世代間移動ベクトル群の収束点推定法

村田, 昇
早稲田大学理工学術院

西井, 龍映
九州大学マス・フォア・インダストリ研究所

髙木, 英行
九州大学大学院芸術工学研究院

裴, 岩
会津大学コンピュータ理工学部

出版情報：進化計算シンポジウム. 2014, 2014-12-21. 進化計算学会
バージョン：
権利関係：

世代間移動ベクトル群の収束点推定法

村田昇 †, 西井龍映 ††, 高木英行 ‡, 裴岩 ‡‡

早稲田大学理工学術院 †, 九州大学マス・フォア・インダストリ研究所 ††,
九州大学大学院芸術工学研究院 ‡, 会津大学コンピュータ理工学部 ‡‡

1 はじめに

進化計算の主要研究の１つは，最適化の高度化である．新しい進化計算アルゴリズムの開発[1, 2, 6]，進化計算演算の改良[3, 4]，fitness空間の近似による大局的最適解の粗く速い推測[5]，等の色々な取り組みがなされて来た．

探索が成功する進化計算は，時には局所最適解に引き寄せながらも，逐次探索で大局的最適解に向かっていく．したがって，世代間の移動ベクトルは，大局的最適解方向を推察する有力な情報である．特に，進化戦略，差分進化，群知能のように，第k世代の探索点と第$k+1$世代の探索点との間に一対一対応がある場合，世代間の個体の移動ベクトルがFig. 1のように複数得られる．d次元探索空間でのこれら複数の移動ベクトルが向かっている点（Fig. 1の★印）は大局的最適解推測に有効な新しい探索点となりえる[7, 8, 10]．

本論文は，遺伝的アルゴリズムや遺伝的プログラミングのようにn個の親個体群から次世代のn個の子個体群を生成する進化計算ではなく，上述のように親個体と子個体とが一対一対応するような進化計算を扱う．その上で，本論文の第1の目的は，これら複数の親子間の移動ベクトルが向かう方向，すなわち，Fig. 1の★印位置を推定する方法を示すことである．第2の目的は，この推測点が最適化の探索における有力な個体になり得ることを示すことである．

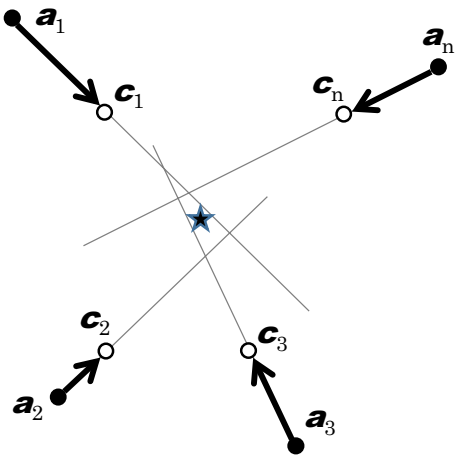

Fig. 1 d次元探索空間上の第k世代の個体（$\boldsymbol{a}_i,\ i=1,2,...,n$）と第$k+1$世代の個体（$\boldsymbol{c}_i,\ i=1,2,...,n$）が構成する世代間移動ベクトルが向かう先（★）．

以下第2節ではこの★印位置の推定方法と，行列演算を行わない近似方法，逐次方法を示す．第3節ではその推定した★印位置は有力なエリート個体になり得ることをfitness順位で示す．

2 世代間移動ベクトルが目指す点

2.1 記号の定義

個体数nでd次元の探索空間を前節で述べた進化計算で探索する際，第i番目の親個体を$\boldsymbol{a}_i$，その子個体を$\boldsymbol{c}_i$，移動ベクトルを$\boldsymbol{b}_i=\boldsymbol{c}_i-\boldsymbol{a}_i$とする．$\{(\boldsymbol{a}_i,\boldsymbol{c}_i),\ i=1,2,...,n;\ \boldsymbol{a}_i,\boldsymbol{c}_i\in\mathbb{R}^d\}$．また，$\boldsymbol{b}_i$の単位方向ベクトル$\boldsymbol{b}_{0i}=\boldsymbol{b}_i/\|\boldsymbol{b}_i\|\quad(\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{b}_{0i}=1)$も定義する．なお，本論文でのベクトル表現は縦ベクトルとしている．

n個のベクトル$\boldsymbol{a}_i$を基点とするベクトル$\boldsymbol{b}_i$が与えられた時，これらの$\boldsymbol{b}_i$を有向線分としそれらを延長して作られた直線群に最も近い点を$\boldsymbol{x}\in\mathbb{R}^d$とする．この$\boldsymbol{x}$を求めることが本論文の第1の目的である．

Estimation Methods of the Convergence Point of Moving Vectors Between Generations

† Noboru MURATA (noboru.murata @ eb.waseda.ac.jp)
†† Ryuei NISHII (nishii @ imi.kyushu-u.ac.jp)
‡ Hideyuki TAKAGI (http://www.design.kyushu-u.ac.jp/~takagi/)
‡‡ Yan PEI (http://web-ext.u-aizu.ac.jp/~peiyan/)
School of Science and Engineering, Waseda University (†)
Institute of Mathematics for Industry, Kyushu University (††)
Faculty of Design, Kyushu University (‡)
School of Computer Science and Engineering, the University of Aizu (‡‡)

2.2 推定法1：行列演算を用いる厳密解

前節の有向線分から作られる直線の式は式(1)で表される．

$$\text{直線}i:\ \boldsymbol{a}_i + t_i\boldsymbol{b}_i,\ t_i \in \mathbb{R} \tag{1}$$

求める点$\boldsymbol{x}$と直線群との距離は式(2)で表される．

$$\mathrm{J}(\boldsymbol{x}, \{t_i\}) = \sum_{i=1}^{n} \|\boldsymbol{a}_i + t_i\boldsymbol{b}_i - \boldsymbol{x}\|^2 \tag{2}$$

したがってこの距離を最小にする点$\boldsymbol{x}$と直線のパラメータ$\{t_i,\ i=1,\ldots,n\}$を求める式(3)を解くことで求める解が得られる．

$$\min_{\boldsymbol{x},\{t_i\}} \mathrm{J}(\boldsymbol{x}, \{t_i\}) = \min_{\boldsymbol{x}} \sum_{i=1}^{n} \min_{t_i} \|\boldsymbol{a}_i + t_i\boldsymbol{b}_i - \boldsymbol{x}\|^2 \tag{3}$$

点$\boldsymbol{x}$を固定すれば，この点から直線i上の最短点を与えるt_iは個別に決められる．これは点$\boldsymbol{x}$から直線iへの直交射影となるので，

$$\boldsymbol{b}_i^{\mathrm{T}}(\boldsymbol{a}_i + t_i\boldsymbol{b}_i - \boldsymbol{x}) = 0 \quad (\text{直交条件}) \tag{4}$$

この式より

$$t_i = \frac{\boldsymbol{b}_i^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{a}_i)}{\|\boldsymbol{b}_i\|^2} \tag{5}$$

となる．($\boldsymbol{x}\cdot\boldsymbol{y} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$ を用いて書いていることに注意)

式(5)を式(3)の$\|\cdot\|$部分に代入して整理すると

$$\boldsymbol{b}_i\frac{(\boldsymbol{b}_i)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{a}_i)}{\|\boldsymbol{b}_i\|^2} - (\boldsymbol{x} - \boldsymbol{a}_i) = \left\{\frac{\boldsymbol{b}_i\boldsymbol{b}_i^{\mathrm{T}}}{\|\boldsymbol{b}_i\|^2} - \mathrm{I}_d\right\}(\boldsymbol{x} - \boldsymbol{a}_i) \tag{6}$$

となる．I_dは単位行列である．ここで

$$\mathrm{I}_d - \frac{\boldsymbol{b}_i\boldsymbol{b}_i^{\mathrm{T}}}{\|\boldsymbol{b}_i\|^2} = \mathrm{I}_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}} = H_i \tag{7}$$

とおくと，式(2)から$\{t_i\}$を消去した目的関数

$$\mathrm{J}(\boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{x} - \boldsymbol{a}_i)^{\mathrm{T}} H_i^{\mathrm{T}} H_i (\boldsymbol{x} - \boldsymbol{a}_i) \tag{8}$$

が得られるので，これを$\boldsymbol{x}$に関して最小化すれば良い．求める点$\boldsymbol{x}$の推定値$\hat{\boldsymbol{x}}$は，$\boldsymbol{x}$の各要素で偏微分し，0とおけばよい．

$$\begin{aligned}\frac{\partial \mathrm{J}(\boldsymbol{x})}{\partial \boldsymbol{x}} &= 2\sum_{i=1}^{n} H_i^{\mathrm{T}} H_i (\boldsymbol{x} - \boldsymbol{a}_i) \\ &= 2\left\{\left(\sum_{i=1}^{n} H_i^{\mathrm{T}} H_i\right)\boldsymbol{x} - \left(\sum_{i=1}^{n} H_i^{\mathrm{T}} H_i \boldsymbol{a}_i\right)\right\} \\ &= 0\end{aligned} \tag{9}$$

これより推定値は

$$\hat{\boldsymbol{x}} = \left(\sum_{i=1}^{n} H_i^{\mathrm{T}} H_i\right)^{-1} \left(\sum_{i=1}^{n} H_i^{\mathrm{T}} H_i \boldsymbol{a}_i\right) \tag{10}$$

で与えられる．

ところで，H_iは$H_i^{\mathrm{T}} H_i = H_i^2 = H_i$という性質を持つ(射影行列)ので，式(10)は以下のように書き直すことができる．

$$\begin{aligned}\hat{\boldsymbol{x}} &= \left(\sum_{i=1}^{n} H_i\right)^{-1} \left(\sum_{i=1}^{n} H_i \boldsymbol{a}_i\right) \\ \therefore \quad &= \left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\right\}^{-1} \left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\boldsymbol{a}_i\right\}\end{aligned} \tag{11}$$

2.3 推定法2：行列演算を行わない近似解

推定法1の式(11)を求めるには行列演算を行う必要がある．元々世代間移動ベクトルは大局的最適解の方向に向かっていることを期待しているが，大局的最適解そのものの一点に向かっている訳ではない．したがって，近似計算でよいので$\hat{\boldsymbol{x}}$を早く計算して最適化の一助に利用した方が有利である，との考えもあろう．

以下では，式(11)をNeumann級数展開し低次の項のみで近似計算することを考える．式(11)は以下のように展開される．

$$\begin{aligned}\hat{\boldsymbol{x}} &= \left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\right\}^{-1} \left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\boldsymbol{a}_i\right\} \\ &= \frac{1}{n}\left\{I_d + \left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right) + \right. \\ &\qquad \left.\left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)^2 + \left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)^3 + \cdots\right\} \\ &\qquad \times \left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\boldsymbol{a}_i\right\}\end{aligned} \tag{12}$$

この級数展開の0次の項(I_dの部分)を用いたものは，式(13)になり，逆行列演算や行列のメモリが不要になり計算時間が簡便になる．計算機資源に応じて用いる展開次数を増やすことで精度を上げることが可能である．

$$\begin{aligned}\hat{\boldsymbol{x}} &\approx \frac{1}{n}\left\{\sum_{i=1}^{n} \left(I_d - \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\right)\boldsymbol{a}_i\right\} \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{a}_i - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{a}_i\end{aligned} \tag{13}$$

$\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}$は行列になるので2次元のメモリが必要である．しかし，変形することでベクトルの1次元メモリを使って演算できる．すなわち，

$$\begin{aligned}\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{a}_i &= \boldsymbol{b}_{0i}(\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{a}_i)\\ &= \boldsymbol{b}_{0i}(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{b}_{0i})\ (()\text{内はスカラ})\\ &= (\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{b}_{0i})\boldsymbol{b}_{0i} \qquad (14)\end{aligned}$$

これを式(13)に代入すると

$$\therefore \hat{\boldsymbol{x}} \approx \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{a}_i - \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{b}_{0i})\boldsymbol{b}_{0i} \qquad (15)$$

なお，式(13)で一般に$\frac{1}{n}\sum_{i=1}\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}} \neq 0$であるから，推定量が位置共変にならないという点に注意する必要がある．

2.4 推定法3：行列演算を行わない逐次計算

式(3)の$\boldsymbol{x}$と$\{t_i\}$の最小化を交互に繰り返す方法を考える．この方法は，収束点においてNeumann級数展開の式が成り立っているので，逐次的に第2.4節のNeumann級数展開を行って近似解を求めているともいえる．

点$\boldsymbol{x}$が与えられたとき，$\boldsymbol{x}$に最も近い直線i上の点を$\boldsymbol{y}_i(\boldsymbol{x})$とする．

$$\boldsymbol{y}_i(\boldsymbol{x}) = \boldsymbol{a}_i + t_i(\boldsymbol{x})\boldsymbol{b}_i \qquad (16)$$

ただし，$t_i(\boldsymbol{x})$は式(5)より式(17)で表される．

$$t_i(\boldsymbol{x}) = \frac{\boldsymbol{b}_i^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{a}_i)}{\|\boldsymbol{b}_i\|^2} \qquad (17)$$

一方，$\boldsymbol{y}_i(\boldsymbol{x})$, $(i = 1, 2, ..., n)$が与えられたとき，これらからの距離の総和が最短となる点$\boldsymbol{x}'$は，$\boldsymbol{y}_i(\boldsymbol{x})$の重心であるので，式(18)で表される．

$$\boldsymbol{x}' = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_i(\boldsymbol{x}) \qquad (18)$$

以上より，次の逐次解法が得られる．

逐次解法

Step 1 $\boldsymbol{x}$の初期化．例えば，$\boldsymbol{x} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{c}_i$

Step 2 $\boldsymbol{x}$から直線に射影し，式(16)で$\boldsymbol{y}_i(\boldsymbol{x})$を求める．

Step 3 式(18)で得られる射影点の重心$\boldsymbol{x}'$を新たな$\boldsymbol{x}$とする．

Step 4 収束するまでStep 2へ

本節の逐次法の収束性は保障される．式(18)の更新によって明らかに

$$\sum_{i=1}^{n}\|\boldsymbol{y}_i(\boldsymbol{x}) - \boldsymbol{x}'\|^2 \leq \sum_{i=1}^{n}\|\boldsymbol{y}_i(\boldsymbol{x}) - \boldsymbol{x}\|^2 \qquad (19)$$

であり，また式(16)の更新によって

$$\sum_{i=1}^{n}\|\boldsymbol{y}_i(\boldsymbol{x}') - \boldsymbol{x}'\|^2 \leq \sum_{i=1}^{n}\|\boldsymbol{y}_i(\boldsymbol{x}) - \boldsymbol{x}'\|^2 \qquad (20)$$

となるので，式(8)において，$\mathrm{J}(\boldsymbol{x}') \leq \mathrm{J}(\boldsymbol{x})$であり，収束性が保証される．

3 有効性の評価

第2節で求めた移動ベクトルが向かう推定座標$\boldsymbol{x}$が有力なエリート個体になり得ることを，表1のベンチマーク関数を使った評価実験で示す．具体的には，差分進化を用い，前世代のtarget vectorと現世代のtarget vectorが異なる場合に形成される世代間移動ベクトルが向かう点$\boldsymbol{x}$を求める．次に，この$\boldsymbol{x}$＋全個体のfitness順位を求め，ベンチマーク関数毎に提案3手法で得られた$\boldsymbol{x}$の順位をTable 2に示して探索個体群と比較する．

Table 1 実験に用いたCEC2005のベンチマーク関数[9]．(Sh=Shifted, Rt=Rotated, GB=Global on Bounds, NS=Non-Separable)

No.	name	Modality	Sh	Rt	GB	NS	Search range	Optimum fitness
f_1	Sphere	Uni-modal	√					-450
f_2	Schwefel 1.2		√			√		-450
f_3	Elliptic		√	√		√	$[-100, 100]$	-450
f_4	f_2 with Noise		√			√		-450
f_5	Schwefel 2.6				√	√		-310
f_6	Rosenbrock	Multi-modal	√			√	$[-100, 100]$	390
f_7	Griewank		√	√		√	$[0, 600]$	-180
f_8	Ackley		√	√	√	√	$[-32, 32]$	-140
f_9	Rastrigin		√				$[-5, 5]$	-330
f_{10}	Rastrigin		√	√		√	$[-5, 5]$	-330
f_{11}	Weierstrass		√	√		√	$[-0.5, 0.5]$	90
f_{12}	Schwefel 2.13		√			√	$[\pi, \pi]$	-460
f_{13}	Expanded F8F2		√			√	$[-3, 1]$	-130
f_{14}	Scaffer F6		√	√		√	$[-100, 100]$	-300

実験条件は，差分進化演算（DE/rand, F=1, CR=1），2, 5, 10次元のベンチマーク関数に，(DE/rand, F=1, CR=1，40個体）の差分進化を100世代までの探索で，実験環境は，Windows 8.1 (x84) のPC (Intel(R) Core(T) i7-4500 U CPU@ 1.80GHz 2.39GHz, 4GRAM) 上のMatlab R2011bである．推定法3は第2.4の反復を10回繰り返して$\boldsymbol{x}$を求めた．

得られた$\boldsymbol{x}$は探索の高速化に色々利用できると考えられる．代表的な方法は，最悪個体と入れ換えることである[7, 8, 10]．しかし，$\boldsymbol{x}$を探索に組み込んでの高速化効果の評価は今後の研究に譲り，本論文では，$\boldsymbol{x}$が有力なエリート候補になり得ることに焦点を当てて評価する．

また，本論文での推定方法を組み込むことによって増加する計算コストを評価するため，14関数で100世代までの探索したCPU時間を計測した．提案手法を通常差分進化を組み込んだ場合の平均CPU時間と組み込まない通常差分進化の場合の平均CPU時間比をTable 3に示す．

Table 3　提案推定法を組み込んだ場合のCPU時間の増加比率＝(差分進化+推定法)/(差分進化のみ)．14関数（2, 5, 10次元）を100世代まで計算して計測．

関数の次元	推定法1	推定法2	推定法3
2	1.09	2.27	2.35
5	1.37	1.99	1.81
10	1.30	1.88	1.53

4　考察

予想どおり，本提案手法は，単峰性関数（f_1〜f_5）には有力解となる点を推定できそうである．今後この推定位置をエリート個体として探索に利用することで進化計算の高速化に寄与することが期待できる．

14のbenchmark関数中，f_8, f_{11}, f_{12}, f_{14}の4関数に対して本提案手法はまったく有効に働かなかった．しかしこれらのfitness景観（Fig. 2）を見れば，探索が徐々に最適解に近づく形状ではないので当然とも言える．

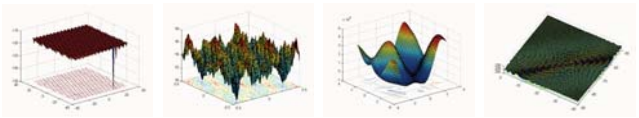

Fig. 2　提案手法がまったく効かなかった4関数．左からf_8, f_{11}, f_{12}, f_{14}．図は文献[9]より．

第2に近似計算を考察する．推定法2ではNeumann級数展開した第0次の項で推定法1の近似をした．しかし，次元数の低い単峰性関数（f_1〜f_5）で推論法1と3に対して結果に違いが出ている．推定する集中点$\boldsymbol{x}$は大局的最適解そのものではないとはいうもの，探索に利用するには近似度合が大きすぎるのであろう．10回反復計算をした推定法3は推定法1と遜色がないので，級数展開した項をもう少し計算すれば同様の性能が得られることが判るが，CPU時間の比較を見る限り，推定法2の次数項を上げて計算することのメリットは少ないと思われる．

逆行列計算を含み最もCPU時間がかかると考えられた推定法1が他の推定法よりも速い．Matlabの行列演算は一般に高速であることが言われているので，このためと思われる．しかし，行列演算ベースでない，例えばC言語でcodingをして比較していないので，推定法2と3が推定法1に比べて今回の実験結果程の差がでるのかどうかは不明である．

5　結論

本論文では，世代間の個体の移動方向ベクトル群がどこに収束しようとしているかを計算する方法を提案した．個体群全体が最適解に向かう単峰性関数では，早くから最適解に近いと思われる推定収束点を求めることができ，進化計算の高速化に寄与できる可能性が示された．この寄与のさせ方には色々考え得られ，今後の研究で取り組んでいく．

多峰性関数では，個体は最適解だけでなく局所最適解にも収束しようとするため，移動ベクトル群は異なる複数の収束点に向かおうとしているはずである．この異なる収束点毎に移動ベクトルを分類できれば，進化計算の高速化に寄与するだけでなく，局所最適解を探す新しいニッチ手法にも使える可能性を秘めている．今後の大きな研究方向の１つである．

参考文献

1) Back, T., Hammel, U., and Schwefel, H.-P., “Evolutionary computation: Comments on the history and current state,” IEEE Trans. on Evolutionary Computation, vol.1, no.1, pp.3–17 (1997).

2) Coello Coello, C.A. “Evolutionary multi-objective optimization: A historical view of the field,” IEEE Computational Intelligence Magazine (2006).

3) Das, S. and Suganthan, P.N. “Differential evolution: A survey of the state-of-the-art,” IEEE Trans. on Evolutionary Computation, vol.15, no.1, pp.4–31 (2011).

4) Eiben, Á.E., Hinterding, R., and Michalewicz, Z., “Parameter control in evolutionary algorithms,” IEEE Trans. on Evolutionary Computation, vol.3, no.2, pp.124–141 (1999).

Table 2　世代間移動ベクトルの推定された収束点$\boldsymbol{x}$のfitness順位（第1〜41位）．第2〜25世代，第26〜50世代，第51〜75世代，第76〜100世代の平均順位．

(a) 推定法 1 によって得られた集中点$\boldsymbol{x}$の25世代間平均順位

世代	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
2-D関数	2次元関数評価時の平均順位				5次元関数評価時の平均順位				10次元関数評価時の平均順位			
f_1	**9.1**	**7.2**	**11.5**	**9.6**	**2.8**	**6.0**	**4.8**	**2.6**	**1.2**	**4.9**	**7.9**	**17.8**
f_2	**10.5**	**6.7**	**11.4**	**10.5**	**4.8**	**1.8**	**2.2**	**4.6**	**2.7**	**1.8**	**5.4**	**12.8**
f_3	**18.2**	**19.8**	**15.2**	**15.0**	**7.5**	**7.7**	**6.5**	**14.4**	**5.7**	**10.0**	**16.2**	**19.4**
f_4	**12.0**	**11.4**	**11.2**	**13.6**	**4.3**	**4.0**	**7.0**	**8.1**	**2.7**	**10.5**	**17.8**	**15.0**
f_5	29.4	28.2	26.2	37.6	**19.7**	**16.0**	**11.8**	**16.8**	**7.8**	**11.8**	**12.5**	**13.1**
f_6	31.0	34.9	33.7	37.3	**4.0**	**13.8**	**13.5**	27.0	**1.8**	**4.3**	**12.2**	**7.3**
f_7	**14.5**	34.8	38.4	24.8	**4.9**	**3.9**	**4.7**	**13.2**	**2.5**	**3.5**	**15.4**	**17.7**
f_8	37.0	40.0	41.0	41.0	37.7	41.0	41.0	41.0	37.0	41.0	41.0	38.5
f_9	34.8	25.5	**11.5**	**7.6**	26.4	35.4	38.1	41.0	**10.0**	30.1	39.6	38.4
f_{10}	26.3	34.2	**11.3**	**10.8**	**14.7**	35.8	40.2	37.7	**7.6**	22.7	33.8	39.2
f_{11}	38.3	40.3	38.8	37.6	36.2	40.8	38.2	39.8	36.6	39.2	39.3	40.7
f_{12}	37.9	41.0	41.0	41.0	34.3	40.1	41.0	41.0	39.0	40.6	41.0	41.0
f_{13}	32.2	39.8	22.3	**7.9**	**5.0**	23.7	39.3	39.5	**2.8**	15.6	23.6	24.9
f_{14}	35.7	30.7	38.7	34.8	35.4	38.4	38.6	40.2	31.9	37.0	40.9	38.4

(b) 推定法2によって得られた集中点$\boldsymbol{x}$の25世代間平均順位

世代	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
5-D関数	2次元関数評価時の平均順位				5次元関数評価時の平均順位				10次元関数評価時の平均順位			
f_1	37.5	41.0	41.0	41.0	**3.9**	28.8	40.2	39.6	**1.2**	**1.8**	**6.9**	**15.6**
f_2	36.5	41.0	41.0	41.0	**6.5**	28.2	41.0	41.0	**2.8**	**3.5**	**8.0**	20.8
f_3	31.8	36.6	41.0	41.0	**5.7**	**5.4**	8.**4**	**10.4**	**6.5**	**8.7**	**12.8**	**13.6**
f_4	37.1	41.0	41.0	41.0	**9.6**	33.0	40.2	41.0	**3.4**	**4.5**	**12.3**	**12.4**
f_5	39.3	41.0	41.0	41.0	31.6	41.0	41.0	41.0	**8.6**	**10.7**	**19.4**	25.2
f_6	37.5	40.5	41.0	41.0	**4.5**	**18.3**	40.0	41.0	**1.8**	**3.4**	**5.2**	**5.6**
f_7	28.8	41.0	41.0	41.0	**6.9**	**16.0**	32.0	38.6	**1.8**	**1.9**	**11.0**	**16.5**
f_8	36.5	40.3	41.0	41.0	35.0	41.0	40.0	40.9	37.6	41.0	41.0	41.0
f_9	32.6	40.8	41.0	41.0	24.2	37.4	40.9	40.8	**6.7**	29.4	39.0	37.8
f_{10}	32.0	40.7	41.0	41.0	**17.6**	29.0	39.2	37.8	**7.3**	22.4	29.4	38.6
f_{11}	37.7	41.0	41.0	41.0	37.1	41.0	41.0	41.0	36.5	40.0	41.0	40.5
f_{12}	39.4	41.0	41.0	41.0	37.5	40.7	41.0	41.0	37.4	40.9	41.0	40.8
f_{13}	30.6	40.3	41.0	41.0	**6.3**	29.0	36.4	39.2	**2.7**	**9.3**	**13.5**	21.0
f_{14}	40.1	40.7	41.0	41.0	30.3	37.8	35.4	39.0	34.5	37.3	40.6	40.6

(c) 推定法3によって得られた集中点$\boldsymbol{x}$の25世代間平均順位

世代	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
10-D関数	2次元関数評価時の平均順位				5次元関数評価時の平均順位				10次元関数評価時の平均順位			
f_1	**9.1**	**6.7**	**11.1**	**9.4**	**2.8**	**6.0**	**4.8**	**2.6**	**1.2**	**5.4**	**9.4**	**18.1**
f_2	**10.1**	**6.6**	**11.0**	**10.3**	**4.8**	**1.7**	**2.2**	**4.3**	**2.7**	**1.8**	**4.2**	**10.9**
f_3	21.2	24.8	**18.6**	**15.0**	**7.0**	**8.2**	**7.1**	**12.7**	**5.8**	**9.4**	**15.4**	**18.4**
f_4	**10.3**	**9.8**	**9.3**	**8.0**	**6.6**	**2.0**	**6.5**	**6.6**	**4.0**	**7.3**	**16.0**	**22.2**
f_5	30.9	29.8	26.2	38.7	19.7	**16.0**	**11.8**	**16.8**	**7.8**	**11.4**	**11.7**	**11.5**
f_6	32.0	34.0	36.3	37.4	**4.0**	**12.9**	**12.6**	28.2	**1.8**	**4.4**	**10.4**	**7.1**
f_7	**13.9**	34.6	40.6	**27.4**	**4.9**	**3.9**	**4.7**	**12.3**	**2.5**	**4.2**	**16.2**	**16.0**
f_8	35.5	39.1	41.0	39.5	35.1	38.5	37.1	38.6	37.4	37.8	36.3	38.6
f_9	34.5	25.4	**11.4**	**7.4**	26.4	35.4	36.9	40.6	**10.0**	29.6	37.9	33.2
f_{10}	26.7	35.3	**11.1**	**10.5**	**14.3**	31.1	36.6	35.5	**7.6**	22.8	31.3	37.9
f_{11}	38.0	40.2	41.0	37.5	35.4	38.4	37.4	38.2	36.5	35.7	38.0	37.7
f_{12}	36.0	40.4	40.1	39.2	35.4	37.4	37.5	37.0	38.9	38.2	38.2	34.6
f_{13}	31.2	36.4	27.1	**7.6**	**5.2**	23.3	35.9	37.2	**3.2**	**12.9**	20.8	25.0
f_{14}	31.1	31.0	33.6	35.2	36.4	33.7	37.2	33.2	32.6	34.4	37.5	38.5

5) Jin, Yaochu, “A Comprehensive survey of fitness approximation in evolutionary computation,” Soft Computing, Springer, Vol.9, No.1, pp.3–12 (2005).

6) Mullen, R.J., Monekosso, D., Barman, S., and Remagnino, P., “A review of ant algorithms,” Expert Systems with Applications, vol.36 , no.6, pp.9608–9617 (2009).

7) Pei, Y. and Takagi, H., “Fourier analysis of the fitness landscape for evolutionary search acceleration,” IEEE Congress on Evolutionary Computation (CEC2012), pp.1–7, Brisbane, Australia (June, 2012).

8) Pei, Y. and Takagi, H., “Comparative study on fitness landscape approximation with Fourier transform,” 6th Int. Conf. on Genetic and Evolutionary Computing (ICGEC2012), Kitakyushu, Japan, pp.400–403 (Aug., 2012).

9) Suganthan, P. N., Hansen, N., Liang, J. J., et al., “Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization”, `https://www.lri.fr/~hansen/Tech-Report-May-30-05.pdf`

10) 高木英行, 印具毅雄, 大西圭「単峰性関数当てはめによるGA収束高速化」知能と情報（日本知能情報ファジィ学会誌）, vol.15, no.2, pp.219–229 (2003）.