実数値型確率的スキーマ貪欲法について

丸山 崇[†] 北 栄輔[††]

確率的スキーマ貪欲法（Stochastic Schemata Exploiter: SSE）は，本来 0/1 組合せ最適化問題を対象に提案された探索手法である．本研究では，SSE を実数値最適化問題に適用するための実数値確率的スキーマ貪欲法（Real-coded SSE: RSSE）を提案する．解析例として複数の実数値テスト関数の最適化問題を考え，実数値 GA（Real-coded Genetic Algorithms: RGA）と探索性能を比較する．その結果，RSSE は，優れた収束速度と RGA と同程度の探索性能を実現できることが分かった．

Investigation of Real-valued Stochastic Schemata Exploiter

Takashi Maruyama[†] and Eisuke Kita[††]

The Stochastic Schemata Exploiter (SSE) was presented for solving 0/1 combinatorial optimization problems alone. This paper describes the development of SSE to the real-coded optimization problem. The algorithm is named as Real-coded SSE (RSSE). The RSSE are compared with the Real-coded Genetic algorithm (RGA) in real-valued problem. As a result, we indicate that RSSE has an excellent convergence property and the global search ability.

1. はじめに

進化的計算手法の 1 つに相澤らが提案した確率的スキーマ貪欲法（Stochastic Schemata Exploiter: SSE）[1] がある．SSE は，0/1 組合せ最適化問題を対象に提案された探索手法である．著者らは，以前の研究において，0/1 組合せ最適化問題における SSE の優れた探索性能を確認し，探索性能の向上を実現した改良アルゴリズムの研究を行った[2),3)]．

一方，実際の最適化問題では，トラス構造物の最適設計[4]，航空機などの空力構造物の最適設計[5]，レンズの最適設計[6]，蛋白質の構造解析[7] など実数値を設計変数に持つ実数値最適化問題が多くある．このような実問題における実数値最適化問題は局所解が多数存在する場合が多く，勾配を用いた解法では，局所解へ収束してしまい，最適解を探索できない場合がある．また，設計変数と目的関数が連続値をとらない場合は，勾配を評価すること自体困難な場合がある．そこで，このような実数値最適化問題においても，良い精度の解を効率良く探索できる遺伝的アルゴリズムが研究されてきた[4)~7)]．

今後，SSE を各種の実問題に適用するためには，実数値最適化問題への拡張が必要がある．そこで，本論文では，SSE を実数値最適化問題に拡張した実数値 SSE（Real-coded SSE: RSSE）を提案する．

ところで，実数値最適化問題への進化的計算法の適用研究として，実数値 GA（Real-coded Genetic Algorithms: RGA）がある．喜多ら[8] は，個体集団が最適解を探索できる有望な領域に分布しているならば，「交叉により生成される子個体のサンプリングは親個体の分布の平均値ベクトルや分散・共分散行列を継承することが望ましい」と述べている．この方針により構築された交叉手法が，SPX 交叉[9] と UNDX-m 交叉[10] である．SPX，UNDX-m は，子個体に親個体の統計量を遺伝させることで，親個体の分布している領域を重点的にサンプリングする．そこで，これらに個体集団の多様性を維持することができる MGG[11] を組み合わせることで，探索空間全体に分布した個体を徐々に収束させて最適解を探索できるアルゴリズムを提案している．

本研究で提案する RSSE は進化的計算法であるが，RGA とは異なるアルゴリズムを持っている．評価値順に並べた個体から部分集団を生成し，部分集団を構成する個体群から子個体を生成する．このとき，評価値

† ソネットエンタテインメント（株）
So-net Entertainment Corporation
†† 名古屋大学大学院情報科学研究科複雑系科学専攻
Graduate School of Information Science, Nagoya University

の高い個体の方が異なる部分集団に繰り返し選択される回数が多くなっている．つまり，良い個体の周辺を繰り返し探索することで，より良い個体を精度良く探索することを目的とした探索法である．そのため，親個体をランダムに選択する RGA と比べて，RSSE は個体集団の多様性が急速に小さくなる．そこで，SPX 交叉と UNDX-m 交叉を用いることで，子個体が適切に選択されるようにし，また，突然変異率を RGA よりも大きくするようにしている．

2 章では，実数値交叉手法と RGA について述べる．3 章では，RSSE について述べる．4 章では，性能比較に用いる 0/1 組合せ最適化問題の概要を述べる．そして，5 章において，RGA，RSSE の探索性能を比較する．最後に，6 章において，本論文のまとめを述べる．

2. RGA について

2.1 RGA と実数値交叉の概要

実数値問題において，効率良く解探索を行う実数値 GA（Real-coded Genetic Algorithms: RGA）が研究されている．RGA では，設計変数を実数ベクトル表現する[12] ので，実数値ベクトルを扱う交叉手法が必要であり，これまでの研究において，多くの実数値交叉手法が提案されている[9),10),13),14)]．

RGA の代表的な交叉手法の 1 つに，BLX-α 交叉がある．BLX-α[13] は，親個体の設計変数の定義域を両側に α だけ拡張した区間から一様分布に従ってランダムに子個体を生成する実数値交叉手法である．

UNDX 交叉[14] は，2 つの親個体を結ぶ直線上およびその近傍に，正規分布に従ってランダムに子個体を生成する．

SPX 交叉[9]，UNDX-m[10] 交叉は，多数の親個体を用いた実数値交叉手法である．SPX，UNDX-m は，いくつか提案されている実数値交叉手法の中で，優れた探索性能を示している．SPX は，多数の親個体が張る多面体を ϵ 倍に相似変換した図形の内部に一様分布に従って子個体を生成する．UNDX-m は，UNDX を多親型に拡張した手法であり，$m+1$ 個の親個体の重心の中心にして，正規分布に従ってランダムに子個体を生成する．

2.2 SPX の概要と手順

SPX 交叉[9] は，$m=n+1$ 個の親個体から，n 次元の探索成分を求め，これを基底ベクトルとした一様乱数により子個体を生成する．子個体は，$m+1$ 個体の親個体が張る m 次元のシンプレクスの重心を中心にして，ϵ 倍に相似変換した図形の内部に一様分布に生成されることになる．

$m+1$ 個の親個体の重心の位置座標を $\mathbf{G}$ とおく．子個体の基になる $\mathbf{x}_k$，$\mathbf{C}_k$ を次式で求める．

$$\mathbf{x}_k = \mathbf{G} + \epsilon(\mathbf{p}_k - \mathbf{G}) \quad (k=1,\ldots,m+1) \tag{1}$$

$$\mathbf{C}_k = r_{k-1}(\mathbf{x}_{k-1} - \mathbf{x}_k + \mathbf{C}_{k-1}) \quad (k=2,\ldots,m+1, \mathbf{C}_1=\mathbf{0}) \tag{2}$$

$$r_k = (u(0,1))^{\frac{1}{k}} \quad (k=1,\ldots,m) \tag{3}$$

そして，子個体 $\mathbf{x}^c$ は次式で求める．

$$\mathbf{x}^c = \mathbf{x}_{m+1} + \mathbf{C}_{m+1} \tag{4}$$

$u(0,1)$ は，区間 $[0,1]$ の一様実数乱数である．ϵ は，個体を分布する多面体の拡張率であり，本研究では，Higuchi らの研究[9] において推奨されている $\epsilon = \sqrt{m+2}$ を用いる．

2.3 UNDX-m の概要と手順

UNDX-m[10] では，$m+2$ 個の親個体から，m 次元の主探索成分とこれに直交する（$n-m$）次元の副探索成分を基底ベクトルとした正規乱数により子個体を生成する実数値交叉手法である．$m+2$ 個目の親個体は副探索成分におけるスケーリングに用いられる．子個体は，$m+1$ 個の親個体の重心の中心にして，$m+1$ 個の親が張る m 次元の部分空間内に正規分布に生成される．

$m+1$ 個の親個体の重心の位置座標を $\mathbf{G}$，親個体と重心の差ベクトルを $\mathbf{d}_i$ とおき，$m+2$ 個目の親個体と重心の差ベクトル $\mathbf{d}_{m+2}$ の $\mathbf{d}_1,\ldots,\mathbf{d}_m$ に直交する成分の大きさを D とする．$\mathbf{d}_1,\ldots,\mathbf{d}_m$ に直交する部分空間の正規直交基底を $\mathbf{e}_1,\ldots,\mathbf{e}_{n-m}$ とおく．子個体 $\mathbf{x}^c$ は次式で求める．

$$\mathbf{x}^c = \mathbf{G} + \sum_{i=1}^{m} w_i \mathbf{d}_i + \sum_{i=1}^{n-m} v_i D e_i \tag{5}$$

ここで，w_i，v_i はそれぞれ $N(0,\sigma_\xi^2)$，$N(0,\sigma_\eta^2)$ に従う正規乱数であり，σ_ξ，σ_η は次式で定義される．

$$\sigma_\xi = \frac{\alpha}{\sqrt{m}} \tag{6}$$

$$\sigma_\eta = \frac{1}{\sqrt{n-m}} \frac{\sqrt{m+1}}{\sqrt{m+2}} \frac{\sqrt{3}}{\sqrt{2}} \beta \tag{7}$$

本研究では，Kita らの研究[10] において推奨されている $\alpha=1$，$\beta=0.50$ を用いる．

2.4 RGA のアルゴリズム

多くの研究で提案されてきた世代交代モデルの中で，MGG[11] の世代交代モデルに基づく GA は，優れた大域的探索性能を示しており，Higuchi ら[9]，Kita ら[10] の研究においても採用されている．本研究においても，MGG に基づく GA に SPX，UNDX-m 交叉を適用し

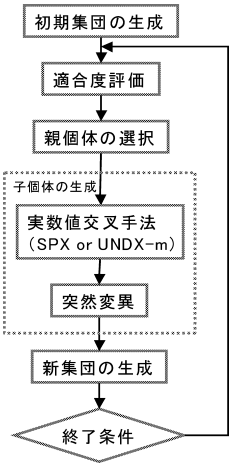

図 1 RGA のフローチャート
Fig. 1 RGA flowchart.

た RGA を考える．

SPX，UNDX-m 交叉と MGG 世代交代モデルを用いた RGA のアルゴリズムを以下に述べる．また，フローチャートを**図 1** に示す．

(1) 初期集団の生成
初期個体 M 個をランダムに生成する．

(2) 個体の評価計算
各個体の適合度を評価する．

(3) 親個体の選択
母集団中からランダムに $m+1$ 個の親個体を選ぶ．このとき，同じ個体は選択しない．

(4) 子個体の生成
- **SPX 交叉を用いる場合**
$m+1$ 個の親個体による SPX 交叉を M 回実行して，M 個の子個体を生成する．また，突然変異を施す．
- **UNDX-m 交叉を用いる場合**
$m+1$ 個の親個体による UNDX-m 交叉を M 回実行して，M 個の子個体を生成する．また，突然変異を施す．

(5) 次世代に残す個体候補の集団 p の生成
$m+1$ 個の親個体からランダムに 2 個体を選択する．このとき，同じ個体は選択しない．選択した 2 個の親個体と M 個の子個体から集団 p を生成する．

(6) 生存選択
集団 p より，最良個体 1 個体とルーレット選択により選択した 1 個体を母集団に戻す．

(7) 終了条件
停止条件が満たされるまで，(2) から (6) を繰り返す．

それぞれの交叉の親個体数 $m+1$ は，Higuchi ら，Kita らの研究で用いられている値を用いる．Higuchi らの研究[9] では，SPX は，m は問題の次元数 n とすることを推奨している．そこで，RGA の SPX では，m は n とする．また，Kita らの研究[10] では，UNDX-m は，$m=2$，4 の場合において良い探索性能を示している．そこで，RGA の UNDX-m では，m は 2 とする．

また，突然変異では，各設計変数において，あらかじめ与えた突然変異率の確率で，各設計変数の値を，その定義域内から一様乱数により選択した実数値に変更することで行う．

3. RSSE について

3.1 RSSE の概要

SSE では，まず多数の良い個体を要素とする個体部分集合を生成する．次に，個体部分集合から抽出される共通スキーマを基にして子個体を生成する．これにより，SSE は，多数の良い親個体と似通った構造を持つ子個体を生成している．しかし，実数値最適化問題に SSE を適用する場合，設計変数が実数値表現されているため，スキーマに相当する共通構造を得ることができない．よって，SSE を実数値に拡張するにあたって，SSE の個体生成のアルゴリズムを根本的に改良しなければならない．

ところで，SPX 交叉[9] と UNDX-m 交叉[10] は，多数の親個体の構造を基にして子個体を生成するアルゴリズムである．そこで，これらの実数値交叉手法を SSE の個体部分集合に適用することで，スキーマに相当する構造がない実数値最適化問題において，多数の良い親個体から似通った解構造を持つ子個体を生成することができる．

3.2 RSSE のアルゴリズムの手順

SPX，UNDX-m 交叉を用いた RSSE のアルゴリズムを以下に述べる．また，フローチャートを**図 2** に示す．

(1) 初期集団の生成
初期個体 M 個をランダムに生成する．

(2) 個体の評価計算
各個体の適合度を評価する．

(3) 個体部分集合の生成
個体を適合度の降順に並べ，半順序関係に従って高い適合度の個体を要素に持つ，平均評価値の高い上位 M 個の個体部分集合を生成する（詳細は 3.3 節において述べる）．

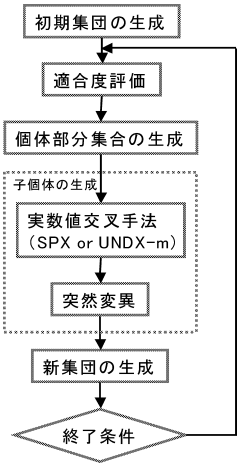

図 2 RSSE のフローチャート
Fig. 2 RSSE flowchart.

(4) 親個体の選択と子個体の生成
- **SPX 交叉を用いる場合**
  M 個の個体部分集合において，個体部分集合の要素である個体を親個体とし，それぞれ SPX 交叉を実行して M 個の子個体を生成する．また，突然変異を施す．
- **UNDX-m 交叉を用いる場合**
  M 個の個体部分集合において，個体部分集合の要素である個体を親個体とし，それぞれ UNDX-m 交叉を実行して M 個の子個体を生成する．また，突然変異を施す．

(5) 世代交代
生成された M 個の子個体を次世代の母集団とする．

(6) 終了条件
停止条件が満たされるまで，(2) から (5) を繰り返す．

RGA では，$m+1$ 個の親個体を母集団から選択している．しかし，RSSE では，すべての個体部分集合が $m+1$ 個の親個体により構成されているとは限らない．そこで，RSSE では，個体部分集合の個体数が $m+1$ 個でない場合でも交叉が行えるように改良しなければならない．

まず，Higuchi ら[9] は，m に様々な値をとった縮退型 SPX を提案している．そこで，RSSE で SPX 交叉を用いる場合には，Higuchi らの提案を応用することにする．

また，UNDX-m 交叉では，親個体が 1 個体の場合では UNDX-m は適用できず，2 個体の場合では UNDX-1（UNDX[14] に相当）を適用することになる．さらに，Kita らの研究[10] では，m が 2，もしくは，4 の場合において良い探索性能を示しており，m が大きい場合では探索性能が低下している．そこで，RSSE に UNDX-m 交叉を用いる場合には，親個体を以下のように選択する．ここで，個体部分集合の親個体の個数を l とおく．

- $l > m+1$
  l 個の個体の中から，$m+1$ 個の個体をランダムに選択して親個体とする．このとき，同じ個体は選択しない．
- $l = m+1$
  l 個の全個体を親個体とする．
- $l < m+1$
  l 個の全個体を親個体とし，母集団中からランダムに $m+1-l$ 個の個体を選択し親個体として追加する．このとき，同じ個体は追加しない．

ここで，RSSE の UNDX-m では，RGA と同じく，m は 2 とする．

また，突然変異では，各設計変数において，あらかじめ与えた突然変異率の確率で，各設計変数の値を，その定義域内から一様乱数により選択した実数値に変更することで行う．

3.3 個体部分集合の生成

RSSE で生成する個体部分集合は，それを構成する個体集団の平均評価値の高い順に作成する．そのために，平均評価値における半順序関係を用いる．

3.3.1 平均評価値の半順序関係

M 個の個体をその適合度の降順に並べたインデックス（番号付け）を $c_1, c_2, \ldots, c_M$ とおく．また，任意の個体部分集合 S（$\neq \phi$）について，その中に含まれる最大のインデックスを $L(S)$ で表す．$(S - c_k)$ は，集合 S から要素 c_k を除いた集合とする．さらに，和集合を $\cup$ で表す．$L(S) < M$ のとき，個体部分集合の間に以下の半順序関係が存在する．

- S の平均評価値は $S \cup c_{(L(S)+1)}$ の平均評価値よりも良い．
- S の平均評価値は $(S - c_{L(S)}) \cup c_{(L(S)+1)}$ の平均評価値よりも良い．

3.3.2 個体部分集合の生成手順

最も平均評価値の高い個体部分集合は，最良個体 c_1 だけからなる集合 $\{c_1\}$ である．半順序関係によれば，$\{c_1\}$ の次に平均評価値の高い部分集合として，$\{c_1, c_2\}$ と $\{c_2\}$ ができる．さらに，$\{c_1, c_2\}$ からは $\{c_1, c_2, c_3\}$ と $\{c_1, c_3\}$ が生成でき，$\{c_2\}$ からは $\{c_2, c_3\}$ と $\{c_3\}$ が生成できる．このようにして生成した個体部分集合を平均評価値の降順にソートしてできたリストの例が

表 1 個体部分集合リストの例
Table 1 Example of subset list.

Rank	Subset
1	$\{c_1\}$
2	$\{c_1, c_2\}$
3	$\{c_2\}$
4	$\{c_1, c_2, c_3\}$
5	$\{c_1, c_3\}$
⋮	⋮

表 2 テスト問題の特徴
Table 2 Features of test functions.

関数	設計変数間の依存関係	関数の形状
Sphere 関数	なし	単峰性
Rastrigin 関数	なし	多峰性
Schwefel 関数	なし	多峰性
Ridge 関数	あり	単峰性
Rosenbrock 関数	あり	単峰性
Griewank 関数	あり	多峰性

表 1 である．

この処理を箇条書きにすると次のようになる．

(1) リストの先頭の要素に $\{c_1\}$ を格納する．$i = 1$ とする．

(2) リストの i 番目の要素において，格納されている個体部分集合から，半順序関係に従って 2 つの新しい個体部分集合を生成する．

(3) 生成した 2 つの個体部分集合を，$i+1$ 番目以降の要素に平均評価値の降順に従って格納する．リストからあふれた要素は捨てる．

(4) $i < M-1$ ならば $i = i+1$ として，(2) から (3) を繰り返す．そうでなければ終了する．

4. テスト問題

実際の実数値最適化問題は，設計変数間に依存関係があり局所解が多数存在する場合が多い．実問題の有効な解探索手法として発展させるためには，まず，設計変数間の依存関係の有無や単峰性，多峰性の特徴などを組み合わせた実数値テスト問題を用いて，解の探索精度や探索特性を検討する必要がある．そこで，本研究では，これらの特徴を組み合わせた実数値テスト問題として，Sphere 関数，Rastrigin 関数，Schwefel 関数，Ridge 関数，Rosenbrock 関数，Griewank 関数[15),16)] を用いて RSSE と RGA の解探索性能を比較する．これらの関数は，最小値が 0 となる関数最小化問題である．これらの関数には**表 2** に示すような特徴がある．

4.1 Sphere 関数

Sphere 関数は，単峰性の関数であり，設計変数間に依存関係はない．

$$F_{sp}(x) = \sum_{i=1}^{n} x_i^2 \quad (-5.12 < x_i < 5.12) \tag{8}$$

$x_i = 0$ （$i = 1, \cdots, n$）のとき最小値 0 をとる．n は設計変数の数であり，本研究では $n = 10$ とする．

4.2 Rastrigin 関数

Rastrigin 関数は，大域的最適解の周辺に格子状に多数の局所解を持つ多峰性の関数であり，設計変数間に依存関係はない．

$$F_{ra}(x) = 10n + \sum_{i=1}^{n} \left\{ x_i^2 - 10\cos(2\pi x_i) \right\} \quad (-5.12 \le x_i < 5.12) \tag{9}$$

設計変数の値が $x_i = 0$ （$i = 1, \cdots, n$）のとき最小値 0 をとる．n は設計変数の数であり，本研究では $n = 10$ とする．

4.3 Schwefel 関数

Schwefel 関数は，定義域の境界付近に最適解を持つ多峰性の関数であり，設計変数間に依存関係はない．

$$F_{sc}(x) = \sum_{i=1}^{n} -x_i \sin\left\{ \sqrt{|x_i|} \right\} + 418.98288727 \cdot n \quad (-512 \le x_i < 512) \tag{10}$$

設計変数の値が $x_i = 420.968750$（$i = 1, \cdots, n$）のとき最小値 0 をとる．定義域の境界付近に最適解を持つので，局所解に収束する可能性が高い．n は設計変数の数であり，本研究では $n = 10$ とする．

4.4 Ridge 関数

Ridge 関数は単峰性の関数であり，設計変数間に強い依存関係が存在する．

$$F_{ri}(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \quad (-64 \le x_i < 64) \tag{11}$$

設計変数の値が $x_i = 0$（$i = 1, \cdots, n$）のとき最小値 0 をとる．n は設計変数の数であり，本研究では $n = 10$ とする．

4.5 Rosenbrock 関数

Rosenbrock 関数は単峰性の関数であり，設計変数間に依存関係がある．

$$F_{ro}(x) = \sum_{i=1}^{n-1} \left\{ 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right\} \quad (-2.048 \le x_i < 2.048) \tag{12}$$

表 3 突然変異率
Table 3 Mutation rate.

関数	RGA-SPX	RGA-UNDX-2	RSSE-SPX	RSSE-UNDX-2
Sphere 関数	0.00e-0	0.00e-0	1.00e-1	0.00e-0
Rastrigin 関数	0.00e-0	5.00e-5	2.65e-1	7.00e-3
Schwefel 関数	1.20e-4	5.00e-5	1.70e-1	5.00e-3
Ridge 関数	2.00e-6	0.00e-0	1.40e-1	5.00e-6
Rosenbrock 関数	1.00e-5	5.00e-6	2.85e-1	3.00e-3
Griewank 関数	1.00e-3	1.00e-4	1.00e-1	1.00e-3

表 4 解を探索できた回数（Sphere 関数の場合）
Table 4 Number of searchable trials (Sphere function).

	RGA-SPX	RGA-UNDX-2	RSSE-SPX	RSSE-UNDX-2
閾値	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250
1.0e-2	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
1.0e-5	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
1.0e-8	0 / 30 / 0	30 / 30 / 0	30 / 30 / 12	30 / 30 / 30

設計変数の値が $x_i = 1$（$i = 1, \cdots, n$）のとき最小値 0 をとる．単峰性の関数であるが，準最適解が湾曲した形状に分布している非凸関数である．n は設計変数の数であり，本研究では $n = 10$ とする．

4.6 Griewank 関数

Griewank 関数は大域的には単峰性の形状であるが，多数の局所解が存在する多峰性の関数であり，設計変数間に依存関係がある．

$$F_{gr}(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \Big(\cos\big(\frac{x_i}{\sqrt{i}}\big) \Big) \quad (-512 \leq x_i < 512) \tag{13}$$

設計変数の値が $x_i = 0$（$i = 1, \cdots, n$）のとき最小値 0 をとる．設計変数間に依存関係が存在し，多峰性の関数であるので，最適解を得るのはきわめて困難である．n は設計変数の数であり，本研究では $n = 10$ とする．

5. 探索性能の比較

前章で述べた実数値最適化問題を用いて，RGA，RSSE の探索性能を比較する．RGA，RSSE ともに，実数値交叉として SPX 交叉，または，UNDX-m 交叉を用いた探索性能を示す．UNDX-m 交叉における m は，RGA，RSSE ともに 2 とする．RGA，1 世代に母集団の個体数に相当する M 個の子個体を生成する．RGA，RSSE ともに，個体数を 50，100，250 とる．各探索手法ともに 1 世代あたりの適合度評価による計算負荷は同じであるので，実行世代数で探索性能を評価することが可能となる．また，RGA，RSSE ともに，突然変異率を変えて十分に実験し，最も良い性能を示した値を用いる（**表 3**）．

各問題において，各探索手法を異なる初期個体集団から 30 回実行し平均値をとり，各探索手法の探索性能とする．また，各関数において実行世代数は，Sphere 関数は 3000，Rastrigin 関数は 20000，Schwefel 関数は 30000，Ridge 関数は 20000，Rosenbrock 関数は 50000，Griewank 関数は 30000 とする．

探索性能を比較するために，2 つの方法で比較を行う．1 つは，異なる初期集団から行った 30 回の試行において，解を得ることができた回数である．解を得たと判定できる閾値に異なる値をとって比較する．そして，もう 1 つは，各世代における最良個体の収束曲線である．

5.1 Sphere 関数の実験結果

閾値を 1.0e-2，1.0e-5，1.0e-8 として解を探索し，閾値を下回る解を探索できた回数を**表 4** に示す．P50，P100，P250 は，それぞれ個体数 50，100，250 を表す．また，各個体数における，最良個体の解の平均値のグラフを**図 3**，**図 4**，**図 5** に示す．各グラフにおいて，横軸に世代数をとり，縦軸に最良個体の解の平均値をとる．

表 4 より，閾値が大きい場合は，RGA と RSSE の探索性能に違いはないが，閾値を 1.0e-8 と小さくすると，RGA よりも RSSE のほうが解を探索できた回数が多くなっており，良い解を探索できていることが分かる．

図 3〜5 より，すべての個体数において，RSSE-UNDX-2 が他の手法と比べて速い収束速度を示していることが分かる．特に，個体数 250 では，RSSE-UNDX-2 は他の手法と比べて非常に優れた収束速度を示している．また，個体数 100，250 において，RSSE-SPX は RGA よりも速い収束速度を示していること

表 5 解を得ることができた回数（Rastrigin 関数の場合）
Table 5 Number of searchable trials (Rastrigin function).

	RGA-SPX	RGA-UNDX-2	RSSE-SPX	RSSE-UNDX-2
閾値	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250
1.0e-2	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
1.0e-5	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
1.0e-8	28 / 26 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30

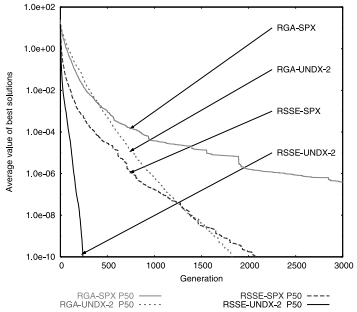

図 3 Sphere 関数における到達解の平均値（個体数 50）
Fig. 3 Mean fitness of best solution (Sphere function, Pop. size = 50).

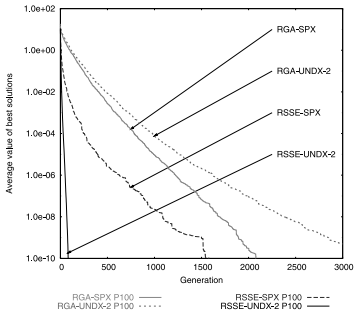

図 4 Sphere 関数における到達解の平均値（個体数 100）
Fig. 4 Mean fitness of best solution (Sphere function, Pop. size = 100).

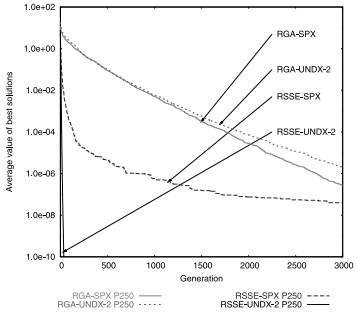

図 5 Sphere 関数における到達解の平均値（個体数 250）
Fig. 5 Mean fitness of best solution (Sphere function, Pop. size = 250).

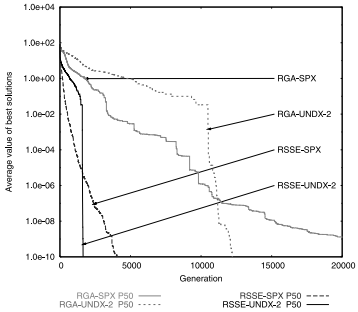

図 6 Rastrigin 関数における到達解の平均値（個体数 50）
Fig. 6 Mean fitness of best solution (Rastrigin function, Pop. size = 50).

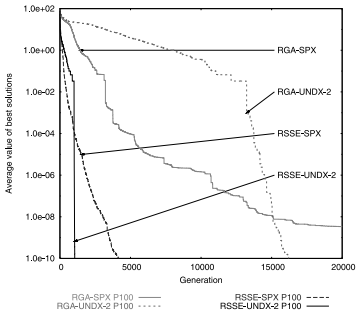

図 7 Rastrigin 関数における到達解の平均値（個体数 100）
Fig. 7 Mean fitness of best solution (Rastrigin function, Pop. size = 100).

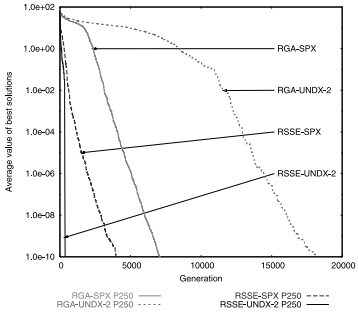

図 8 Rastrigin 関数における到達解の平均値（個体数 250）
Fig. 8 Mean fitness of best solution (Rastrigin function, Pop. size = 250).

が分かる．

5.2 Rastrigin 関数の実験結果

解を探索することができた回数を**表 5** に示す．各個体数における最良個体の解の平均値のグラフを**図 6**, **図 7**, **図 8** に示す．

表 5 より，閾値が大きい場合 RGA と RSSE は同等

表 6 解を探索できた回数（Schwefel 関数の場合）
Table 6 Number of searchable trials (Schwefel function).

閾値	RGA-SPX P50 / P100 / P250	RGA-UNDX-2 P50 / P100 / P250	RSSE-SPX P50 / P100 / P250	RSSE-UNDX-2 P50 / P100 / P250
1.0e-2	30 / 30 / 30	30 / 30 / 6	30 / 30 / 30	30 / 30 / 30
1.0e-5	30 / 30 / 30	30 / 30 / 4	30 / 30 / 30	30 / 30 / 30
1.0e-8	30 / 16 / 30	30 / 30 / 2	30 / 30 / 6	30 / 30 / 30

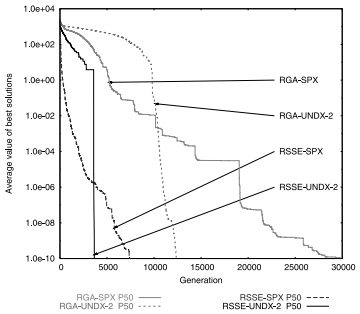

図 9 Schwefel 関数における到達解の平均値（個体数 50）
Fig. 9 Mean fitness of best solution (Schwefel function, Pop. size = 50).

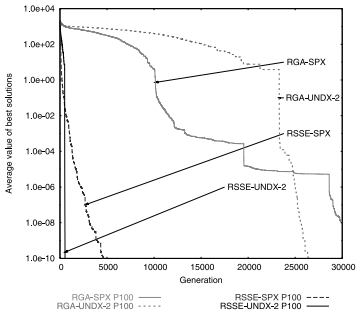

図 10 Schwefel 関数における到達解の平均値（個体数 100）
Fig. 10 Mean fitness of best solution (Schwefel function, Pop. size = 100).

の探索性能を示しているが，閾値を 1.0e-8 とすると，個体数 50 および個体数 100 の SPX において，閾値 1.0e-8 より多く解を探索できており，良い探索性能を示していることが分かる．

図 6〜8 より，すべての個体数において，RSSE-UNDX-2 が他の手法と比べて速い収束速度を示していることが分かる．特に，個体数 250 では，RSSE-UNDX-2 は他の手法と比べて非常に優れた収束速度を示している．また，RSSE-SPX は，RSSE-UNDX-2 に次いで収束速度が速いことが分かる．

5.3 Schwefel 関数の実験結果

解を探索することができた回数を**表 6** に示す．各個体数における，最良個体の解の平均値のグラフを**図 9**，**図 10**，**図 11** に示す．

表 6 より，閾値が大きい場合は，RSSE は RGA と同程度の探索性能を示していることが分かる．閾値を 1.0e-8 とすると，RSSE-UNDX-2 は他のアルゴリズムより多く探索できており，良い探索性能を示していることが分かる．

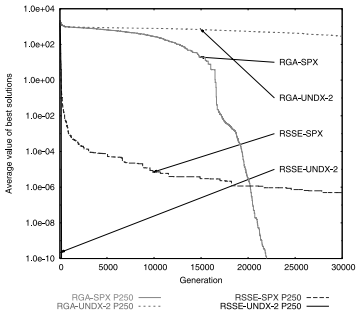

図 11 Schwefel 関数における到達解の平均値（個体数 250）
Fig. 11 Mean fitness of best solution (Schwefel function, Pop. size = 250).

図 9〜11 より，すべての個体数において RSSE-UNDX-2 が他の手法と比べて速い収束速度を示していることが分かる．特に，個体数 250 では，RSSE-UNDX-2 は他の手法と比べて非常に優れた収束速度を示している．RSSE-SPX は，個体数 50，100 では RSSE-UNDX-2 に次いで速い収束速度を示しているが，個体数 250 では 1.0e-4 付近から収束速度が遅くなっていることが分かる．

5.4 Ridge 関数の実験結果

解を探索することができた回数を**表 7** に示す．各個体数における，最良個体の解の平均値のグラフを**図 12**，**図 13**，**図 14** に示す．

表 7 より，閾値が大きい場合は，RSSE は RGA と同程度の探索性能を示していることが分かる．閾値を 1.0e-8 と小さくすると，RGA-UNDX-2 と RSSE-UNDX-2 がすべての試行で解を探索できていることが分かる．

図 12〜14 より，すべての個体数において，RSSE-UNDX-2 が他の手法と比べて速い収束速度を示していることが分かる．特に，個体数 250 では，RSSE-UNDX-2 は他の手法と比べて非常に優れた収束速度を示している．また，RSSE-SPX は，個体数 100，250 において，他の手法と比べ収束速度が遅いことが分

表 7 解を得ることができた回数（Ridge 関数の場合）
Table 7 Number of searchable trials (Ridge function).

閾値	RGA-SPX P50 / P100 / P250	RGA-UNDX-2 P50 / P100 / P250	RSSE-SPX P50 / P100 / P250	RSSE-UNDX-2 P50 / P100 / P250
1.0e-2	30 / 30 / 30	30 / 30 / 30	30 / 30 / 0	30 / 30 / 30
1.0e-5	30 / 30 / 30	30 / 30 / 30	30 / 30 / 0	30 / 30 / 30
1.0e-8	0 / 30 / 30	30 / 30 / 30	30 / 30 / 0	30 / 30 / 30

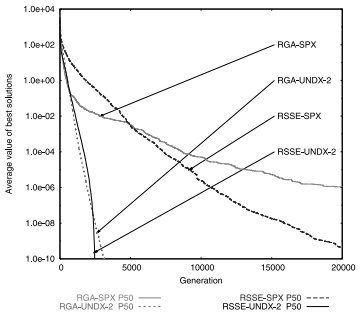

図 12 Ridge 関数における到達解の平均値（個体数 50）
Fig. 12 Mean fitness of best solution (Ridge function, Pop. size = 50).

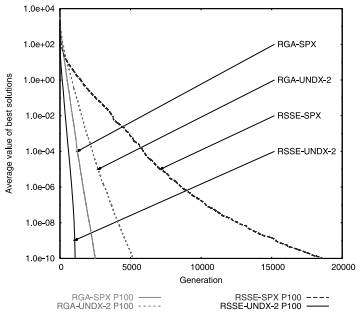

図 13 Ridge 関数における到達解の平均値（個体数 100）
Fig. 13 Mean fitness of best solution (Ridge function, Pop. size = 100).

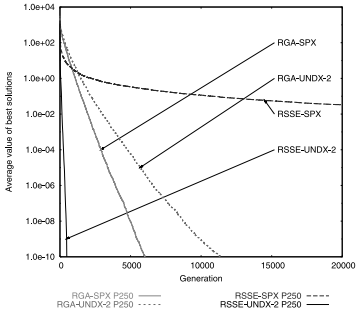

図 14 Ridge 関数における到達解の平均値（個体数 250）
Fig. 14 Mean fitness of best solution (Ridge function, Pop. size = 250).

かる．

5.5 Rosenbrock 関数の実験結果

解を探索することができた回数を**表 8** に示す．各個体数における，最良個体の解の平均値のグラフを**図 15**，**図 16**，**図 17** に示す．

表 8 より，RSSE-SPX は RGA-SPX よりも探索性能が良くないことが分かる．そして，RGA-UNIDX-2 と RSSE-UNDX-2 を比較すると，閾値が大きいときには同程度の探索性能を示すが，閾値を小さくすると RSSE-UNDX-2 は RGA-UNIDX-2 より多くの個体を必要とすることが分かる．

図 15〜17 より，RGA と比べ RSSE は，収束速度が遅く，このことからより良い閾値に到達するには時間がかかると思われる．

5.6 Griewank 関数の実験結果

解を探索することができた回数を**表 9** に示す．また，各個体数における，最良個体の解の平均値のグラフを**図 18**，**図 19**，**図 20** に示す．

表 9 より，いずれの場合においても，RSSE は RGA よりも解を探索できた回数が少なくなっていることが分かる．図 18〜20 より，RGA，RSSE は，急速に収束したあと，ほとんど解が向上していないことが分かる．悪い到達解を示す試行が若干でもある場合，平均性能は大幅に悪化してしまうので，RGA においても，RSSE と同様の推移をした解の収束となる．これより，Griewank 関数では，RGA，RSSE どちらにおいても，初期母集団に設定に依存して局所解に陥ってしまうことが多いことが分かる．

5.7 最適解の座標をオフセットした Rastrigin 関数の実験結果

SPX と UNDX-m は交叉により生成される子個体が親個体集団の共分散行列をよく保存するので，親個体が分布している領域の中心部を重点的にサンプリングする性質がある[9),10),17)]．そこで，Rastrigin 関数において最適解の位置を原点からずらした（オフセットした）Rastrigin 関数の解探索問題を考える．30 回の試行において解を探索できた世代数の平均値を**表 10** に示す．表において O0.0/O2.0/O3.0 は各設計変数において最適値を原点からずらした値を示す．したがって，O0.0 はオフセットしない場合を示す．

RGA では，オフセットしない場合（O0.0）に比べて多くの世代数が必要であることが分かる．O1.0 よ

表 8 解を探索できた回数（Rosenbrock 関数の場合）
Table 8 Number of searchable trials (Rosenbrock function).

	RGA-SPX	RGA-UNDX-2	RSSE-SPX	RSSE-UNDX-2
閾値	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250	P50 / P100 / P250
1.0e-2	27 / 30 / 30	30 / 30 / 30	0 / 13 / 30	30 / 30 / 30
1.0e-5	1 / 29 / 30	30 / 30 / 30	0 / 0 / 0	30 / 30 / 30
1.0e-8	0 / 3 / 30	30 / 30 / 30	0 / 0 / 0	0 / 30 / 30

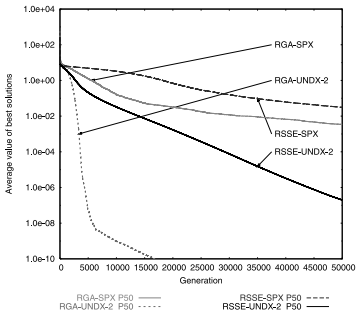

図 15 Rosenbrock 関数における到達解の平均値（個体数 50）
Fig. 15 Mean fitness of best solution (Rosenbrock function, Pop. size = 50).

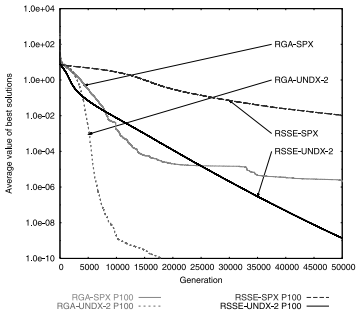

図 16 Rosenbrock 関数における到達解の平均値（個体数 100）
Fig. 16 Mean fitness of best solution (Rosenbrock function, Pop. size = 100).

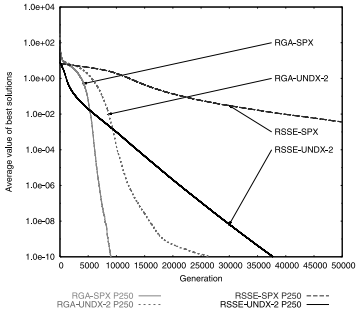

図 17 Rosenbrock 関数における到達解の平均値（個体数 250）
Fig. 17 Mean fitness of best solution (Rosenbrock function, Pop. size = 250).

りも O2.0 において多くの世代数を必要とすることから，最適解の位置が原点から遠ざかるほど解探索が困難になることが分かる．これに対して，RSSE は，オフセットの程度によらず解を探索できていることが分かる．

RGA は，親個体をランダムに選択するので，その結果解空間全体から均一に親個体を選択することになる．SPX と UNDX-m が親個体が分布している中心部に重点的にサンプリングすることから，RGA では結果的に解空間の中心付近を重点的に探索し，中心から離れたところの解探索が困難になる場合がある．これに対して，RSSE は評価値の高い個体を親個体に選択するので，最適解が中心から離れていても，探索が解空間の中心から（それより離れた）最適解付近に移動できるので，最適解の位置を中央からずらした Rastrigin 関数においても，通常の Rastrigin 関数と同様の探索性能を示すことができたと考えられる．

5.8 探索性能のまとめ

設計変数間に依存関係がない問題では，単峰性関数，多峰性関数どちらにおいても，RSSE-SPX，RSSE-UNDX-2 ともに，全試行において，RGA と同様に解を探索できていることが分かった．また，RSSE-SPX，RSSE-UNDX-2 ともに，RGA と比べて速い収束速度を示している．特に，RSSE-UNDX-2 は，他の探索手法と比べて優れた収束速度を示すことが分かった．

RSSE では，良い親個体を多く集め，そこから似通った構造を持つ子個体を生成する．これにより，RGA と比べて優れた収束速度を示すことができた．また，RSSE-SPX に比べ RSSE-UNDX-2 は，速い収束速度を示している．SPX は，親個体の実数値ベクトルからランダムに子個体を生成するのに対して，UNDX-m は，正規分布により各親個体の重心の近くに子個体を生成する．これにより RSSE-SPX に比べ RSSE-UNDX-2 は親個体に近い構造を持つ子個体を生成しており，母集団の個体が似通った構造に早く集中して，より速い収束速度を示した．

次に，設計変数間に依存関係がある問題について述べる．

単峰性関数では，RSSE-SPX は，他の探索手法と比べて収束が遅く，全試行において解を探索できなかった．それに対して，RSSE-UNDX-2 は，全試行において，RGA と同様の解探索性能を示した．単峰性関数で

表 9 解を探索できた回数（Griewank 関数の場合）
Table 9 Number of searchable trials (Griewank function).

閾値	RGA-SPX P50 / P100 / P250	RGA-UNDX-2 P50 / P100 / P250	RSSE-SPX P50 / P100 / P250	RSSE-UNDX-2 P50 / P100 / P250
1.0e-2	23 / 30 / 30	14 / 23 / 28	1 / 8 / 1	1 / 3 / 3
1.0e-5	17 / 19 / 28	6 / 17 / 26	1 / 5 / 0	0 / 0 / 3
1.0e-8	0 / 0 / 4	0 / 17 / 25	1 / 5 / 0	0 / 0 / 3

表 10 解を探索するまでに要した平均世代数（オフセットされた Rastrigin 関数の場合）
Table 10 Mean generation to search solution (Off-set Rastrign function).

閾値	RGA-SPX O0.0 / O1.0 / O2.0	RGA-UNDX-2 O0.0 / O1.0 / O2.0	RSSE-SPX O0.0 / O1.0 / O2.0	RSSE-UNDX-2 O0.0 / O1.0 / O2.0
1.0e-2	3142 / 8859 / 10647	11624 / 13980 / 14405	621 / 575 / 604	334 / 222 / 219
1.0e-5	4542 / 8902 / 10789	13646 / 15669 / 16701	1540 / 1484 / 1517	342 / 233 / 228
1.0e-8	6008 / 9198 / 10906	16045 / 18533 / 19056	2909 / 2882 / 2861	357 / 263 / 265

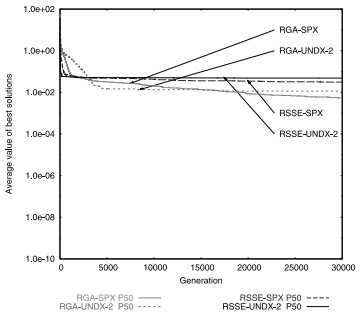

図 18 Griewank 関数における到達解の平均値（個体数 50）
Fig. 18 Mean fitness of best solution (Griewank function, Pop. size = 50).

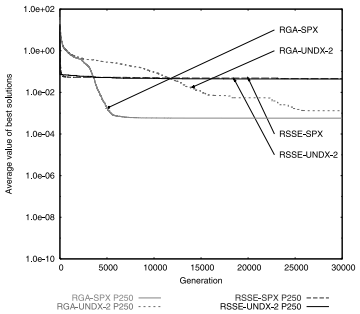

図 20 Griewank 関数における到達解の平均値（個体数 250）
Fig. 20 Mean fitness of best solution (Griewank function, Pop. size = 250).

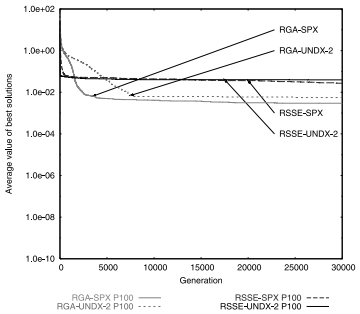

図 19 Griewank 関数における到達解の平均値（個体数 100）
Fig. 19 Mean fitness of best solution (Griewank function, Pop. size = 100).

ある Ridge 関数において，RSSE-UNDX-2 は，RGA と比べて優れた収束速度を示した．しかし，同じ単峰性関数である Rosenbrock 関数において，RSSE-UNDX-2 は，RGA と比べて収束速度が遅くなった．Ridge 関数は単峰性かつ凸関数なのに対して，Rosenbrock 関数は単峰性の関数であるが，準最適解が湾曲した形状に分布している非凸関数である．RSSE-UNDX-2 は，その特性から最適解近傍の局所解に集中する場合があり，良い閾値に到達するには RGA よりも多くの世代数を必要とする結果となった．

次に，多峰性関数における結果について述べる．Griwank 関数では，RSSE-SPX，RSSE-UNDX-2 ともに，RGA に比べて探索性能が劣っていた．多峰性関数は局所解が多数存在するので，大域的探索性能を改善するために多様性を維持することが重要である．適合度の良い親個体に似通った解構造を持つ子個体を生成する RSSE では，ランダムな親個体の選択をする RGA よりも多様性を維持しにくい．よって，多峰性関数では，RGA と比べ RSSE の探索性能は劣った結果となったと考えられる．これより，設計変数間に依存関係がある多峰性関数では，RGA に比べ多様性の維持が困難である RSSE は，より大きな母集団が必要であると考えられる．

ところで，Griwank 関数では RSSE は RGA と比べてより小さい閾値で解を探索できる試行回数が少なかったが，その少ない試行回数の中に RGA よりも早く収束できた試行があった．そこで，Griewank 関数を例に，RGA と RSSE が最も早く解を探索できたと

表 11 最も早く解を探索できた場合の世代数（Griewank 関数の場合）
Table 11 Min. generation to search solution (Griewank function).

閾値	RGA-SPX P50 / P100 / P250	RGA-UNDX-2 P50 / P100 / P250	RSSE-SPX P50 / P100 / P250	RSSE-UNDX-2 P50 / P100 / P250
1.0e-2	6922 / 2251 / 3818	1936 / 3151 / 10301	10340 / 404 / 331	279 / 44 / 17
1.0e-5	7583 / 6849 / 14665	2842 / 4216 / 12534	11251 / 1002 / -	- / - /26
1.0e-8	- / - / -	- / 5280 / 15060	12533 / 2086 / -	- / - / 35

きの世代数をまとめて**表 11** に示す．これより，RSSE は RGA よりもかなり早く解を探索できていることが分かる．特に，個体数 250 での RSSE-UNDX-2 は，RGA と比べて非常に早く解を探索できている．このことは，初期集団などのパラメータを適切に設定できれば，Griwank 関数などの多峰性関数でも RSSE は RGA よりもかなり優れた収束特性を示すといえる．

最後に，Rastrigin 関数において最適解を解空間の中央からオフセットした問題における探索性能を比較した．その結果，RSSE は，RGA と比べて探索性能を低下させずに解探索することができた．RSSE は，必ずしも最適解が中央にない問題においても，RGA よりも優れた探索性能を示すことが分かった．

6. ま と め

本論文では，多数の個体を用いる実数値交叉手法である SPX と UNDX-m を用いて，SSE を実数値問題へ拡張した RSSE を提案した．そして，実数値最適化問題を通して，RGA と RSSE の探索性能を比較した．その結果，以下のことが分かった．

RSSE-SPX は，設計変数間に依存関係のない単峰性関数と多峰性関数において，RGA と同様の最も良い閾値を探索していることが分かった．また，RSSE-UNDX-2 は，設計変数間に依存関係がない単峰性関数と多峰性関数，さらに，設計変数間に依存関係がある単峰性関数において，RGA と同様の最も良い閾値を優れた収束速度で探索していることが分かった．

設計変数間に依存関係がある多峰性関数では，RSSE-UNDX-2 は，多くの試行において良い閾値を示すことができなかったが，RGA と比べて優れた早さで最も良い閾値を探索している試行があることが分かった．RSSE-UNDX-2 は，設計変数間に依存関係がある多峰性関数においても，個体数を十分に大きくとることで，優れた収束速度で RGA と同等の最も良い閾値を示す解探索が安定して実現できると考えられ，今後の課題である．

参 考 文 献

1) 相澤彰子：スキーマ処理に基づく集団型探索アルゴリズム，情報処理学会研究報告，Vol.1994, No.093, pp.1–8 (1994).
2) 丸山 崇，北 栄輔：確率的スキーマ貪欲法の検討と拡張，性能比較について，情報処理学会論文誌：数理モデル化と応用，Vol.47, No.SIG14(TOM15), pp.16–30 (2006).
3) 丸山 崇，北 栄輔：SSE の世代交代モデルを改良した cSSE について，情報処理学会論文誌：数理モデル化と応用，Vol.48, No.SIG2(TOM16), pp.78–90 (2006).
4) 川面恵司，横山正明，長谷川浩志：最適化理論の基礎と応用—GA および MDO を中心にして，コロナ社 (2000).
5) 橋本崇史：遺伝的アルゴリズムを用いた風車翼断面の最適設計法に関する研究，東京工業大学大学院総合理工学研究科知能システム科学専攻修士論文 (2006).
6) 小野 功，小林重信，吉田幸司：遺伝的アルゴリズムによる光学系の設計，数理モデル化と問題解決シンポジウム論文集，pp.39–46 (2000).
7) Takahashi, O., Kita, H. and Kobayashi, S.: Protein folding by a hierarchical genetic algorithm, *The 4th International Symposium on Artificial Life and Robotics*, pp.334–339 (1999).
8) 喜多 一，小野 功，小林重信：実数値 GA のための正規分布交叉の多数の親を用いた拡張法の提案，計算自動制御学会論文集，Vol.36, No.10, pp.875–883 (2000).
9) Higuchi, T., Tsutsui, S. and Yamamura, M.: Theoretical analysis of simplex crossover for realcoded genetic algorithms, *Proc. 6th International Conference on Parallel Problem Solving from Nature*, Schoenauer, M., et al. (Eds.), pp.365–374 (2000).
10) Kita, H., Ono, I. and Kobayashi, S.: Multi-parental extention of the unimodal normal distribution crossover for real-coded genetic algorithm, *Proc. 1999 Congress on Evolutionary Computation*, Porto, V.W. (Ed.), pp.1581–1588 (1999).
11) 佐藤 浩，小野 功，小林重信：遺伝的アルゴリズムにおける世代交代モデルの提案と評価，人

工知能学会誌，Vol.12, No.5, pp.734–744 (1997).
12) 小野　功，山村雅幸，喜多一：実数値 GA とその応用，人工知能学会誌，Vol.15, No.2, pp.259–266 (2000).
13) Eshleman, L. and Schaer, J.D.: Real-coded genetic algorithms and interval-schemata, *Foundations of Genetic Algorithms 1993*, Vol.2, pp.187–202 (1993).
14) Ono, I. and Kobayashi, S.: A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover, *Proc. 7th International Conference on Genetic Algorithms*, Back, T., et al. (Eds.), pp.246–253 (1997).
15) De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems, Ph.D. thesis, University of Michigan, 1975, Dissertation Abstracts International, Vol.36, No.10, 5140B; UMI 76-9381.
16) Whitley, D., Mathias, K., Rana, S. and Dzubera, J.: Building better test functions, Eshelman, L. (Ed.), *Proc. 6th International Conference on Genetic Algorithms*, pp.239–246 (1995).
17) 染谷博司，山村雅幸：最適解の位置にロバストな実数値 GA を実現する Toroidal Search Space Conversion の提案，人工知能学会論文誌，Vol.16, No.3, pp.333–343 (2001).

(平成 18 年 12 月 15 日受付)
(平成 19 年 6 月 4 日再受付)
(平成 19 年 8 月 24 日採録)

丸山　崇（学生会員）

1978 年生．ソネットエンタテインメント（株）勤務（当時，名古屋大学大学院情報科学研究科博士課程後期課程在学）．遺伝的アルゴリズム等の進化的計算手法の性能向上と実問題への応用，Web マイニングへの応用について研究している．

北　栄輔（正会員）

1964 年生．1991 年名古屋大学大学院工学研究科博士課程後期課程修了．博士（工学）．1999 年より名古屋大学助教授，2007 年より准教授，現在に至る．数値解析法（BEM, Trefftz 法），セル・オートマトン（Cellular Automata）等の研究に従事．著書に，『偏微分方程式の数値解法』，『計算のための線形代数』，『Trefftz 法入門』等．IEEE, ISBE，応用数理学会，日本機械学会，シミュレーション学会，日本計算工学会各会員．