z} dz \\ &= \epsilon + \theta \int_0^\infty z^{1/k} e^{-z} dz \\ &= \epsilon + \theta \int_0^\infty z^{(1+1/k)-1} e^{-z} dz \\ &= \epsilon + \theta\, \Gamma\left(1 + \frac{1}{k}\right) \end{aligned} \tag{33}$$

本論文では，期待値を多重度 m の関数 $\mu_m(m)$ として表現したい．そのため θ を元に戻すと，

$$E(Y_m) = \epsilon + \frac{\delta}{m^{1/k}} \Gamma\left(1 + \frac{1}{k}\right) \tag{34}$$

以上から，k，δ，ϵ が固定の状況では，期待値は m の関数として

$$\mu_m(m) = \epsilon + \frac{\delta}{m^{1/k}} \Gamma\left(1 + \frac{1}{k}\right) \tag{35}$$

が得られる．これが式 (3) に相当する．同様な計算をすることで，式 (4) の分散が求められる．

多重化前の元の確率分布の期待値を μ_s とすると

$$\mu_s = \mu_m(1) = \epsilon + \frac{\delta}{1^{(1/k)}} \Gamma\left(1 + \frac{1}{k}\right) \tag{36}$$

$$\therefore \ \delta\, \Gamma\left(1 + \frac{1}{k}\right) = \mu_s - \epsilon \tag{37}$$

であるから，これを式 (35) に代入して，$h = 1/k$ とすることで，

$$\mu_m(m) = \epsilon + m^{-h}(\mu_s - \epsilon) \tag{38}$$

が得られる．これが式 (5) である．

(平成 21 年 4 月 6 日受付)
(平成 21 年 9 月 11 日採録)

飯塚　泰樹（学生会員）

平成元年東北大学工学部情報工学科卒業．平成 3 年同大学大学院博士前期課程修了．同年松下電器産業（株）入社．平成 18 年東京大学大学院博士課程入学．

竹内　郁雄（正会員）

昭和 44 年東京大学理学部数学科卒業．昭和 46 年同大学大学院修士課程修了．平成 8 年博士（工学）東京大学．電電公社電気通信研究所，NTT 基礎研究所，NTT ソフトウェア研究所，電気通信大学教授を経て，現在，東京大学大学院情報理工学系研究科教授．ACM，日本ソフトウェア科学会各会員．