適応的ランダム探索法 RasID の拡張とその評価

平 澤 宏太郎*・宮 崎 弘 幸**・胡 敬 炳**

Enhancement of RasID and Its Evaluation

Kotaro HIRASAWA*, Hiroyuki MIYAZAKI** and Jinglu HU**

In this paper, RasID is enhanced and evaluated systematically. RasID is an abbreviation of Random Search with Intensification and Diversification. RasID can search for a global minimum based on a probability density function, which can be modified adaptively using information based on success and failure of the past searching. As a result, RasID performs intensified search and diversified search iteratively to find a global minimum.

The improvements of RasID are, (1) the probability density function is modified so that each variable can have its own unique density function, and (2) multiple candidates for solutions are created in the search to enhance the efficiency of the searching.

Modified RasID is compared systematically with typical optimization methods such as Evolutionary Programming and Fast Evolutionary Programming introduced recently using 23 different complicated functions. From simulation results, it has been clarified that the performance of Modified RasID is comparable to the ones of EP and FEP in spite of using one individual.

Key Words: random search, intensification, diversification, optimization

1. はじめに

最適化問題は理学、工学等、さまざまな場面で重要であり、そのため、適応的ランダム探索法 RasID (Random Search with Intensification and Diversification) が提案されている^{1)~3)}。RasID は通常の進化論的計算手法 (GA, GP, EP, ES) とは異なり、1 個体の集中化探索と多様化探索により大域的最適解を探索する手法であり、探索のための確率密度関数を過去の探索の成功あるいは不成功の情報により適応的に修正していく点に特徴がある。

EP⁴⁾や ES⁵⁾においてもランダム探索法の探索幅を過去の探索情報により変更しているが、探索のための確率密度関数はガウス分布であり、RasID のように左右非対称のきめこまかな探索ができるようにはなっていない。

本論文では、RasID をつぎの 2 点で改良し、23 個の複雑な目的関数を利用して最近発表された進化論的計算手法 (EP, FEP)⁶⁾ との体系的な比較評価を行なっている。改良点は、個々の変数の探索幅を個別に調整できるようにして、するどい溝のある複雑な関数の最適化にも対応できるようにした事、

および確率密度関数の修正の信頼性を高めるために、1 候補解ではなく複数個の候補解により確率密度関数を変更している点である。

その結果、RasID は局所最適解を持たない関数および局所最適解を持つてはいるがその数が少ない関数については有効である事を明らかにした。また、RasID は一点探索を基本とするため、目的関数の計算に多くの時間を要する実応用問題では、他の多点探索に比較し計算コストの点で有利である事を簡単なコスト計算により明らかにしている。

なお、RasID との比較に、すばやく収束し、かなり精度も高いが次元数が大きくなると限界のある ES や収束は比較的遅いが最も精度が高いと言われている実数値 GA⁷⁾ではなく EP と FEP を選んだ理由は下記のとおりである。

(1) RasID は一点探索であるため進化の操作として突然変異のみを使用している。EP と FEP も同様突然変異のみを活用しており、したがって交叉の影響を除いた比較が可能である。

(2) ES と同様 EP においても適応的な突然変異を使用しており、適応的にランダム探索の確率密度関数を修正していく RasID との比較が可能である。

(3) 探索の速度と精度を向上させるためガウス分布ではなくコーシー分布を利用して探索を行なう FEP は、実数値最適化手法として最近注目されつつある⁶⁾。

* 早稲田大学 北九州市若松区ひびきの 2-2

** 九州大学 福岡市東区箱崎 6-10-1

* Waseda University 2-2 Hibikino, Wakamatsu-ku, Kitakyushu

** Kyushu University 6-10-1 Hakozaki, Higashi-ku, Fukuoka

(Received April 1, 2001)

(Revised June 1, 2002)

Fig. 1 Intensification search and Diversification search

2. RasID の基本構成

n 個の変数, $x_1, x_2, \dots, x_m, \dots, x_M$ からなる最適化問題は以下のように定式化される⁸⁾.

$$\min_x f(x) \quad (1)$$

$$\text{subject to } x \in W, W \subseteq \mathbb{R}^n \quad (2)$$

ここで, W は制約条件を満たす $x = (x_1, x_2, \dots, x_m, \dots, x_M)$ の集合(実行可能領域)を表わしている.

RasIDは, パラメータ探索の成功・不成功情報を活用し, 探索の集中化と多様化を統一した枠組で取り扱う最適化手法である. RasIDでは探索成功率が比較的高い場合は, その近傍には改善解が多数存在すると判断し, 探索効率を高めるため探索範囲を広めながら近傍を重点的に探索する. その後, 解が局所最適解へと接近し, 探索成功率が低くなると, 局所最適解を確実に発見するように探索を狭い範囲で行なう. 局所最適解を発見した後は, 探索成功率がほぼ0に近くなるので, その場合には発見した局所最適解からの脱出を行なうために, 大きい探索幅で広領域を多様に探索する(Fig. 1).

RasIDは以上の探索を繰り返すことによって大域的な最適解を求めることを目的としている.

2.1 RasID 探索の基本アルゴリズム

RasIDの探索の基本手順は以下ようになる.

ステップ1 (初期設定) $x(0) \in W, f(x(0)), k = 0$

ステップ2 (探索) ランダム探索ベクトル $\Theta(k)$ を生成する. もし, $(x(k) + \Theta(k)) \in W$ を満たさなければ $x(k+1) = x(k)$ としてステップ3へ. そうでなければ

(1) $f(x(k) + \Theta(k))$ を計算する.

$f(x(k) + \Theta(k)) < f(x(k))$ であれば探索成功であり

$x(k+1) = x(k) + \Theta(k)$ とする. そうでなければ

(2) 探索失敗であり

$x(k+1) = x(k)$ とする.

ステップ3 (反復) 終了条件を満足すれば終了, そうでなければ $k = k+1$ としてステップ2へ.

Fig. 2 Probability density function for searching Θ_m

ここで

f : 目的関数

$x = [x_1, x_2, \dots, x_m, \dots, x_M]$: 変数ベクトル

$\Theta = [\Theta_1, \Theta_2, \dots, \Theta_m, \dots, \Theta_M]$: ランダム探索ベクトル

である.

上記手順は J. Matyas⁹⁾ および F. J. Solis¹⁰⁾ の手順と基本的に同じである. RasID の特徴はランダム探索ベクトル Θ を生成する確率密度関数 $g(\Theta_m)$ を過去の探索の成功および不成功の情報により適応的に修正している点である.

2.2 ランダム探索の確率密度関数

RasIDのランダム探索の確率密度関数 $g(\Theta_m)$ は Fig. 2 に示すように

$$g(\Theta_m) = \begin{cases} (1-q_m)\beta_m e^{\beta_m \Theta_m} & \Theta_m \leq 0 \\ q_m \beta_m e^{-\beta_m \Theta_m} & \Theta_m > 0 \end{cases} \quad (3)$$

の形式をとる.

RasIDでは式中の二つの変数 $q_m \in [0, 1]$ と β_m を調整することによって集中化探索と多様化探索を実現する. また, ランダム探索変数 Θ_m は(3)式より乱数 $z_m \in (0, 1)$ を使用してつぎのように求める事ができる.

$$\Theta_m = \begin{cases} \frac{1}{\beta_m} \ln \left(\frac{z_m}{1-q_m} \right) & \text{if } 0 \leq z_m \leq (1-q_m) \\ -\frac{1}{\beta_m} \ln \left(\frac{1-z_m}{q_m} \right) & \text{if } 1-q_m < z_m \leq 1.0 \end{cases} \quad (4)$$

β_m はランダム探索 Θ_m の大きさを調整, つまり探索範囲を調整するためのパラメータであり, β_m が大きくなれば探索範囲は狭くなり, 小さくと広くなる. 探索の集中化, 多様化の調整はこの β_m を適応的に変更することにより実現する.

また, β_m は(5)式により計算する(Fig. 3 参照).

$$\beta_m = (\bar{\beta}_m - \beta_m) e^{-\phi I} + \beta_m \quad (5)$$

後述するように(5)式の ϕ は集中化探索時に β_m を制御するためのパラメータ, I は多様化探索時に β_m を制御するためのパラメータである.

q_m はランダム探索変数 Θ_m の正負の方向の探索割合を調整するパラメータであり, $q_m = 0.5$ の場合は正負等確率で変数 Θ を探索することになる.

Fig. 3 Relation β_m and ϕ, I

2.3 集中化探索

集中化探索は局所解の早期発見のためのフェイズである。集中化探索では探索の成功率 P を用い、探索成功率 P が比較的高い場合は、その近傍には改善解が多数存在すると判断し探索範囲を比較的に広くして探索を行なう。その後、解が局所最適解へと接近し、探索成功率 P が低くなれば、探索を狭い範囲で行ない、局所最適解を確実に発見するようにする。局所解を発見した時点では、探索範囲は非常に狭くなっているので、そのような場合には多様化探索に遷移する。上記を実現するため $\phi \geq \phi_{min}$ の場合は集中化探索と考え、 ϕ をつぎのように修正する。

$$\phi = \begin{cases} C_i \phi & \text{if } P \geq P_0 \\ C_d \phi & \text{if } P < P_0 \\ \phi_0 > \phi_{min} & \text{初期およびリセット時} \end{cases} \quad (6)$$

$\phi < \phi_{min}$ となると多様化探索へ遷移する

ここで、 $C_i \geq 1.0$, $0.0 < C_d \leq 1.0$: ϕ を調整する定数

P_0 : 探索成功率を判定するための定数

ϕ_{min} : 集中化探索と多様化探索を判定する定数

である。

2.4 多様化探索

多様化探索は局所解からの脱出を目的としており、 $\phi < \phi_{min}$ を満たしている場合にのみ実行される。多様化探索では探索の成功・不成功情報をもとにパラメータ I の変更を行なう。

$I \leq I_{max}$ の場合、解の探索が失敗するとその探索範囲は狭いと判断して探索範囲を少し広げる。一方、探索が成功すると逆に探索範囲を少し狭める。 $I > I_{max}$ で探索が成功すると、局所最適解から充分脱出したと判断し集中化探索に遷移する。上記を実現するため、パラメータ I の変更はつぎのように行なう。

$$I = \begin{cases} I - \Delta I_1 & \text{if 探索成功} \\ I + \Delta I_2 & \text{if 探索失敗} \\ I_0 < I_{max} & \text{初期およびリセット時} \end{cases} \quad (7)$$

$I > I_{max}$ で探索に成功すると $I = I_0$, $\phi = \phi_0$ として集中化探索に遷移する。

ここで、 $\Delta I_1, \Delta I_2$: I を調整するための変数

I_{max} : 集中化探索に遷移するために使用する定数

2.5 q_m の更新

q_m は探索変数 Θ_m の正負方向を探索する割合を調節するものである。 q_m は探索が成功した場合にのみ以下のように更新される。

$$q_m = \begin{cases} \alpha q_m & \text{if } \Theta_m < 0 \\ q_m & \text{if } \Theta_m = 0 \\ \alpha q_m + (1 - \alpha) & \text{if } \Theta_m > 0 \end{cases} \quad (8)$$

ここで、 $\alpha \in (0, 1)$ は定数である。 q_m は探索が成功した場合、その時に生成した変数の方向に改善解がある可能性が高いと考え、よりその方向の探索が生成されやすいように確率密度関数を調整する。 q_m の初期値は 0.5 とする。

3. RasID の改善

従来の RasID からの改善点は、 $\overline{\beta_m}$ を過去の探索の成功・失敗情報に応じて適応的に変更することにより、探索範囲を調整するパラメータ β_m を各変数ごとに個別に調整している点である。

別の改善点は、確率密度関数の修正の信頼性を高める目的で複数個の候補解により確率密度関数を変更している点である。

3.1 β_m の更新

前述のように、探索範囲を調整するパラメータ β_m を各変数ごとに個別に調整することによってきめこまかな探索が可能となる。たとえば、すべての要素で 0 の場合に最適となる関数において $x(k) = (0, \dots, 10, \dots, 0)$ という解が得られた場合、10 の値を持つ変数のみ比較的に探索範囲を広くして大きな値を生成する必要がある。このため、 $\overline{\beta_m}$ の更新を以下のように行なう。

$$\overline{\beta_m} = \begin{cases} \overline{\beta_m} + si|\Theta_m| & \text{if 探索失敗} \\ \overline{\beta_m} - sd|\Theta_m| & \text{if 探索成功} \end{cases} \quad (9)$$

ここで、 si, sd はそれぞれ $\overline{\beta_m}$ を変更するための正の定数であり、 $\overline{\beta_m}$ の初期値を $\overline{\beta_{m0}}$ とする。探索が失敗した場合、(9) 式はより大きな値を生成した変数ほどつぎの探索ではそれほど大きな値を生成しないように $\overline{\beta_m}$ の調整を行なう。逆に探索が成功した場合、(9) 式はより大きな値を生成して成功した変数ほどつぎの探索ではさらにより大きな値を生成するように $\overline{\beta_m}$ を調整する。これにより要素ごとに異なった探索範囲を持たせることが可能となる。

3.2 複数個の候補解の生成

前節によれば生成されたランダム探索変数 Θ_m の値に基づいて変数 $\overline{\beta_m}$ が調整されることになる。しかし、探索変数はあくまで乱数に基づいて生成されるので変数 $\overline{\beta_m}$ が必ずしも正確に調整されるという保証はない。そこで $\overline{\beta_m}$ の更新の

Fig. 4 Flow of RasID searching

信頼性を高めるために複数個 (M 個) の候補解の生成を以下のように行ない、 $\overline{\beta_m}$ の更新を行なう。

- (1) 新たな候補解を M 個生成する
- (2) M 個の候補解の中で現在の解よりも評価のよいものが複数あれば、その中で最も評価の良いものを改善解として採用し、それに基づいて $\overline{\beta_m}$ を調整する
- (3) すべての候補解が現在の解よりも劣る場合、最も評価の悪いものを解候補として採用し、それに基づいて $\overline{\beta_m}$ の調整を行なう

以上が RasID の第 2 の改善点である。本章で記述した RasID の計算フローを Fig. 4 に示す。

4. RasID の計算コスト

本節では基本的に一点探索である RasID の計算コストを多点探索である GA, ES, EP 等の計算コストと比較するための簡単な式を導き、RasID が計算コストの点でどのような特徴があるのかを整理する。

RasID において、確率密度関数の計算時間 (β_m, q_m の更新時間) を T_p 、探索ベクトル θ の計算時間を T_θ 、目的関数 $f(x)$ の計算時間を T_f 、および変数ベクトル x の更新時間を T_x とすると RasID の 1 ステップに要する計算コスト T_R は概略

$$T_R = T_p + T_\theta + MT_f + T_x \quad (10)$$

となる。ただし、 M は 3.2 節で述べた候補解の数である。

Table 1 RasID parameters for studying intensified search and diversified search

β_m	$\overline{\beta_m}$	α	ϕ_0	ϕ_{min}	I_0	I_{max}	P_0
0.1	100	0.995	0.1	0.001	10	100	0.3
ci	cd	ΔI_1	ΔI_2	si	sd	M	
1.01	0.995	0.02	0.5	0.0	0.0	1	

一方、多点探索においては、個体の数を N 、個体ごとの突然変異の計算時間を T_m 、個体間ごとの交叉の計算時間を T_c 、および淘汰の計算時間を T_s とすると多点探索の 1 ステップに要する計算コスト T_G は概略

$$T_G = N(T_m + \frac{N}{2}T_c + T_f) + T_s \quad (11)$$

となる。

したがって、当然ながら一点探索を基本とする RasID の 1 ステップあたりの計算コストは T_p, T_θ が大きくなければ N 個の多点探索を行なう一般の進化論的計算手法に比較し小さくなる。特に、目的関数 $f(x)$ の計算に多くの時間を要する実際の応用システム、たとえば、船の形状の最適化問題¹¹⁾では数百回の $f(x)$ の計算しか許容されない。このような場合には、 $T_R = MT_f$ 、 $T_G = NT_f$ となり、 $\frac{M}{N}$ が充分小さければ RasID の 1 ステップあたりの計算コストは通常の進化論的計算手法に比較しはるかに小さくなる。

5. RasID の基本特性のシミュレーション

5.1 RasID の集中化探索と多様化探索

本節ではニューラルネットワークの誤差逆伝搬法 (BP 法) のベンチマーク^{12), 13)}として知られている Goldstein and Prince 関数と Six Hump Camel Back 関数を使用して集中化探索と多様化探索により、RasID が大域的最適解を探索する事を検証した。

Goldstein and Price 関数は

$$f(x, y) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)] \times [30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)] \quad (12)$$

Six Hump Camel Back 関数は

$$f(x, y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + 4y^2(y^2 - 1) \quad (13)$$

である。使用した RasID のパラメータは RasID の基本特性を調べる事を目的としたため Table 1 の値を使用した。したがって、 β_m はすべての変数 θ_m について共通な値 $\overline{\beta_m} = 100$ であり、また、解候補の個数も $M = 1$ としている。

Fig. 5 および Fig. 6 に目的関数と β の探索推移を示す。集中化探索の初期の段階では、探索成功率が高いため、探索効率を高めるために β を小さくして、探索幅を比較的大きくしている。局所最適解に接近すると、探索成功率が低くなり、 β を大きくして局所最適解を精度よく求めている。局所最適解が求まると $\phi < \phi_{min}$ が成立するため、多様化探索に移り、 β を小さくすることによって広域探索を実現する。広域探索により改善解が見つかり集中化探索に再び移り、以

Fig. 5 Simulation results of the Goldstein and Price Function

Fig. 6 Simulation results of the Six Hump Camel Back Function

降同様にして多様化探索と集中化探索を繰り返している様子がみてとれる。

5.2 複数の候補解の生成の検討

本節では、複数の候補解を生成する効果について検討する。

シミュレーションの対象として Table A.1 の f_2 を用いた。

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i| \quad (14)$$

この関数は x の要素がすべて 0 になるときに最小値をとる関数であるが、探索の過程において x のある要素がいったん 0 に近付くと、その要素は 0 に近い値を保ちつつ他の要素の値を変化させなければならないというような、要素ごとの探索範囲の調整が必要となるような関数である。

この関数に対して、RasID における候補解の生成数 (M) を変化させてシミュレーションを行なった。Fig. 7 にその結果を示す。また、このときの RasID の内部パラメータのほとんどは文献²⁾に示す代表的な値を使用しているが、 si, sd は β_m の効果がより大きくなるように Table 2 のように設定した。

Fig. 7 によると、候補解の数が大きくなるほど最適値が小

Table 2 RasID parameters for studying the effect of using multiple candidates for solutions

β_m	β_{m0}	α	ϕ_0	ϕ_{min}	I_0	I_{max}	P_0
0.1	2000	0.995	10	0.001	10	100	0.3
ci	cd	ΔI_1	ΔI_2	si	sd		
1.01	0.995	0.1	0.5	1.0	0.5

Fig. 7 Relation between the number of candidates for solutions and optimal value of the objective function

さくなっている。しかしながら候補解の数が大きくなればなるほど、計算に要する時間が増えてしまうので実際に使用する場合には候補解の数を大きくし過ぎないように注意する必要がある。(14) 式の例題では $M = 10$ 程度が適切である。

Table 1 および 2 に示した RasID のパラメータの多くはそれほど問題に依存しない。ランダム探索の範囲が β_{m0} と β_m により決まるのがガウス分布による従来の探索との基本的違いである。 β_m は 0.1 に設定し、複雑な探索問題でなければ、 $\beta_{m0} = 100$ 、やや複雑な問題では β_{m0} を 1000 から 2000 程度に設定すれば充分である。 ϕ_{min} と I_0 は集中化探索から多様化探索へ切替えるためのパラメータで通常 $\phi_{min} \times I_0 < 0.05$ を満足するように設定すれば、各種の問題に適用可能である。

他のパラメータはほとんど問題に依存せず、たとえば $\alpha = 0.995, \phi_0 = 0.1, I_{max} = 100, P_0 = 0.3, C_i = 1.01, C_d = 0.995, \Delta I_1 = 0.1, \Delta I_2 = 0.5, si = 1.0, sd = 0.5$ を通常使用している²⁾。

6. RasID と EP および FEP との総合比較シミュレーション

本節では多点探索で探索効率の優れている文献 6) の EP (Evolutionary Programming) および FEP (Fast Evolutionary Programming) と RasID を 23 種の目的関数を使用して計算コストと探索時の精度について比較評価する。

注目すべきは RasID が一点探索であるのに対し、EP および FEP は多点探索であり、したがって、一点探索でも RasID を使用すれば多点探索と同程度あるいはそれ以上の探索精度が得られる事を明らかにする。

RasID のパラメータは 23 種もの多くの目的関数について

Table 3 Minimum value of objective functions($f_1 \sim f_7$) after searching

	ϕ_0	f_{min}	κ	RasID	EP	FEP
f_1	0.01	0	1	1.2×10^{-6}	2.2×10^{-4}	5.7×10^{-4}
f_2	10	0	0	4.3×10^{-3}	2.6×10^{-3}	8.1×10^{-3}
f_3	0.01	0	1	2.6×10^{-6}	5.0×10^{-2}	1.6×10^{-2}
f_4	0.01	0	0.88	6.1×10^{-3}	2.0	0.3
f_5	0.01	0	0	56.1	6.2	5.1
f_6	10	0	0.94	0.06	577.76	0
f_7	0.01	0	1	1.3×10^{-11}	1.8×10^{-2}	7.6×10^{-3}

Fig. 8 Simulation result of function 4

すべて Table 2 の同じ値を用いることにする。ただし、初期の探索範囲を決定する ϕ_0 だけは、各関数に対して悪影響を及ぼさないような値を調べて、適用している。

シミュレーションで使用する目的関数 $f_1 \sim f_{23}$ は大きく 3 つのタイプに分けることができる。Table A.1 で示される目的関数 $f_1 \sim f_7$ は局所最適解を持たない目的関数である。Table A.2 で示される目的関数 $f_8 \sim f_{13}$ は非常に多くの局所最適解を持つ目的関数であり、局所最適解の数は変数の数が多くなると指数関数的に増加する。そのため一般的に最適化が非常に困難な問題とされている。Table A.3 で示される目的関数 $f_{14} \sim f_{23}$ は少数の局所最適解を持つ目的関数である。

シミュレーションでは RasID の探索回数を $f_1 \sim f_{20}$ の関数においては 1 万回、 $f_{21} \sim f_{23}$ の関数においては 10 万回とし、解候補の数 M は f_1 でのみ 10、その他の関数は 30 とした。またシミュレーションは 50 通りの初期値に対して行ない、その平均値で評価している。

Table 3 は局所最適解を持たない目的関数 $f_1 \sim f_7$ を用いた場合の RasID, EP, FEP の最終的に得られた評価値の平均値を示している。また、表中の κ は RasID の最適解の発見率であり、評価値と最適値との誤差が 10^{-3} 以下である場合を最適解の発見として計算してある。なお、EP および FEP の結果に関しては文献⁶⁾で示されている値を使用しており、以下のシミュレーション結果においても同様である。表中で最も良い値を探索できた手法には下線を引いている。

Fig. 8 は f_4 を用いた場合の RasID, EP および FEP の学習曲線であり、目的関数の計算回数に対する評価の値を示している (RasID の候補解の数 $M = 30$, EP および FEP

Table 4 Minimum values of objective functions($f_8 \sim f_{13}$) after searching

	ϕ_0	f_{min}	κ	RasID	EP	FEP
f_8	0.01	-12569.8	0	-7858.1	-7917.1	-12554.5
f_9	100	0	0	65.9	89.0	4.6×10^{-2}
f_{10}	3	0	0.70	0.42	9.2	1.8×10^{-2}
f_{11}	0.01	0	0.36	1.56×10^{-2}	8.6×10^{-2}	1.6×10^{-2}
f_{12}	10	0	0.72	9.1×10^{-2}	1.76	9.2×10^{-6}
f_{13}	10	0	1	7.0×10^{-8}	1.4	1.6×10^{-4}

の集団の個体の数 $\mu = 100$ を考慮した収束の様子を比較するために横軸は世代ではなく、目的関数の計算の回数としている)。1 万回の学習に要した時間は Linux を使用した 500MHz の PentiumII 上で約 1 秒であった。

Table 3 によると、 f_5 を除けば RasID が EP, FEP の両方もしくはどちらか一方よりも優れた値を算出している。特に f_1, f_3, f_4 , および f_7 では EP, FEP のいずれよりも良い結果が得られている。これらの関数は最適解が比較的探索しやすい所にある。したがって、このような関数に対しては、初期段階では β_m を小さくして比較的大きな探索幅で探索効率をあげ、局所最適解に接近すると β_m を大きくして小さな探索幅で精度高く最適解を探索する RasID の集中化探索の効果が表われている。

また、最終的 (RasID では 1 万回, EP および FEP では 5000 世代) に得られた評価値のみならず、目的関数の計算回数で評価した学習曲線についても、Fig. 8 より RasID が EP, FEP より優れている事がわかる。

Table 4 は多数の局所最適解を持つ目的関数 $f_8 \sim f_{13}$ を用いた場合の RasID, EP, FEP の最終的に得られた評価値の平均値を示している。また、表中の κ は RasID の最適解の発見率であり、探索値と最適値との誤差が 10^{-3} 以下である場合を最適解の発見として計算してある。

たとえば f_{13} の 1 万回の学習に要した時間は同上 PentiumII 上で約 2 秒であった。

Table 4 によるとほとんどの関数について FEP が最も良い結果を得ていることがわかる。 $f_8 \sim f_{13}$ の関数は局所最適解が非常に多い関数であるため、RasID では一つの局所最適解から脱出をしてもすぐにつぎの局所最適解に陥ってしまう。このため $f_8 \sim f_{13}$ の関数については探索を多点で行ない、かつ局所最適解に陥っても脱出容易なコーシー分布の探索を行なう FEP が優れている。しかし RasID は一点探索でありながら、多点探索である EP より良い結果を得ている。

なお、Table 4 の RasID では、目的関数の計算回数は 30 万回であるが (実質 f_{13} では 3 万回前後で最適解に収束している)、EP および FEP では各々 90 万回 (f_8), 50 万回 (f_9), 15 万回 (f_{10}), 20 万回 (f_{11}), 15 万回 (f_{12}), 15 万回 (f_{13}) の目的関数の計算回数を要している。

Table 5 は少数の局所最適解を持つ目的関数 $f_{14} \sim f_{23}$ を用いた場合の RasID, EP, FEP の最終的に得られた評価値の平均値を示している。また、表中の κ は同様に RasID の最適解の発見率である。

Table 5 Minimum values of objective functions ($f_{14} \sim f_{23}$) after searching

	ϕ_0	f_{min}	κ	RasID	EP	FEP
f_{14}	0.01	1	1	1.0	1.66	1.22
f_{15}	100	3.075×10^{-4}	0.50	9.2×10^{-4}	4.7×10^{-4}	5.0×10^{-4}
f_{16}	0.01	-1.03	1	-1.03	-1.03	-1.03
f_{17}	0.01	0.398	1	0.398	0.398	0.398
f_{18}	1	3	1	3.0	3.0	3.02
f_{19}	1	-3.86	1	-3.86	-3.86	-3.86
f_{20}	0.01	-3.32	0.78	-3.29	-3.28	-3.27
f_{21}	0.01	-10.2	0.88	-9.44	-8.27	-5.52
f_{22}	0.01	-10.4	0.92	-9.42	-8.27	-5.52
f_{23}	0.01	-10.5	0.90	-9.92	-9.10	-6.57

たとえば, f_{20} の 10 万回の学習に要した時間は同上 PentiumII 上で約 8 秒であった。

Table 5 よりほぼすべての目的関数で RasID は他手法と同等かまたはそれ以上の良い結果を得ることがわかる。 $f_{14} \sim f_{23}$ の関数は, いくつかの局所最適解を持っているが, 局所最適解の数が少ない。したがって, RasID では少数の局所解から脱出できれば, 大域的最適解を探索する可能性が高くなり, ほぼすべての目的関数で平均値が大域的最適値に近い値になっている。EP や FEP では局所最適解から脱出しても, 以降のきめ細かな評価値の改善を行なうことが難しい。以上のことから少数の局所最適値をもつ目的関数の場合には RasID が有効である。

なお, Table 5 の RasID では目的関数の計算回数は $f_{14} \sim f_{20}$ では 30 万回, $f_{21} \sim f_{23}$ では 300 万回であるが(実質 f_{20} では 1 万回前後で最適値 -3.32 の近傍 -3.29 に収束している), EP および FEP では f_{15} で 40 万回, f_{20} で 2 万回, その他の関数で 1 万回の目的関数の計算回数を要している。

以上, 23 種の目的関数を用いて, 多点探索を使用した EP および FEP と一点探索の RasID の探索性能を比較した。RasID は局所最適解をもたない関数, 局所最適解の少ない関数において EP および FEP に比較して比較的良好な性能を示した。また, 局所最適解の多い関数においても EP より優れた性能を示した。一方, RasID は一点探索の手法であり, かつ局所最適解からの脱出に比較的時間を要するため局所最適解の多い関数では FEP に比較し, 大域的最適解の探索効率が悪くなる事も明らかにした。

7. 結 論

本論文では, 探索の集中化と多様化を同一の枠組で実現する RasID を, (1) β_m を各変数ごとに個別に修正し, さらに, (2) β_m の更新の信頼性を高めるため複数個の候補解を生成するように改善した。RasID は正規分布乱数を探索の基本とする従来のランダム探索法や進化的戦略等とは異なり, 過去の探索の成功・不成功情報を活用して適応的に探索のための確率密度関数を修正し, 探索の集中化と多様化を繰り返す方式である。

また RasID を 23 種の色々な性質をもつ目的関数に適用す

ることによりその性能を明らかにした。用いた 23 種の目的関数は 3 つのグループに分類することができる。ひとつのグループは局所最適解を全く持たず, 大域的最適解のみを持つグループ, ふたつめのグループは局所最適解をたくさん持ち, 変数の数が増加すればその局所最適解が指数的に増加する非常に探索困難なグループ, 最後に局所最適解を持ってはいるがその数が少ないグループである。これらの関数群に対して, 多点探索手法である EP および FEP と RasID との比較評価を行なった。その結果, RasID は局所最適解を持たない関数および局所最適解を持ってはいるがその数が少ない関数について有効であることを明らかにした。また, 関数が非常に複雑で, 大域的最適解まで到達するために局所最適解を何度も乗り越えていかなければならないような関数に対しては, 一点探索の RasID は多点探索の FEP より探索が若干困難である事も明らかにした。

なお, RasID の有効性をさらに検討するために収束性の点で優れている ES および精度の点で進展のみられる実数値 GA との比較を今後行なっていく予定である。

参 考 文 献

- 1) 平澤, 東郷, 胡, 大林, 村田: ニューラルネットワークの適応的ランダム探索最適化手法-RasID-, 計測自動制御学会論文集, **34-8**, 1088/1096 (1998)
- 2) J. Hu, K. Hirasawa and J. Murata: RasID - random search for neural network training, Journal of Advanced Computational Intelligence, **2-4**, 134/141 (1998)
- 3) J. Hu and K. Hirasawa: Adaptive random search approach to identification of neural network model, Proceedings of the 31st ISICIE international symposium on stochastic systems theory and its applications Yokohama, Nov11-12 (1999)
- 4) T. Back and H.-P. Schwefel: An overview of evolutionary algorithms for parameter optimization, Evol. Comput., **1-1**, 1/23 (1993)
- 5) I. Rechenberg: Evolutions strategie - optimierung technischer systeme nach prinzipien der biologischen evolution, Stuttgart, Frommann-Holzboog (1973)
- 6) X. Yao, L. Liu and G. Lin: Evolutionary Programming Made Faster, IEEE tran. on Evolutionary Computation, **3-2**, 83/102 (1999)
- 7) 小野, 喜多, 山村: 実数値 GA とその応用, 人工知能学会誌, **15-2**, 259/266 (2000)
- 8) 山川宏: 最適化デザイン, 培風館 (1993)
- 9) J. Matyas: Random optimization, Automation and Remote Control, **26**, 244/251 (1965)
- 10) F.J. Solis and J.B. Wets: Minimization by random search techniques, Mathematics of Operations Research, **6**, 19/30 (1981)
- 11) 鈴木他: 線形計画法と線形線図, 日本造船学会推進性能研究委員会 第 5 回シンポジウム (1993)
- 12) L.C.W. Dixon and G.P. Szego, Eds: Toward Global Optimization, North Holland (1975)
- 13) L.C.W. Dixon and G.P. Szego, Eds: Toward Global Optimization 2, North Holland (1978)

[著者紹介]

平澤 宏太郎 (正会員)

1966年九州大学大学院工学研究科電気工学専攻修士課程修了。同年(株)日立製作所入社日立研究所勤務。91年8月大みか工場主管技師長。92年12月九州大学システム情報科学研究科教授。2002年9月早稲田大学情報生産システム研究科教授、現在に至る。電気学会、情報処理学会、IEEEの各会員。工学博士。

宮崎 弘幸

1999年九州大学工学部電気系学科卒業。現在、同大大学院システム情報科学研究科電気電子システム工学専攻在学中。

胡 敬 炉 (正会員)

1986年中国中山大学大学院修士修了。同年、同大学電子工学科助手、88年同講師、93年来日、97年九州工業大学情報工学研究科博士後期過程修了。同年九州大学ベンチャービジネスラボラトリ非常勤研究員を経て、同大学システム情報科学研究科助手、現在に至る。

《付 録》

A. シミュレーションに用いた目的関数

Table A.1 Objective functions having no local minima
(n : The number of variables, W : The feasible solutions, f_{min} : Minimum)

Test function	n	W	f_{min}
$f_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	$[-100, 100]^n$	0
$f_4(x) = \max_i \{ x_i , 1 \leq i \leq n \}$	30	$[-100, 100]^n$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^n$	0
$f_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	$[-100, 100]^n$	0
$f_7(x) = \sum_{i=1}^n ix_i^4 + random[0, 1]$	30	$[-1.28, 1.28]^n$	0

Table A.2 Objective functions having a large number of local minima
(n : The number of variables, W : The feasible solutions, f_{min} : Minimum)

Test function	n	W	f_{min}
$f_8(x) = \sum_{i=1}^n -x_i \sin \sqrt{ x_i }$	30	$[-500, 500]^n$	-12569.5
$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos 2\pi x_i + 10]$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i \right) + 20 + \exp(1)$	30	$[-32, 32]^n$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]^n$	0
$f_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1),$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	30	$[-50, 50]^n$	0
$f_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1) [1 + \sin^2(2\pi x_n)] \right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0

Table A.3 Objective functions having a small number of local minima
(n : The number of variables, W : The feasible solutions, f_{min} : Minimum)

Test function	n	W	f_{min}
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \left\{ j + \sum_{i=1}^2 (x_i - a_{ij})^6 \right\}^{-1} \right]^{-1}$	2	$[-65.536, 65.536]^n$	0.998
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, 5]^n$	3.075×10^{-4}
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]^n$	3
$f_{19}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2 \right]$	3	$[0, 1]^n$	-3.86
$f_{20}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2 \right]$	6	$[0, 1]^n$	-3.32
$f_{21}(x) = -\sum_{i=1}^5 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.2
$f_{22}(x) = -\sum_{i=1}^7 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.4
$f_{23}(x) = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.5