計算数理工学論文集 Vol. 7, No.1 (2007年6月), 論文 No. 04-070629 JASCOME

SOM-GAの提案と検討について

Investigation of SOM-GA

沈侃[1)], Zhai 菲[1)], 北 栄輔[1)]

Shen Kan, Zhai Fei, and Eisuke Kita

1) 名古屋大学情報科学研究科 (〒464-8601 名古屋市千種区不老町1, E-mail: kita@is.nagoya-u.ac.jp)

Many researchers have been studying the real-coded genetic algorithm (RCGA), one of the evolutionary algorithms which is designed for finding an optimum solution of continous function. In this study, the authors describe Self-organizing maps for genetic algorithms (SOM-GA). In the present algorithm, individuals in a population are classified into sub-populations by using SOM. After performing RCGA in each sub-population, each best individual is gathered to new population. The process is repeated until obtainig an optimal solution. The present algorithm is applied to three test functions to evaluate its effectiveness.

Key Words: Genetic algorithm, Real-coded GA, Self-organizing maps.

1. はじめに

遺伝的アルゴリズムは，最初設計変数が2進数で定義できる最適化問題に対して適用されていたが，近年では設計変数が連続値として定義された問題に対する適用が熱心に研究されている．従来のビットストリングを用いる方法と異なり，設計変数値を実数値ベクトルで表現する遺伝的アルゴリズムは，実数値遺伝アルゴリズム (Real-coded Genetic Algorithms: RCGA)[1] と呼ばれている．設計変数が実数値として定義された問題では，設計変数を実数値のまま扱うほうが親個体の性質を効率よく引き継ぐことができるために，ビットストリングを用いるよりも効率的に解を探索できるとされている．そこで，本研究ではRCGAの探索性能を向上するために自己組織化マップを用いたSelf-Organizing Maps for Genetic Algorithm (SOM-GA) について述べる．

SOM-GAでは，集団を構成する個体を自己組織化マップ (Self-Organizing Maps)[2, 3, 4] によって分類し，類似の個体群からなる複数の部分集団を構成する．部分集団に対して通常のRCGAを一定回数実行し，その集団における優良個体を求める．各部分集団で求められた優良個体を集めて次世代の集団を構成する．この処理を繰り返すことで，最適解を探索していく．SOMを学習するために有る程度の回数の繰り返し処理を必要なので，このような処理を用いれば計算コストがかかることが想像される．実際，探索の初期においては提案手法の計算コストは大きく，通常のRCGAのほうが収束は早い．しかし，探索が進むにつれて，提案手法は通常のRCGAよりも加速していき，最終的にはより精度の良い解を短い時間で探索できる．

ところで，自己組織化マップ (SOM) を用いた進化的計算法としては，Bucheら[5] が提案したSelf-Organizing Maps for Multi-Objective Evolutionary Algorithms (SOM-MOEA) がある．この方法は，その名称にある "Multi-Objective" から分かるように多目的最適化問題に適用するために設計された手法である．多目的最適化問題ではパレート解を探索することが重要である．SOM-MOEAではSOMを用いてパレート解を格納することと，SOMの特長を生かして，探索パスを学習することで探索効率を改善することなどを目的としている．SOM-MOEAでは，最初に集団を構成する個体で自己組織化マップを学習する．そして，ある個体の最整合ノードをSOMマップ上で探し，その近傍ノードとの間で交叉を行って新しい個体を生成する．この方法では，交叉によって解の探索性能が向上する可能性は必ずしも高くなく，かえって性能を下げる可能性がある．著者らが，以前の研究[6] において複数のテスト問題でSOM-MOEAを代表的な多目的進化的計算法であるNSGA-II[7] やSPEA2[8] と比較したところ，従来の方法に比べて特別に優秀な探索性能を示すことはなかった．そこで，この問題を解決するために，提案手法では類似個体群からなる部分集団内で通常のRCGAを一定回数実施することとした．このような処理を用いれば，部分集団内での探索をある程度収束させることができるので，新たに生成された個体が改悪に進むことは起こらず，探索を十分すすめることができる．なお，本論文では単一目的の最適化問題を解析対象としているので，多目的最適化問題のために開発されたSOM-MOEAとの比較は行っていない．

本論文の構成は以下のようになっている．第2節では，研

究の背景について述べる．第3節では，自己組織化マップと本研究で提案するSOM-GAのアルゴリズムについて述べる．第4節では，本研究が提案したSOM-GAと実数値SGAが三つの評価関数対する計算結果について比較して，評価する．第5節はまとめである．

2. 研究の背景

2.1. SOM-MOEAの概要[5]

SOM-MOEAのおおよそのアルゴリズムは以下のようになる．

1. 集団の個体について適応度を評価し，SOMマップを学習する．
2. ランダムに個体を選択する．
3. 選択された個体に対してSOMマップ上での最整合ノードを選択する．
4. 最整合ノードの近接ノードを選択する．
5. 近接ノードに最整合する個体と元の個体の内挿点として新たな個体を生成する．
6. 正規乱数を加えることで，個体に突然変異を与える．

この方法では，選択された個体の最整合ノードをSOMマップ上で選択し，それに近接するノードに属する個体との間で1回だけ交叉を行う．SOMマップで近接するノード同士は一般に類似していると想像されるので，この交叉処理は各個体の近傍を探索することになる．そこで，近傍以外のところの探索を行うために突然変異が用いられる．突然変異では，個体に乱数を加えることで交叉では得られない個体を生成することができ，解の多様性を改善して，近傍以外のところでの探索をすすめることになる．

SOMを用いれば多数の個体を類似の個体ごとに分類することができるので，適切に利用すれば類似個体間での探索から局所探索を早め，最適解への収束を早める可能性がある．しかし，Bucheらの方法では，選択された個体の近傍から選択される個体が解の性能を改善するように適切に選択される保証がない．その結果，従来の方法に比べて有効な探索性能を示さなかったと考えられる．

2.2. SOM-GAの概要

Self-Organizing Maps for Genetic Algorithms (SOM-GA)の処理を箇条書きにすると以下のようになる．

1. 元の集団にRCGAを1回だけ適用して別の集団を作る．
2. 個体の設計変数値と目的関数値を入力ベクトルとして，SOMマップを学習させる．これにより，類似の個体はSOMマップ上で近くに写像されることになる．
3. 各個体のSOMマップ上での最整合ノードを探索する．
4. 最整合ノードの近傍にマップされている複数の個体で部分集団を構成する．
5. 各部分集団にRCGAを適用する．
6. 各部分集団で得られた最良個体を次世代の個体とする．
7. 新たに構成された集団を用いて同様の操作を繰り返す．
8. 最終的に得られた最良個体を解とする．

SOM-MOEAでは選択された個体間で交叉は1回しか行われない．交叉される個体はランダムに選択されるので適切に選択されないと，新たな個体は親個体に比べて改悪される可能性があり，近傍探索が進まないことになる．これに対して，提案手法では，自己組織化マップを用いて集団全体から複数の部分集団を作成し，部分集団ごとにRCGAを行う．複数の個体で部分集団を構成し，そこで一定回数の探索を行うことで，近傍探索をある程度収束させることができるので，新たに生成された個体が改悪に進むことは起こらず，近傍探索を十分すすめることができる．

3. SOM-GAアルゴリズム

3.1. 自己組織化マップ(SOM)[2, 3, 4]

自己組織化マップは動物や人間の視覚に関係する細胞で行われる自己組織化過程の研究に基づいており，教師なし学習を経てパターン間の関係を表出させる2層ネットワークである．第1層は入力層であり，第2層は競合層(マップ層)と呼ばれる．これを用いることで，多次元データを2次元のマップに写像し，拡散した情報に含まれる潜在的な知識が獲得できる．本研究では，ノードを六角形に配置した2次元のSOMマップを用いている．

SOMマップのi番目のノードにおかれた参照ベクトルを

$$\boldsymbol{w}^i = \left\{w_1^i, w_2^i, \cdots, w_n^i\right\}^T, \tag{1}$$

j番目の入力ベクトルを

$$\boldsymbol{v}^j = \left\{v_1^j, v_2^j, \cdots, v_n^j\right\}^T \in \Re^n \tag{2}$$

と定義する．

ノルムとしてユークリッド距離$\left\|\boldsymbol{v}^j - \boldsymbol{w}^i\right\|$をとり，入力$\boldsymbol{v}^j$に対する最整合ノードをノルムを最小にするノードとして選択する．これを$c\left(\boldsymbol{v}^j\right)$とすると，次式で定義できる．

$$c\left(\boldsymbol{v}^j\right) = \arg\min\{\left\|\boldsymbol{v}^j - \boldsymbol{w}^i\right\|\} \tag{3}$$

ひとたび$c(\boldsymbol{v}^j)$が決定されると，入力ベクトルに近づくように参照ベクトルは次式で更新される．

$$\boldsymbol{w}^i(t+1) = \boldsymbol{w}^i(t) + h_{ci}(t)\left\{\boldsymbol{v}^j(t) - \boldsymbol{w}^i(t)\right\}$$

ここで，$t = 0, 1, 2, \cdots$は離散時間である．また，$h_{ci}(t)$は近傍関数と呼ばれ，格子点上に定義される平滑効果カーネルとして作用する．収束するためには，$t \to \infty$において$h_{ci}(t) \to 0$であることが必要であるので，次式のような関数として定義される．

$$h_{ci}(t) = \alpha_s \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) \tag{4}$$

ここで，r_iはノードiの位置ベクトルを，r_cは最整合ノードの位置ベクトルを示す．α_sは学習率と呼ばれ，$0 < \alpha_s < 1$にとられる．α_sを時間の単調減少関数とする場合もあるが，

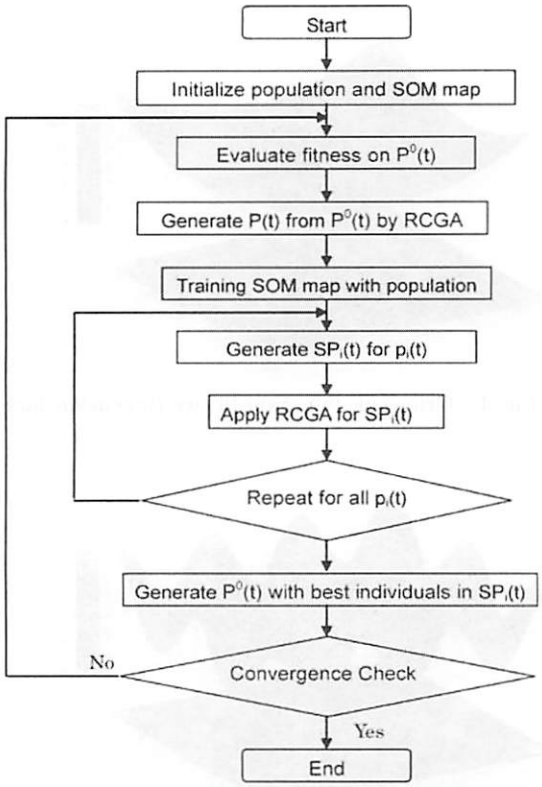

Fig. 1　SOM-GA algorithm

本研究では定数としている．また，$\sigma(t)$ は時間の単調減少関数として定義され，次式のように与える．

$$\sigma(t) = \sigma(t-1) - \frac{R}{TS} \tag{5}$$

ここで，TS は学習ステップ回数，R は初期半径である．また，$\sigma(0) = R, \sigma(TS) = 0$ である．

SOM アルゴリズムの流れは以下のようになる．

1. 入力層とマップ層間の初期重みをランダムに設定する．
2. 入力サンプルからランダムに選択して．マップ層で入力ベクトルとの距離を計算し，式 (3) を満たすノード $c(v^j)$ を選択する．
3. 式 (4) により参照ベクトルを更新する．
4. 式 (4) により近傍関数を更新する
5. ステップ (2) へ戻る．

3.2. 提案手法のアルゴリズム

提案手法のアルゴリズムは以下のようになる．また，フローチャートを 1 に示す．以下において，世代 t における集団を $\boldsymbol{P}(t)$，そこに属する個体を $p_i(t)(i = 1, 2, \cdots, M)$，探索の過程で作成する部分集団を $\boldsymbol{SP}(t)$ とする．また，自己組織化マップを Map とする．

1. $t = 0$ とする．
2. 自己組織化マップ Map を初期化する．
3. ランダムに生成した M 個の個体から初期集団 $\boldsymbol{P}^0(t)$ を構成する．

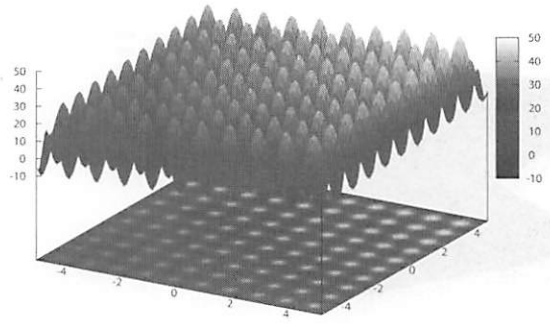

Fig. 2　Rastrigin Function in two-dimension

4. 個体の適応度を評価する．
5. 集団 $\boldsymbol{P}(t)$ を生成する．
   (a) 集団 $\boldsymbol{P}^0(t)$ からルーレット選択によって 2 個体を選択し，BLX-α 交叉により新たな個体を生成する．
   (b) (5a) の操作を繰り返して，M 個の個体からなる集団 $\boldsymbol{P}(t)$ を生成する．
6. 集団 $\boldsymbol{P}(t)$ に属する全個体によって Map を学習する．このとき，入力ベクトルとして各個体の設計変数値と目的関数値をとる．
7. 以下の操作により新たな個体を生成する．
   (a) Map 上において，個体 $p_i(t)$ を中心としてユークリッド距離が Rn 以内に含まれる全ての個体で部分集団 $\boldsymbol{SP}_i(t)$ を構成する．
   (b) $\boldsymbol{SP}_i(t)$ において RCGA を N_g 回行い，得られた最良個体を $p_i(t+1)$ とする．
   (c) $p_i(t)(i = 1, 2, \cdots, M)$ について上記の操作を行い，$p_i(t+1)(i = 1, 2, \cdots, M)$ を得る．
8. 最良個体が必要な精度を有していれば出力して，処理を終える．そうでなければ，以下の処理を行う．
9. 個体 $p_i(t+1)(i = 1, 2, \cdots, M)$ により次世代の集団 $\boldsymbol{P}^0(t+1)$ を構成する．
10. $t \leftarrow t+1$ として，ステップ (4) へ戻る．

4. 数値実験

ここでは，Rastrigin 関数，Rosenbrock 関数，Griewank 関数の三つの評価関数を用いて，本研究で提案した SOM-GA の探索性能を RCGA と比較する．提案手法である SOM-GA の中で用いる RCGA，さらに，比較に用いる RCGA においても，そのアルゴリズムの中で交叉法に BLX-α を，突然変異に実数値突然変異を用いる．

4.1. 評価関数

Rastrigin 関数

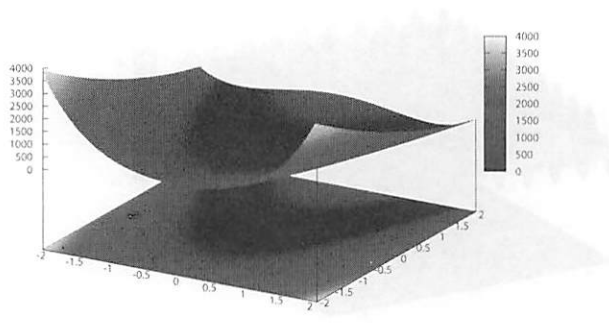

Fig. 3　Rosenbrock Function in two-dimension

Rastrigin 関数を式 (6) で定義する．設計変数が 2 個の場合のグラフを Fig.2 に示す．

$$\begin{aligned} F(x) &= 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right) \\ &(-5.12 < x_i < 5.12) \\ &\min(F(x)) = F(0,0,\ldots,0) = 0 \end{aligned} \tag{6}$$

Rastrigin 関数は，最適解の周辺に格子状に準最適解を持つ多峰性関数である．また設計変数間に依存関係がない問題である．

Rosenbrock 関数

Rosenbrock 関数を式 (7) で定義する．設計変数が 2 個の場合のグラフを Fig.3 に示す．

$$\begin{aligned} F(x) &= \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right) \\ &(-2.048 < x_i < 2.048) \\ &\min(F(x)) = F(1,1,\ldots,1) = 0 \end{aligned} \tag{7}$$

Rosenbrock 関数は設計変数間に依存関係を持つ単峰性関数である．また，この関数は一般的に GA にとって解き難い問題と言われる．

Griewank 関数

Griewank 関数を式 (8) で定義する．設計変数が 2 個の場合のグラフを Fig.4 に，最適解近傍の様子を Fig.5 に示す．

$$\begin{aligned} F(x) &= 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\left(\cos(\frac{x_i}{\sqrt{i}})\right) \\ &(-512 < x_i < 512) \\ &\min(F(x)) = F(0,0,\ldots,0) = 0 \end{aligned} \tag{8}$$

Griewank 関数は設計変数間に依存関係を有する多峰性関数である．大域的には単峰性関数のような性質を持つため，準最適解は比較的容易に求まる．しかし，局所的には多数の局所的最適解が存在するため，最適解を探索するのは困難である．

4.2. パラメータ設定

SOM-GA と RCGA に用いる共通パラメータの設定値を Table 1 に示す．各最適化問題において設計変数は 20 個である．個体集団は 50 個体からなり，交叉率 1.0，突然変異率 0.002 としている．選択法としてルーレット選択，交叉法と

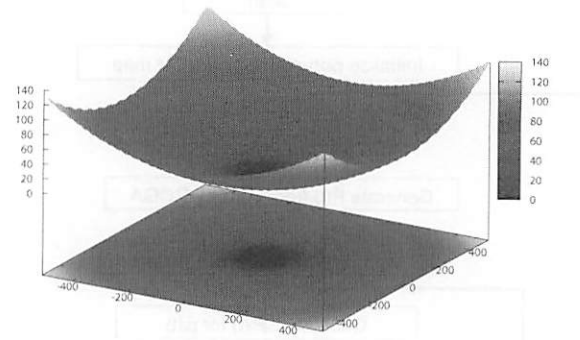

Fig. 4　Griewank Function in two-dimension(large)

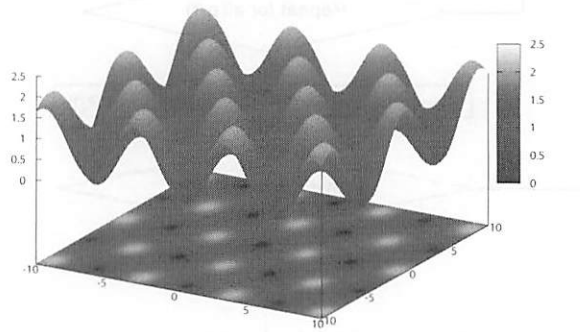

Fig. 5　Griewank Function in two-dimension (Near optimum)

Table 1　Common parameters

Number of design variables	20
Population size	50
Selection operater	Roulette selection
Crossover operator	BLX-α, $\alpha = 0.5$
Crossover rate	1.0
Mutation rate	0.002
Max. number of generation	10

Table 2　SOM-GA parameters

SOM Map	Hexagon (20×20)
Init. neighborhood radius	$R = 10$
Training rate	$\alpha_s = 0.8$
Number of training	$TS = 1000$
Neighborhood radius for sub-pop.	$Rn = 2$
Max. generations in sub-pop.	$N_g = 1000$

Table 3　Maximum generation step

	Rastrigin	Rosenbrock	Griewank
RCGA	1,000,000	20,000,000	1,000,000
SOM-GA	20	200	20

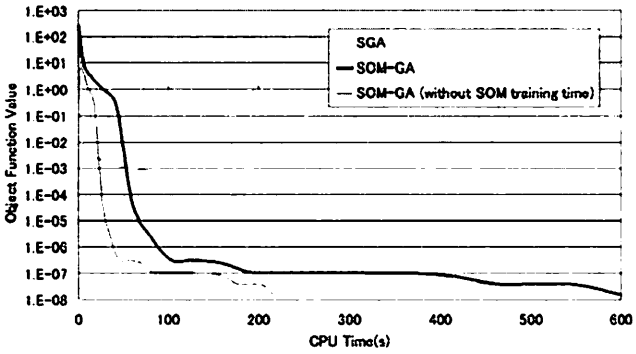

Fig. 6 History of best solution (Rastrigin function)

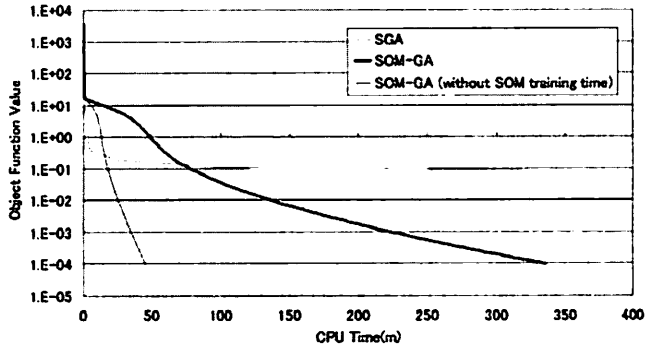

Fig. 7 History of best solution (Rosenbrock function)

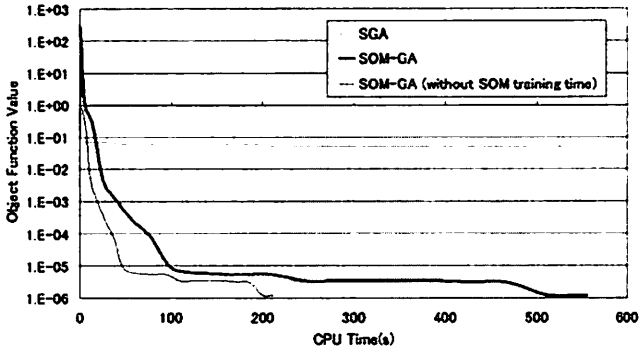

Fig. 8 History of best solution (Griewank function)

して BLX-α を用いる．また，本研究で用いる突然変異では，設計変数の定義域内でランダムに値を変更する方法を用いる．これらのパラメータは他の研究，数値実験などを通じて定めている．

SOM-GA のパラメータ設定を Table 2 に示す．ここで，部分集団の近傍半径 $Rn = 2$ は，各個体を含む部分集団を定義するための，自己組織化マップ上でのユークリッド距離である．部分集団進化世代数 $N_g = 1000$ は，部分集団において RCGA を実行する回数を示す．

各手法における最大世代数を Table 3 に示す．SOM-GA では，1 つの世代で 1 個体ごとに部分集団を構成し，そこで $N_g = 1000$ 回の RCGA を実行する．今回は個体数 50 なので，1 つの世代ごとに $50 \times 1000 = 50,000$ 回の RCGA を行うことになる．そこで，RCGA と SOM-GA の進化的処理の回数をそろえるために，

$$\text{RCGA の最大世代数} = \text{SOM-GA の最大世代数} \times 50,000$$

となるように設定している．

4.3. 実験結果

Rastrigin 関数

解析結果を Fig.6 に示す．縦軸には各世代の最良個体の目的関数値を，横軸には計算に使用した CPU 時間を示している．グラフには，RCGA の収束状況，本研究で提案した SOM-GA の収束状況に加えて，SOM-GA の処理時間から SOM 学習時間を除いた場合の結果を示す．この図から，RCGA は最初の収束が速いが，目的関数値が 0.001 あたりをすぎてから収束速度が遅くなることがわかる．SOM-GA では最初の収束速度は RCGA に劣るものの，目的関数値が 0.001 あたりから 0.0000001 あたりまでの収束速度が速く，CPU 時間が 50 秒あたりをすぎると，RCGA よりかなり大域的最適解に近い値に収束することがわかる．

SOM の学習時間を除くと目的関数値が 0.0001 あたりで計算時間が RCGA を上回っていることがわかる．

Rosenbrock 関数

解析結果を Fig.7 に示す．縦軸には各世代の最良個体の目的関数値を，横軸には計算に使用した CPU 時間を単位を分として示している．Fig.7 からわかるように，RCGA では 0.1 あたりまでしか収束できないのに対して，SOM-GA では 0.0001 ぐらいまで収束できることがわかる．

しかし，先の例題に比べると計算時間は長いことがわかる．これは Rosenbrock 関数が単峰性であり，提案した手法では近傍個体で部分集団を構成して探索を行うので，最適解に到達するまでに毎世代で少しづつしか進化できないため，収束速度が遅くなると考えられる．

SOM の学習時間をみると，他の 2 つの関数に比べて多くの時間が必要であることがわかる．

Griewank 関数

解析結果を Fig.8 に示す．縦軸には各世代の最良個体の目的関数値を，横軸には計算に使用した CPU 時間を示している．この問題では，提案した SOM-GA の収束速度が，RCGA に比べてかなり速いことがわかる．そして，RCGA では目的関数値が 0.1 あたりからあまり収束していないが，SOM-GA では目的関数値が 0.00001 あたりまで収束できることがわかる．

Griewank 関数は設計変数間に依存関係を有する多峰性関数である．大域的には単峰性関数のような性質を持つため準最適解は比較的容易に求めることができるが，局所的には多数の準最適解が存在し，最適解を発見するのは困難となる．SOM-GA では，自己組織化マップの学習により，多峰性関数の設計変数間の近傍関係を容易に発見することができ，そのために効率的に最適解を探索することができたと考えられる．

この場合，SOM の学習時間は Rosenbrock 関数よりも少

なく，Rastrigin 関数と同程度もしくはやや短いといえる．

5. まとめ

本研究では，多峰性のある連続関数の最適化問題に対する進化的計算法として，自己組織化マップを用いた遺伝的アルゴリズムである Self-Organizing Maps for Genetic Algorithms (SOM-GA) を提案した．提案手法では，各世代において集団で自己組織化マップを学習し，集団から複数の部分集団を構成する．そして，部分集団に実数値遺伝的アルゴリズム (RCGA) を適用して新たな個体を生成する．この処理を繰り返すことで解を探索する．

本研究では Rastrigin，Rosenbrock，Griewank の三つのテスト関数に対して，提案した SOM-GA とそのもととなった RCGA を比較した．計算結果より，すべての関数において提案手法が RCGA より良い最適解を探索できたことがわかった．今後は，自己組織化マップの学習時間を減少するためにアルゴリズムを改良する必要があると考えている．さらに，実用的な問題に対する手法の応用についても研究を進めていく予定である．

参考文献

(1) L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval schemata. In L. D. Whitley, editor, *Foundation of Genetic Algorithms 2*, pp. 187–202. Morgan Kaufmann Publications, 1992.

(2) T. コホネン (著), 徳高平蔵, 岸田悟, 藤村喜久郎 (訳). 自己組織化マップ. シュプリンガー・フェアラーク東京, 1996.

(3) 徳高平蔵, 岸田悟, 藤村喜久郎. 自己組織化マップの応用. 海文堂, 1999.

(4) マーク M. ヴァン・フッレ (著), 徳高平蔵, 岸田悟, 藤村喜久郎 (訳). 自己組織化マップ—理論・設計・応用. 海文堂, 2001.

(5) D. Buche, M. Milano, and P. Koumoutsakos. Self-organizing maps for multi-objective optimization. In A.M. Barry, editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, New York, AAAI*, pp. 152–155, 2002.

(6) 沈侃, 北栄輔. 自己組織化マップを用いた進化的最適化手法の性能評価について. 情報処理学会研究報告, No. 56, pp. 5–8, 2006.

(7) K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, 2000.

(8) E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou et al, editor, *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. Athens, Greece, Septermber 12-21, 2001*, pp. 21–24, 2001.