岡山理科大学紀要 第42号 A pp. 95-104(2006)

進化的計算手法による関数最適化性能の比較

林　孝志郎・谷口　隆裕・上野　伸郎・片山　謙吾*
南原　英生*・成久　洋之*

岡山理科大学大学院工学研究科情報工学専攻
*岡山理科大学工学部情報工学科

(2006年10月2日受付、2006年11月6日受理)

1　はじめに

進化的計算は，生物の進化過程や生物の群の挙動を模倣した最適化手法の総称であり，一般に複数の解候補を利用し探索する特徴を持つ．その代表例として，遺伝的アルゴリズム (Genetic Algorithm; GA) [1] [2] [4] [5] [6]，Particle Swarm Optimization (PSO) [7] [8]，進化的プログラミング (Evolutionary Programming; EP) [9] [10] [11] などのアルゴリズムが知られており，関数最適化問題などの現実的な最適化問題に対して適用されてきた．しかしながら，これら手法間の枠組みを越えて，基本的な探索性能の比較を行った研究はほとんど存在しない．これらの枠組みを越えた探索性能を検討することで，現実問題に対して，アルゴリズムを改良する手がかりや探索特性等に関する見知等が得られ易いものと考えられる．そこで本論文では，関数最適化問題を対象として，代表的な交叉法（UNDX [5], BLX-α [4] 等）を有する5つのGAと，代表的なEPとして知られているCEP [9]，FEP [9]，EEP [10] の3つのEPおよびPSO [7] の基本的な探索性能を比較検討し，アルゴリズム改良の手がかりや特性に関する見知等の獲得を試みる．

2　関数最適化問題

最適化問題は一般的に次のように表される．

$$\min_{x} \quad f(x) \tag{2.1}$$

$$\text{subject to} \quad x \in F \tag{2.2}$$

$$F \subseteq X \tag{2.3}$$

$f(x)$ は目的関数 (objective function) と呼ばれ，解 x の良さを表す尺度で実数値をとるものとし，集合 F は可能領域 (feasible region) を表す．x は通常ベクトル量で，問題の性質から定まる基本空間 X の要素となるが，実際にはその部分集合である F の要素しかとることを許さない．この関係を規定した (2.2) 式を制約条件 (constraint) と呼ぶ．

3　進化的計算

進化的計算とは，実行可能解 $x \in F$ を複数個有する個体集団（population）を進化させて，適応度の改善された新しい個体集団を生成する操作を繰り返すことで，集団中の個体の適応度を良質なものしていく手法である．以下では，本論文で検討する代表的な進化的計算手法のGA，EP，PSOの詳細に関してそれぞれ述べる．

3.1 遺伝的アルゴリズム (Genetic Algorithm : GA)

遺伝的アルゴリズム (GA) の研究は，1960年代から米国のJ. Hollandとその弟子らがミシガン大学で行った研究にさかのぼる [12]．このアルゴリズムは，生物の遺伝的機構を比較的忠実にモデル化したもので集団的最適化手法の一種である．一般にGAは，選択，交叉，突然変異のオペレータから構成される．選択 (serection) オペレータは，解 (個体) 集団における適応度順に良いものから次世代に生存する個体集団を決定する．よって，選択は集団の中から適切となりうる個体を取捨選択するだけで，新しい個体を生成するもの

ではない．交叉 (crossover) オペレータは，両親の染色体を部分的に組み替えることにより新しい子の個体を生成するもので，GA による探索の主要オペレータである．探索空間を探索することで，得られた個体群から個体のペアを選び交叉させることによって生成された子孫が，親の有効な特質を受け継ぎ，探索空間の新しい領域に移動することができるというメカニズムを用いている．本論文では，遺伝子を実数で表した実数値 GA を対象とする．次にその実数値 GA の処理の流れを以下に示す．

Step1 (初期個体集団の生成)：μ 個の n 次元実数ベクトル対 $(x_{1i}, x_{2i}), i = 1, 2, \ldots, \mu$ を生成する．

Step2 (各個体の適応度評価)：$f(x_i), i = 1, 2, \ldots, \mu$ を計算する．

Step3 (交叉)：GA における交叉法を用い子孫を生成する．

Step4 (子孫の適応度評価)：子孫 $(x_i^{(t+1)})$ における $f(x_i^{(t+1)}), i = 1, 2, \ldots \mu$ を計算する．

Step5 (各個体の並べ替え)：適応度の優れている物から順に並べ替える．

Step6 (次世代の親の選択)：Step5 で適応度が優れている物より μ 個選択し次世代の親とする．

Step7 (停止・続行判定)：終了条件を満たせば処理停止，そうでなければ Step3 へ戻る．

実数値 GA における代表的な交叉法としては，線形交叉 (Linear Crossover)，混合交叉 (BLX-α)，擬似二進交叉 (SBX)，単峰性正規分布交叉 (UNDX) などがこれまでに提案されている．それらの交叉法の概要を以下にそれぞれ記述する．なお，t 世代における 2 個の親を $x^{(1,\,t)}$，$x^{(2,\,t)} \in R^n$，$x = \{x_1\}$ と仮定する．

3.1.1 線形交叉 (Linear Crossover) [3]

線形交叉法において，4 次元目を交叉箇所と仮定した場合の交叉の例について次に示す．

$$
\begin{array}{lcccccc}
x^{(1,\,t)}: & (x_1^{(1,t)} & x_2^{(1,t)} & x_3^{(1,t)} & x_4^{(1,t)} & \ldots & x_n^{(1,t)}) \\
x^{(2,\,t)}: & (x_1^{(2,t)} & x_2^{(2,t)} & x_3^{(2,t)} & x_4^{(2,t)} & \ldots & x_n^{(2,t)}) \\
x^{(1,\,t+1)}: & (x_1^{(1,t)} & x_2^{(1,t)} & x_3^{(1,t)} & x_4^{(2,t)} & \ldots & x_n^{(2,t)}) \\
x^{(2,\,t+1)}: & (x_1^{(2,t)} & x_2^{(2,t)} & x_3^{(2,t)} & x_4^{(1,t)} & \ldots & x_n^{(1,t)})
\end{array}
$$

3.1.2 混合交叉 (BLX-α) [4]

混合交叉 (BLX-α) は，1993 年に Eshelman と Schaffer らにより提案されたものである．この手法は，次の交叉式により子孫を生成する．

$$x_i^{(1,t+1)} = (1 - \gamma')\, x_i^{(1,\,t)} + \gamma_i' x_i^{(2,\,t)} \tag{3.1}$$

$$\gamma_i' = (1 + 2\alpha)\, u_i - \alpha \tag{3.2}$$

$$u_i\text{: [0, 1] 区間の一様乱数} \tag{3.3}$$

3.1.3 擬似二進交叉 (SBX) [2]

擬似二進交叉 (Simulated Binary Crossover: SBX) は Bitstring GA における一点交叉の原理を模倣する交叉法である．具体的には 2 個体の親から 2 つの個体を生成する際に，一つの設計変数に対してのみ SBX を適応する．以下に SBX の交叉式を示す．

$$x_i^{(1,t+1)} = 0.5((1 + \beta_{qi})x_i^{(1,t)} + (1 - \beta_{qi})x_i^{(2,t)}) \tag{3.4}$$

$$x_i^{(2,t+1)} = 0.5((1 - \beta_{qi})x_i^{(1,t)} + (1 + \beta_{qi})x_i^{(2,t)}) \tag{3.5}$$

$$\beta_{qi} = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}} & if \quad u_i \le 0.5, \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}}, & otherwise. \end{cases} \tag{3.6}$$

$$u_i\text{: [0, 1] 区間の一様乱数} \tag{3.7}$$

3.1.4 改良型擬似二進交叉 (vSBX) [13]

改良型擬似二進交叉 (vSBX) は，従来の擬似二進交叉 (SBX) を変形した方法である．以下に vSBX の交叉式を示す．

$$x_i^{(1,t+1)} = \begin{cases} 0.5((1+\beta_{1i})x_i^{(1,t)} + (1-\beta_{1i})x_i^{(2,t)}) & 0 < u \le 0.5 \\ 0.5((1-\beta_{1i})x_i^{(1,t)} + (1+\beta_{1i})x_i^{(2,t)}) & otherwise. \end{cases} \tag{3.8}$$

$$\beta_{1i} = (1/2u_i)^{\frac{1}{\eta+1}} \tag{3.9}$$

$$x_i^{(2,t+1)} = \begin{cases} 0.5((3-\beta_{2i})x_i^{(1,t)} - (1-\beta_{2i})x_i^{(2,t)}) & 0 < u \le 0.5 \\ 0.5(-(1-\beta_{2i})x_i^{(1,t)} + (3-\beta_{2i})x_i^{(2,t)}) & otherwise. \end{cases} \tag{3.10}$$

$$\beta_{2i} = (1/2(1-u_i))^{\frac{1}{\eta+1}} \tag{3.11}$$

$$u_i: [0, 1] \text{ 区間の一様乱数} \tag{3.12}$$

3.1.5 単峰性正規分布交叉 (UNDX) [1]

単峰性正規分布交叉 (UNDX) は，小野らによって提案された交叉法で，多峰性や設計変数間に依存関係のあるような目的関数の最適化において良好な探索性能を示した．以下に UNDX の交叉式を示す．

$$x_i^{(t+1)} = m + \xi d \tag{3.13}$$

$$m = (x_i^{(1,t)} + x_i^{(2,t)}/2) \tag{3.14}$$

$$d = (x_i^{(1,t)} - x_i^{(2,t)}) \tag{3.15}$$

$$\xi \sim N(0,\sigma^2) \tag{3.16}$$

ただし，$N(0,\sigma^2)$ は平均 0，分散 σ^2 の正規分布を表す．

3.2 進化的プログラミング (EP)[9] [10] [11]

進化的プログラミング (EP) [9] [10] [11] は，遺伝的アルゴリズム (GA) と同様に，生物の進化過程と自然淘汰のアイデアを模倣した最適化手法であり，複雑かつ大規模な問題に対する解法として高性能な探索が可能であると考えられている．GA も EP も個体集団 (population) を形成し，集団中で精度の良い近似解を求めようとする手法であるが，GA の主要オペレータが交叉 (crossover) であるのに対し，EP では突然変異 (mutation) である．そもそも EP は，L.J. Fogel が人工知能に対するアプローチとして提案したもので，後に D.B. Fogel によって最適化手法として発展してきた．この EP は Gaussian Mutation (Gauss 分布に従う乱数を使用した突然変異によって解を進化させる) を主体としており，これは CEP と一般的に呼ばれている．1999 年に，Yao らは Cauchy Mutation (Cauchy 分布に従う乱数を使用した突然変異により解を進化させる) を主体とした EP を提案し，FEP と呼んでいる [9]．2002 年に，Narihisa らは Exponential Mutation (復号指数分布に従う乱数を使用した突然変異により解を進化させる) を利用した EP[16] [17] [18] を提案し，更に 2005 年には，従来の EP 手法で重視されている戦略パラメータを使用しない nsEEP[10] を提案している．

以下に一般的な EP の手順を示す．

Step1 (初期個体集団の生成)：μ 個の n 次元実数ベクトル対 $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i), i = 1, 2, \ldots, \mu$ を生成する．

Step2 (各個体の適応度評価)：$f(\boldsymbol{x}_i), i = 1, 2, \ldots, \mu$ を計算する．

Step3 (子孫の生成)：各個体 $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$ から単一の子孫 $(\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)$ を次のように生成する．

$$\sigma'_i(j) = \sigma_i(j) \exp[\tau' N(0,1) + \tau N_j(0,1)], \tag{3.17}$$

$$x'_i(j) = x_i(j) + \sigma'_i(j) N_j(0,1), \tag{3.18}$$

$$i = 1, 2, \ldots, \mu, \quad j = 1, 2, \ldots, n$$

ただし，$x_i(j), x'_i(j), \sigma_i(j), \sigma'_i(j)$ はそれぞれ，ベクトル $\boldsymbol{x}_i, \boldsymbol{x'}_i, \boldsymbol{\sigma}_i, \boldsymbol{\sigma'}_i$ の j 番目の成分を示す．$N(0,1)$ は正規乱数 (平均=0, 標準偏差=1) を表し，$N_j(0,1)$ は各 j 毎に新たに乱数を発生させるものである．

Step4 (子孫の適応度評価)：子孫 $(\boldsymbol{x'}_i, \boldsymbol{\sigma'}_i)$ における $f(\boldsymbol{x'}_i), i = 1,2,\ldots,\mu$ を計算する．

Step5 (各個体のトーナメント評価)：2μ 個の親と子孫 $\{(\boldsymbol{x}_i, \boldsymbol{\sigma}_i), (\boldsymbol{x}'_i, \boldsymbol{\sigma}'_i)\}, i = 1,2,\ldots,\mu$ の中から任意の q 個をランダムに選ぶ．これらの q 個の x ベクトルを $s_1, s_2, \ldots, s_q$ とする．$f(x_i) \geq f(s_k)$ であれば個体 (x_i, σ_i) は勝ち点 1 点を取得する．すべての $i(i = 1,2,\ldots,2\mu)$ について，$k = 1,2,\ldots,q$ の比較を行う．

Step6 (次世代の親の選択)：Step5 での勝ち点の多い方から μ 個選択し次世代の親とする．

Step7 (停止・続行判定)：終了条件を満たせば処理停止，そうでなければ Step2 へ戻る．

なお，上記した手順は CEP であるが，(3.16) 式の $N_j(0,1)$ をコーシー乱数を発生させる $C_j(0,1)$ に変更することにより FEP となる．また，複合指数乱数を発生させる $E_j(0,\lambda)$ に変更することにより EEP となる．

3.3 Particle Swarm Optimization(PSO) [7] [8]

Particle Swarm Optimization (PSO) は 1995 年に J. Kennedy と R. Eberhart によって提案された進化的計算手法の一つである．PSO の概要は以下のとおりである．

Step1 (初期個体集団の生成)：

- m 個の個体を n 次元空間に配置し，各個体は適応度の計算をして各個体にそれぞれ自分のメモリに必要な情報 ($\mathbf{X_i}$, $\mathbf{V_i}$, $\mathbf{P_i}$, $\mathbf{P_j}$, $\mathbf{P_n}$) を記憶させる．なお，$\mathbf{X_i}$, $\mathbf{V_i}$, $\mathbf{P_i}$, $\mathbf{P_j}$, $\mathbf{P_n}$ は，自分の位置，速度，自分自身がこれまでに発見した最良点，集団全体がこれまでに発見した最良点，$\mathbf{X_i}$のより近くにいてより優れている個体 をそれぞれ表す．
- 個体全体をランダムに N 個のグループに分ける．

Step2 (各個体の速度更新)：

- 重み ($\psi_1 \sim \psi_3$) の値を決定する．
- 速度更新にはベクトルの考えを採用して以下の式により更新を行う．

$$V_{id}^{t+1} = \omega \times V_{id}^{(t)} + \psi_1 \times (p_{id} - X_{id}^t) + \psi_3 \times (p_{nd} - X_{id}^t) \tag{3.19}$$

- 速度更新の後，以下の式により ω の値を更新させる．

$$\omega = \omega - \frac{W_0 - W_T}{Generation} \tag{3.20}$$

なお，W_0 は開始時の ω，W_T は終了時の ω，$Generation$ は世代数を表している．

Step3 (各個体の位置更新)：

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)} \tag{3.21}$$

Step4 (適応度の計算)：$f(X_i), i = 1,2,\ldots,m$ を計算する．

Step5 (メモリの更新)：各個体のメモリ情報を更新する．

Step6 (停止・続行判定)：終了条件を満たせば処理停止，そうでなければ Step2 へ戻る．

4 数値実験と結果

代表的な進化的計算手法である遺伝的アルゴリズム (GA)，Particle Swarm Optimization (PSO)，進化的プログラム (EP) の基本的な探索性能を比較する研究はこれまで殆ど行われていない．このことから本論文では，関数最適化問題を対象とした数値実験を通して，上述した実数値 GA，EP，PSO の探索性能について比較検討する．数値実験で使用する関数最適化問題の例題は，文献 [9] から与えられた，Sphere Model 関数を含む代表的な問題から 23 例題 ($f_1 \sim f_{23}$)，文献 [14] より与えられた，Rosenbrock 関数を含む変数間依存のある問題例から 2 例題 (f_{24}，f_{25})，文献 [15] より与えられた Fletcher and Powell 関数の 1 例題 (f_{26}) の計 26 例題を対象とする．その詳細を表 1 に示す．なお，表 1 では，対象とする関数の次元数を n とし，x の範囲を S，最適値を f_{min} としている．

表 1: 関数最適化問題の 26 問題例

Test Function	n	S	f_{min}
$f_1 = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]^n$	0
$f_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^n$	0
$f_4 = \max \lvert x_i \rvert,\ 1 \le i \le n$	30	$[-100, 100]^n$	0
$f_5 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^n$	0
$f_6 = \sum_{i=1}^{n} (\lfloor x_i + 500 \rfloor)^2$	30	$[-100, 100]^n$	0
$f_7 = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_8 = \sum_{i=1}^{n}$	30	$[-500, 500]^n$	-12569.5
$f_9 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]^n$	0
$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	30	$[-32, 32]^n$	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$	0
$f_{12} = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	30	$[-50, 50]^n$	0
$f_{13} = 0{,}1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30 30	$[-50, 50]^n$ $[-50, 50]^n$	0 0
$f_{14} = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^n$	1
$f_{15} = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]^n$	0.0003075
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18} = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	$[-2, 2]^n$	3
$f_{19} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2\right]$	4	$[0, 1]^n$	-3.86
$f_{20} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$	6	$[0, 1]^n$	-3.32
$f_{21} = -\sum_{i=1}^{5}\left[(x - a_i)(x - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]^n$	-10
$f_{22} = -\sum_{i=1}^{7}\left[(x - a_i)(x - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]^n$	-10
$f_{23} = -\sum_{i=1}^{10}\left[(x - a_i)(x - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]^n$	-10
$f_{24} = \sum_{i=1}^{n}\left\{100(x_1 - x_i^2)^2 + (x_i - 1)^2\right\}$	30	$[-2.048, 2.048]^n$	0
$f_{25} = \sum_{i=2}^{n}\left\{100(x_1 - (ix_i)^2)^2 + (ix_i - 1)^2\right\}$	30	$[-2.048/i, 2.048/i]^n$	0
$f_{26} = \sum_{i=1}^{n}(A_i - B_i)^2$ $A_i = \sum_{j=1}^{n}(a_{ij}\sin a_j + b_{ij}\cos\alpha_j)$ $B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$ ※α, a_{ij}, b_{ij} の値は文献 [15] を参照.	30	$[-\pi, \pi]^n$	0

公平な比較実験を行うために，実数値GA，EP，PSOのいずれの手法も，個体数を100，評価回数を1.0×10^6回(世代数を10000)とする．以下では，各手法において設定の必要な個別パラメータ値について記述する．実数値GAでは，交叉率を1.0，突然変異率を0とし，BLX-αで用いる$\alpha = 0.5$，vSBXで用いる$\eta = 0.1$，UNDXで用いる$\sigma = 0.5$とする．PSOでは，$\omega = 0.9 \rightsquigarrow 0.8$，$\psi 1 = \psi 2 = \psi 3 = 1.0$とする．nsEEPでは$\lambda_1 = 0.5$とし，FEPおよびCEPでは$\sigma$の下限を$\varepsilon = 10^{-4}$とする．ただし，以上のパラメータ設定値は，各手法において推奨されている値と同じ，もしくはほぼ同程度の値で設定している．

使用計算機として，FedoraCore 5のOSと3.50GByteのメモリとIntel社製のPentium®4 3.40GHzのCPUを有したHP xw4300 Workstationを使用した．使用言語はC言語で，コンパイラはgcc4.1.1（最適化オプション-O3適用）を用いた．

上述した問題例に対して，GA，EP，PSOの各手法を適用した結果を表2に示し，評価回数1.0×10^6回を100回試行した際の全処理の時間(秒)と最良解が発見された世代数を表3に示す．なお，表2は100回試行の平均値を与えており，最適解が負の数をとる場合を除いては，探索中に1.0×10^{-15}以下の解の評価値を得た場合は最適解を発見したと見なし0で示す．

表2: 個体数100のときの評価回数1.0×10^6回においての解比較

	UNDX(100)	Linear(100)	BLX-α(100)	SBX(100)	vSBX(100)	PSO(100)	CEP(100)	FEP(100)	nsEEP(100)
f_1	6.802014e-02	9.995916e+02	0	0	0	0	5.000253E-05	4.007158E-02	0
f_2	1.690503e-03	1.270178e+01	0	0	0	7.108517e-01	2.719174E-02	5.734784E-01	8.045215E-10
f_3	6.072261e+02	1.501840e+03	1.812311e-15	7.125987e-15	5.000000e+02	8.322589e-01	5.363910E-02	4.600210E+00	0
f_4	9.813295e+00	1.003833e+01	5.592297e+00	2.637108e+01	1.322172e+01	3.138958e-09	1.409596E+00	1.242656E-02	7.846427E-11
f_5	1.567227e+02	6.547078e+04	2.808332e+01	1.679025e+01	1.598401e+01	2.253901e+01	7.749705E+01	8.285637E+01	3.567763E+01
f_6	2.730000e+00	8.243082e+02	0	0	0	1.900000e-01	2.878990E+01	1.792000E-01	0
f_7	2.048840e-04	2.845148e-02	1.516167e-09	4.568701e-05	7.650947e-08	8.571323e-04	2.979808E-02	7.352695E-02	8.545261E-03
f_8	-8.064585e+03	-5.434609e+03	-9.509341e+03	-9.206381e+03	-9.999087e+03	-1.122181e+04	-7.266755E+03	-1.250544E+04	-1.103154E+04
f_9	1.168198e-03	6.491965e+00	0	0	0	3.537403e+01	1.053131E+01	1.693890E+01	3.096310E+01
f_{10}	2.892296e-01	6.946084e+00	1.753930e-15	1.158803e-01	2.176037e-15	7.760500e-01	1.379441E+00	1.895310E-01	1.355523E-10
f_{11}	5.314903e-04	8.990153e+00	0	0	0	1.453872e-02	9.640528E-02	1.599916E-02	5.025374E-03
f_{12}	3.805590e+01	8.359452e+03	1.134736e-02	5.385439e-0 1	2.603099e-02	7.859936e-02	1.247813E+00	2.916831E-04	0
f_{13}	8.368402e-01	5.246443e+02	9.888629e-04	2.317406e-01	8.789893e-04	9.415131e-03	1.505819E-02	7.993388E-03	0
f_{14}	1.968140e+00	1.781865e+00	1.551962e+00	1.443977e+00	1.314598e+00	9.980038e-01	4.495576E+00	1.799507E+00	7.607932E-01
f_{15}	8.983542e-04	9.725766e-04	7.264811e-04	3.355968e-04	4.783955e-04	8.184855e-04	5.115696E-02	5.107184E-02	5.548782E-04
f_{16}	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.031623e+00	-1.0316284E+00	-1.0316284E+00	-1.0316284E+00
f_{17}	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874E-01	3.978874E-01	3.978874E-01
f_{18}	3	3	3	3	3	3	3	3	3
f_{19}	-3.051823e+00	-3.051280e+00	-3.051881e+00	-3.051881e+00	-3.051881e+00	-3.862782e+00	-3.862782E+00	-3.862782E+00	-3.860000E+00
f_{20}	-3.312484e+00	-3.261212e+00	-3.297030e+00	-3.253037e+00	-3.279194e+00	-3.268488e+00	-3.000882E+00	-3.000882E+00	-3.228070E+00
f_{21}	-2.560803e+00	-2.560803e+00	-2.558381e+00	-2.560803e+00	-2.560803e+00	-2.619721e+00	-7.451259E+00	-7.451259E+00	-7.961539E+00
f_{22}	-8.057830e+00	-9.170422e+00	-8.608608e+00	-8.955255e+00	-7.980968e+00	-1.318773e+00	-8.582115E+00	-8.223680E+00	-9.353320E+00
f_{23}	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.318787e+00	-9.171497E+00	-9.508903E+00	-9.654228E+00
f_{24}	2.134516e+01	4.223187e+01	7.127177e+00	1.282301e+01	1.397777e+01	1.025928e+01	1.115991E+01	1.559334E+01	2.960314E+00
f_{25}	1.791593e+01	2.730554e+01	1.595391e+01	1.595937e+01	1.595391e+01	9.527164e+00	1.651909E+01	1.579638E+01	1.219243E+01
f_{26}	5.228198e+04	1.013432e+06	6.641450e+03	3.625170e+04	1.268350e+04	3.141968e+06	2.260778E+04	2.445086E+04	2.889286E+03

表3: 個体数100のときの全処理時間（秒）と最良解算出の世代数

	UNDX(100)		Linear(100)		BLX-α(100)		SBX(100)		vSBX(100)		PSO(100)		CEP(100)		FEP(100)		nsEEP(100)	
	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間
f_1	3506	488	594	108	439	401	1136	605	607	1060	1048	678	10000	3578	10000	2459	8288	517
f_2	3670	490	914	110	628	394	1584	606	820	1054	9707	760	10000	3601	10000	2482	10000	546
f_3	3692	569	315	230	10000	467	10000	679	3655	1127	7941	751	10000	3697	10000	2577	8552	626
f_4	300	491	1266	117	639	387	10000	604	6868	1051	10000	723	10000	3605	10000	2486	10000	544
f_5	5010	495	413	122	10000	384	10000	595	10000	1041	10000	678	10000	3585	10000	2470	8238	532
f_6	179	519	9980	155	118	410	380	625	167	1071	5366	978	9757	3609	9720	2492	2563	547
f_7	10000	853	9998	657	10000	742	10000	941	10000	1394	10000	993	10000	3912	10000	2805	10000	538
f_8	2156	716	7978	418	376	604	532	823	332	1270	10000	875	10000	3841	10000	2711	3485	773
f_9	3669	491	1732	111	618	394	1559	606	806	1054	10000	844	10000	3774	10000	2648	4965	708
f_{10}	2548	730	341	451	9925	578	1036	800	1062	1240	976	896	10000	3997	10000	2672	10000	738
f_{11}	3320	781	416	511	394	842	1020	874	544	1355	3629	976	9948	3895	10000	2748	5904	795
f_{12}	3242	738	722	543	745	624	971	842	495	1283	4598	956	9130	3823	10000	2721	7459	758
f_{13}	2897	719	395	521	347	619	787	836	481	1278	3732	946	10000	3806	10000	2693	6868	565
f_{14}	198	716	3663	1032	169	925	120	707	107	734	2354	741	164	3603	1770	2510	3563	550
f_{15}	3644	136	2654	127	9990	334	10000	167	10000	227	7559	153	9946	3587	9784	2489	8258	542
f_{16}	53	103	40	119	54	315	59	111	61	142	1346	120	113	3570	134	2459	2814	510
f_{17}	80	72	26	55	38	282	36	80	39	111	790	88	90	3576	113	2474	3110	517
f_{18}	30	68	25	48	34	277	33	77	37	107	8895	83	135	3572	192	2463	3376	511
f_{19}	131	172	178	156	40	356	56	185	45	244	9983	159	468	3634	127	2525	3636	563
f_{20}	86	196	321	164	233	364	78	225	312	313	9992	230	4077	3644	1392	2542	3955	564
f_{21}	166	108	43	67	410	295	50	125	45	184	9987	132	1045	3599	348	2472	4068	516
f_{22}	1094	110	316	73	590	297	285	246	174	185	9960	133	1599	3601	282	2475	4119	519
f_{23}	28	116	28	79	32	304	38	133	36	192	9942	132	514	3606	473	2479	4390	524
f_{24}	2364	496	7978	102	10000	382	10000	594	10000	1042	9561	684	10000	3596	10000	2486	8664	547
f_{25}	2953	505	284	115	277	393	610	614	302	1062	9997	759	10000	3674	10000	2600	9496	602
f_{26}	3933	22840	301	33803	10000	22839	10000	23175	10000	23626	1	24803	10000	25398	10000	24306	7829	22308

表2の結果よりBLX-α，SBX，vSBX，nsEEPは，$f_1 \sim f_{13}$の高次元の問題例の内，7つの単峰性関数($f_1 \sim f_7$)中3例題に対して最適解を発見しており，6つの多峰性関数($f_8 \sim f_{13}$)の内2例題に対しても最適解の探索に成功している．$f_{14} \sim f_{23}$の低次元の問題例および$f_{24} \sim f_{26}$の変数間に依存のある問題例については，nsEEPが他の手法に比べて最適解に近い解を算出できていることがわかる．

次いで，処理時間および最良解算出の世代数について考察する．表 3 の結果より各手法の中で最も速く処理を終えているのは，線形交叉 (Linear Crossover) にもとづく GA である．処理時間が他の手法に比べて速い理由は，交叉のアルゴリズムがシンプルであることに起因するものと考えられる．世代数に着目すると，比較的早い探索段階から最良解を算出している手法は，BLX-α 等の GA 関連の手法である．本研究で検討した GA 手法の枠組みには，突然変異オペレータを使用していないため，探索の早い段階で局所最適解に陥ったことが推測される．一方，EP 手法の多くは，最良解を発見する世代数が GA の場合に比べ遅れていることが観測できる．このことより，EP 手法は局所最適解に比較的陥りづらい探索が実現されていると考えられる．この理由として，GA では，交叉オペレータが主要操作であるのに対して，EP は突然変異オペレータを主要とした探索法であることが考えられる．

文献 [6] においては，UNDX に関して個体数を増やすことにより変数間に依存関係のある問題例に関しても探索の改善が見られることを指摘している．そこで以下では，前述した 9 手法に対して個体数を 400 に増加し，評価回数を個体数 100 のときと同じ 1.0×10^6 回 (世代数 : 2500) にしたときの結果について検討する．

表 4: 個体数 400 のときの評価回数 1.0×10^6 回においての解比較

	UNDX(400)	Linear(400)	BLX-α(400)	SBX(400)	vSBX(400)	PSO(400)	CEP(400)	FEP(400)	nsEEP(400)
f_1	0	1.574968e+02	0	0	0	0	8.368689E-05	9.321882E-02	0
f_2	0	4.764899e+00	0	0	0	7.890767e-01	3.421166E-02	8.212661E-01	1.218473E-09
f_3	1.952487e+01	2.042772e+02	4.388776e-04	1.028397e-03	1.435082e-03	1.581773e+00	2.883840E+01	2.696853E+01	1.124818E+02
f_4	9.796417e-01	3.958223e+00	0	1.037924e-08	1.149065e-14	4.833917e-06	2.316831E+01	1.564057E-01	1.289781E-10
f_5	3.174965e+01	1.996338e+03	2.569706e+01	3.229999e+01	3.225513e+01	5.445223e+01	8.152927E+01	9.578677E+01	3.889875E+01
f_6	0	6.088625e+01	0	0	0	2.000000e-01	3.778900E+01	8.982500E-02	0
f_7	4.447433e-06	8.303898e-04	7.077880e-05	2.088778e-03	6.813221e-04	2.461337e-03	5.334060E-02	1.356774E-01	2.017293E-02
f_8	-9.276118e+03	-6.284833e+03	-1.061493e+04	-1.012420e+04	-1.081021e+04	-1.088175e+04	-7.625218E+03	-1.253021E+04	-1.096109E+04
f_9	0	2.440394e+00	0	0	0	4.484083e+01	9.457108E+00	2.487908E+01	2.833641E+01
f_{10}	6.883383e-15	3.214860e+00	2.041534e-15	2.220446e-15	2.176037e-15	5.739414e-01	4.527558E-01	3.086271E-01	2.075043E-10
f_{11}	0	1.416853e+00	0	0	0	1.180670e-02	5.416554E+01	1.780968E-02	5.149221E-03
f_{12}	3.110071e-03	2.156137e+00	0	1.036690e-02	0	2.246831e-01	8.489296E-01	6.597140E-04	0
f_{13}	4.944315e-03	4.735723e+00	0	2.856715e-03	0	2.407701e-03	1.593688E-02	1.660098E-02	0
f_{14}	1.037765e+00	1.037735e+00	1.057645e+00	1.007944e+00	9.980038e-01	9.980038e-01	1.583139E+00	1.017884E+00	7.607939E-01
f_{15}	6.730234e-04	6.774710e-04	6.927115e-04	3.536701e-04	4.243764e-04	3.898985e-04	5.146163E-02	5.140191E-02	5.197535E-04
f_{16}	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.0316284e+00	-1.0316273e+00	-1.0316284E+00	-1.0316284E+00	-1.0316284E+00
f_{17}	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874e-01	3.978874E-01	3.978874E-01	3.978874E-01
f_{18}	3	3	3	3	3	3	3	3	3
f_{19}	-3.051881e+00	-3.051881e+00	-3.051881e+00	-3.051881e+00	-3.051881e+00	-3.862781e+00	-3.862782E+00	-3.862782E+00	-3.860000E+00
f_{20}	-3.321995e+00	-3.305570e+00	-3.316051e+00	-3.275628e+00	-3.301783e+00	-3.257724e+00	-3.000882E+00	-3.000882E+00	-3.248281E+00
f_{21}	-2.560803e+00	-2.560803e+00	-2.560803e+00	-2.560803e+00	-2.560803e+00	-2.623375e+00	-9.541439E+00	-9.541439E+00	-9.395336E+00
f_{22}	-1.007850e+01	-1.015320e+01	-1.007850e+01	-1.015320e+01	-1.002797e+01	-1.357348e+00	-1.013718E+01	-1.003087E+01	-1.019198E+01
f_{23}	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.021651e+00	-1.356917e+00	-1.042825E+01	-1.037417E+01	-1.037559E+01
f_{24}	8.732562e+00	2.806546e+01	3.410960e+00	4.884917e+00	3.339472e+00	1.268458e+01	1.098501E+01	1.694090E+01	1.075460E+01
f_{25}	1.595392e+01	2.563742e+01	1.595391e+01	1.595391e+01	1.595391e+01	1.100630e+01	1.806195E+01	1.726175E+01	1.575490E+01
f_{26}	1.075811e+04	4.075064e+05	2.942107e+03	1.652639e+04	3.718630e+03	2.811752e+06	2.132647E+04	2.649656E+04	2.728408E+03

表 5: 個体数 400 のときの全処理時間（秒）と最良解算出の世代数

	UNDX(400)		Linear(400)		BLX-α(400)		SBX(400)		vSBX(400)		PSO(400)		CEP(400)		FEP(400)		nsEEP(400)	
	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間	世代数	時間
f_1	505	542	1184	130	412	410	962	614	582	1067	732	1747	2500	3605	2500	2470	2109	526
f_2	540	544	654	128	612	412	1512	616	810	1069	2120	1833	2500	3643	2500	2492	2500	555
f_3	2500	605	781	250	2500	472	2500	679	2500	2150	2346	1829	2500	3802	2500	2592	1826	646
f_4	831	536	543	137	1722	417	2500	616	2500	1068	2500	1798	2500	3636	2500	2501	2500	551
f_5	2500	537	1364	141	2500	400	2500	606	2500	1058	2500	1762	2500	3605	2500	2480	2069	546
f_6	73	551	2500	168	110	423	247	633	153	1083	1610	1989	2497	3626	2500	2499	684	547
f_7	2500	895	2500	657	2500	749	2500	953	2500	1406	2500	2073	2500	3914	2500	2808	2500	538
f_8	504	760	2450	436	636	618	552	828	505	1280	2500	1950	2308	3860	2500	2728	893	779
f_9	531	553	513	128	602	412	1486	616	796	1069	2500	1914	2500	3785	2500	2651	1287	716
f_{10}	1142	741	850	477	2500	599	1661	820	992	1262	658	1970	2500	3812	2500	2676	2500	748
f_{11}	441	830	1147	533	371	684	859	902	523	1345	1051	2030	2500	3891	2500	2752	1538	803
f_{12}	262	777	1497	487	401	641	603	863	573	1306	1988	2031	2489	3836	2500	2703	1908	768
f_{13}	299	770	1863	465	412	638	665	855	587	1301	1373	2025	2500	3820	2500	2681	1768	567
f_{14}	101	721	755	1054	115	939	84	717	81	747	896	1441	40	3604	58	2492	771	561
f_{15}	2500	175	2500	133	2500	357	2500	184	2500	245	2489	904	2495	3595	2499	2485	2161	552
f_{16}	48	121	38	129	47	328	57	123	58	154	2500	823	44	3572	125	2461	762	509
f_{17}	30	77	26	67	31	293	34	92	36	123	588	790	50	3577	55	2476	844	517
f_{18}	27	73	24	59	30	290	31	88	33	118	535	786	66	3607	86	2465	912	510
f_{19}	35	187	37	163	38	310	44	197	42	256	2444	914	90	3843	95	2530	999	562
f_{20}	142	218	2480	180	89	378	1462	235	2074	325	2500	1013	362	3675	822	2545	928	564
f_{21}	37	116	29	79	37	307	38	136	40	195	2500	884	1929	3601	381	2477	1206	518
f_{22}	667	116	74	80	812	309	150	136	210	196	2500	884	199	3604	228	2480	1081	520
f_{23}	27	125	25	92	31	316	34	144	34	204	2500	884	298	3608	287	2484	1044	524
f_{24}	2500	532	1024	124	2500	399	2500	606	2500	1057	2493	1760	2500	3610	2500	2515	2029	563
f_{25}	618	538	668	138	232	347	534	624	283	1073	2500	1835	2500	3683	2500	2641	2031	616
f_{26}	2500	22979	1399	33753	2500	22885	2500	23208	2500	23586	1	25841	2500	25438	2500	24304	1838	22340

個体数を 400 に増加させた場合の各手法の結果を表 4 に示し，全処理時間と最良解を発見した世代数を表 5 に示す．なお，評価回数は前実験結果と同じ 1.0×10^6 回としている．従って，計算の打ち切り世代数は 2500 世代となっている．他の詳細事項は前実験の場合と同一である．

表 4 の結果より，GA は殆どの問題例に対して探索性能の向上が見られた．その中でも，UNDX に関して

は劇的に向上していることから，文献 [6] の指摘通り UNDX の探索性能は個体数の設定値に依存するところが大きいと考えられる．一方，PSO と EP に関しては，解探索性能が改善するものも一部の問題例に対して観測されるが，多くの問題例に対しては，UNDX の性能向上に比べ，大きな変化は見られない．

以上の結果を踏まえ，GA 手法，特に UNDX の探索性能は集団の個体数の設定値に依存する傾向が強いが，nsEEP 等の EP 手法の探索性能は，個体数の設定値に大きく依存しないようである．これは，EP 手法の良い点と考えられ，1 つの解のみに対して適用する突然変異オペレータの有用性とも捉えられる．一方，本研究で取り扱った GA 手法は，交叉オペレータが主要操作となっているため，探索空間における個体集団の多様性が，個体数の設定値に大きく影響していると考えられる．個体集団の多様性をさらに維持するための改良策として，GA に突然変異オペレータを導入することなどが必要と考えられる．

上述の結果を通して，進化的計算手法を中心とする各解法の探索性能に関する特性やパラメータ値の依存性等の見知を観測することができた．しかしながら，本論文で対象とした f_{26} のような極めて困難な問題例の場合には，検討した手法および設定したパラメータ値では，最適解への接近が極めて難しいことも明らかとなった．よって，これらの困難な問題例に対する対応も今後検討する必要があると思われる．

5 むすび

代表的な進化的計算手法である遺伝的アルゴリズム (GA)，進化的プログラミング (EP)，Particle Swarm Optimization (PSO) の研究はこれまでに多く存在しているが，それらの枠組みを超えた基本的な探索性能の比較研究はほとんど行われていない．これらの枠組みを越えた基本的な探索性能を検討することで，現実問題に対してさらに効率的なアルゴリズムへ改良する手がかりや指標が得られるものと考えられる．

そこで，本論文では，関数最小化問題例を取り上げ，GA，EP，PSO の基本的な探索性能について検討した．その結果，個々の進化的計算手法の主要オペレータにより解探索性能に差があることがわかった．個体数を 100 に設定した時，容易な問題例 (変数間依存を含まない問題例) については，BLX-α，SBX，vSBX，nsEEP の各手法は，他の手法に比べて最良解を多く発見した．また，個体数を増加させることによって，GA の探索性能の向上が見られたが，他のアルゴリズムでは，個体数を増やしても，大きな性能の向上は確認できなかった．このことから，GA については個体数を増やすことや突然変異を導入することにより探索性能を向上できると考えられるが，他の手法については個体数を増やすことが必ずしも探索性能の向上につながらないため，子孫の選択方法など他の要因も考慮する必要があると考えられる．加えて，本論文で対象とした f_{26} のような極めて困難な問題例の場合には，検討した手法では，最適解への接近が極めて難しいことも明らかとなった．これらの最適化の困難な問題例に対する対応も今後検討する必要があると思われる．

参考文献

[1] 小野 功，佐藤 浩，小林重信，“単峰性正規分布交叉 UNDX を用いた実数値 GA による関数最適化，" 人工知能学会，Vol.14，No.6，pp.1146–1155，1999.

[2] K. Deb and R.B. Agrawal, “Simulated Binary Crossover for Continuous Serch Space," Complex Systems, Vol. 9, pp.115–148.1995.

[3] K. Deb, “Multi-Objective Optimization using Evolutionary Algorithms," John Wiley Sons, pp.106–114, 2001.

[4] L. J. Eshelman and J.D. Schaffer, “Real Coded Genetic Algorithms and Interval-Schemata," Foundation of Genetic Algorithms 2, pp.187–202, 1993.

[5] 樋口隆英，筒井茂義，山村雅幸，“実数値 GA におけるシンプレクス交叉の提案，" 人工知能学会論文誌，16 巻 1 号 Q，pp.147–155，2001.

[6] 高橋 治，小林重信，“個体間距離に着目した進化型アルゴリズムとその応用," 東京工業大学 博士 (工学) 論文，2001.

[7] J. Kennedy and R. Eberhart, “Particle Swarm Optimization," Proc. of IEEE the International Conference on Neural Networks, pp.1942–1948, 1995.

[8] K. Veeramachaneni, T. Peram and C. Mohan, “Optimization Using Particle Swarms with Near Neighbor Interactions," Proc. of Genetic and Evolutionary Computation (GECCO-2003), LNCS2723, pp.110–121, Springer, 2003.

[9] X. Yao, Y. Lin and G. Lin, “Evolutionary Programming Made Faster," IEEE Transaction on Evolutionary Computation, Vol. 3, No. 2, pp.82–102, July 1999.

[10] 成久洋之，太田真由美，谷口隆裕，片山謙吾，“戦略パラメータを使用しない効率的な EP について," 岡山理科大学紀要，第 41 号，pp.59–68，2005.

[11] H.Narihisa, T. Taniguchi, M. Ohta and K. Katayama, “Exponential Evolutionary Programming without using Strategy Parameter," Proc. of IEEE World Congress on Evolutionary Computations (CEC2006), pp.2559–2566，2006.

[12] 三宮信夫，喜多 一，玉置 久，岩本貴司，“遺伝的アルゴリズムと最適化," システム制御情報ライブラリー 17, 朝倉書店，pp.1–20，1998.

[13] P.J. Ballester and J.N. Carter, “Real-Parameter Genetic Algorithms for Finding Multiple Optimal Solutions in Multi-modal Optimization," Proc. of Genetic and Evolutionary Computation (GECCO 2003)，pp.706–717, 2003.

[14] 樋口隆英，筒井茂義，山口雅幸，“実数値 GA におけるシンプレクス交叉の提案," 人工知能学会論文誌 16 巻 1 号 Q，2001.

[15] T. Bäck, “Evolutionary Algorithms in Theory and Practice," Oxford, 1996.

[16] K. Kohmoto, H. Narihisa and K. Katayama, “Evolutionary Programming Using Exponential Mutation," Proc. of the 6th World Multiconference on Systematics, Cybernetics and Informatics, vol.11, Computer Science2, July 14-18, USA, pp.405–410, 2002.

[17] H. Narihisa, K. Kohmoto and K. Katayama, “Evolutionary Programming with Double Exponential Probability Distribution," Proc. of the Second International Association of Science and Technology for Development (IASTED) International Conference on Artifical Intelligence and Applications (AIA2002)，pp.358–363, 2002.

[18] K. Kohmoto, H. Narihisa and K. Katayama, “Performance of Evolutionary Programming Using Exponential Mutation," Proc. of The 4th Asia-Pacific Comference on Simulated Evolution And Learning (SEAL2002), vol. 2, pp.454–458, 2002.

Performance Comparison of Evolutionary Computation Approaches to the Function Optimization Problem

Koushirou HAYASHI, Takahiro TANIGUCHI, Nobuo UENO,
Kengo KATAYAMA*, Hideo MINAMIHARA* and Hiroyuki NARIHISA*

Graduate School of Engineering,

**Department of Information and Computer Engineering,*

Faculty of Engineering,

Okayama University of Science,

1-1 Ridai-cho, Okayama 700-0005, Japan

(Received October 2, 2006; accepted November 6, 2006)

Representative approaches in evolutionary computation include Genetic Algorithm (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO). These approaches have been applied to real-world problems. Unfortunately, no comparison on the search performance among them has been reported for a specific optimization problem. In this paper we compare the performance of several GAs, EPs, and PSO for the function optimization problem through extensive computational experiments. The results showed that EP with exponential mutation has good performance when the population size is smaller and the solutions obtained by GA approaches can be improved if the population size is larger.

Keywords: evolutionary computation; evolutionary programming; particle swarm optimization; genetic algorithm.