Analytical Estimation of the Convergence Point of Populations

Murata, Noboru
School of Science and Engineering, Waseda University : Professor

Nishii, Ryuei
Institute of Mathematics for Industry, Kyushu University : Professor

Takagi, Hideyuki
Faculty of Design, Kyushu University : Professor

Pei, Yan
School of Computer Science and Engineering, the University of Aizu : Assistant Professor

出版情報：Proceedings of 2015 IEEE Congress on Evolutionary Computation, 2015-05-25. IEEE
バージョン：
権利関係：

Analytical Estimation of the Convergence Point of Populations

Noboru Murata*, Ryuei Nishii†, Hideyuki Takagi‡, and Yan Pei§
*School of Science and Engineering, Waseda University, Tokyo, Japan (noboru.murata @ eb.waseda.ac.jp)
†Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan (nishii @ imi.kyushu-u.ac.jp)
‡Faculty of Design, Kyushu University, Fukuoka, Japan (http://www.design.kyushu-u.ac.jp/~takagi/)
§School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu, Japan (http://web-ext.u-aizu.ac.jp/~peiyan/)

***Abstract*—We propose methods of estimating the convergence point for the moving vectors of individuals between generations or evolution paths and show that the estimated convergence point can be useful information for accelerating evolutionary computation (EC). As the first stage of this new approach, we do not combine the proposed methods with EC search in this paper, but rather evaluate how power an individual the the estimated convergence point is by comparing fitness values. Through experimental evaluations, we show that the estimated point can be a powerful elite for unimodal fitness landscapes and that clustering moving vectors according to the aimed points is the next research target for multimodal fitness landscape.**

I. Introduction

One of the major main objectives in evolutionary computation (EC) research is to improve EC convergence and find the global optimum (or local optima in some cases) quickly. There have been several trials for this purpose such as developing new EC algorithms [1], [2], [6], improving EC operators [3], [4], approximating the fitness landscape for a rough but quick search [3], [4], and others.

In a successful EC search, populations converge to the global optimum even if they are influenced by local optima during the search. Consequently, evolutionary paths, or the moving vectors between generations, provide useful information for estimating the direction of the global optimum. When there is a one-to-one correspondence between a parent individual and its offspring, which is true in the case of evolution strategy, differential evolution, swarm intelligence, and others, we can obtain multiple evolutionary paths or moving vectors as shown in the Fig. 1. The point aimed at by these multiple moving vectors in the d-dimensional search space can be a powerful elite information to estimate the global optimum from the analogy of our research on approximating a fitness landscape [7], [8], [10].

In this paper, we do not address EC in which n parent populations generate n offspring populations in the next generation, such as genetic algorithms and genetic programming, but only EC in which there there exists a one-to-one correspondence between one parent individual and one offspring individual, as mentioned in the above. We use the term of *moving vector* in this paper because the term, *evolutionary path*, implies a trace across several generations, whereas we consider only the line segment between two subsequent generations as shown in Fig.s 1 and 2. Using information derived from evolutionary paths will be the subject of our work in the future.

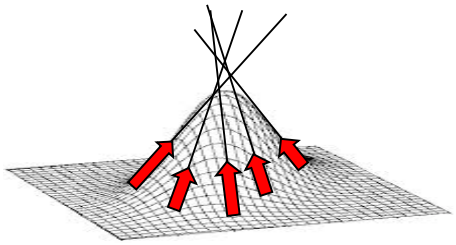


Fig. 1. Moving vectors between individuals in the previous generation and the next generation aims the global optimum area.

The first objective of this paper is to propose methods for estimating the point implied by the the multiple moving vectors between parent individuals and offspring individuals aim, i.e. the $\star$ coordinate in the Fig. 2. The second objective is to show that this estimated point can be a powerful elite individual for optimization.

We describe methods for estimating the $\star$ and approximation methods that do not use matrix operations in section II. Then, we show that the $\star$ point can be a powerful elite individual by showing the fitness order in the section III.

II. Convergence Point of Moving Vectors Between Generations

A. Definition of symbols

When we search a d-dimensional space using one of the evolutionary computation algorithms described in the previous section with n individuals, let the i-th parent individual, its offspring individual, and moving vector be $\boldsymbol{a}_i$, $\boldsymbol{c}_i$, $\boldsymbol{b}_i = \boldsymbol{c}_i - \boldsymbol{a}_i$, respectively; $\{(\boldsymbol{a}_i, \boldsymbol{c}_i),\ i = 1, 2, ..., n;\ \boldsymbol{a}_i, \boldsymbol{c}_i \in \mathbb{R}^d\}$. (See Fig. 2.) The unit directional vector of $\boldsymbol{b}_i$ is also defined as $\boldsymbol{b}_{0i} = \boldsymbol{b}_i / \|\boldsymbol{b}_i\|$ $(\boldsymbol{b}_{0i}^{\mathrm{T}} \boldsymbol{b}_{0i} = 1)$. In this paper, when we refer to a vector we mean a column vector.

Given n vectors, $\boldsymbol{b}_i$, that extend from n vectors, $\boldsymbol{a}_i$, let $\boldsymbol{x} \in \mathbb{R}^d$ be the point that is nearest to the lines made by extending the line segments $\boldsymbol{b}_i$ ($\boldsymbol{x}$ is indicated by the $\star$ mark in the Fig. 2.) The first objective of this paper is to show how to estimate $\boldsymbol{x}$.

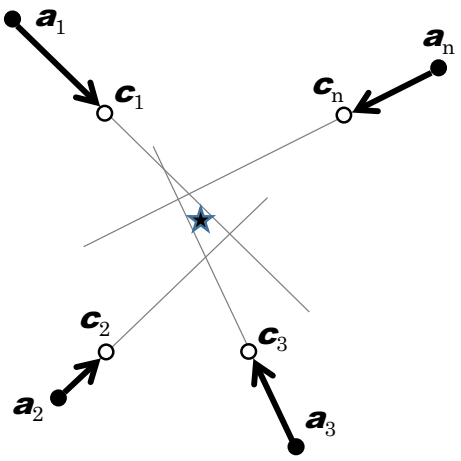


Fig. 2. The convergence point ($\star$) aimed at by the moving vectors between individuals ($\boldsymbol{a}_i,\ i = 1, 2, ..., n$) in the d-dimensional search space in the k-th generation and those ($\boldsymbol{c}_i,\ i = 1, 2, ..., n$) in the $(k+1)$-th generation.

B. Estimation method 1: Exact solution using matrix operations

Lines made by directed line segments mentioned in the previous section are shown in the Eq.(1).

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

The distance between the target point, $\boldsymbol{x}$, and the lines is given by the Eq.(2).

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

We can obtain the solution by solving Eq.(3) consisting of the minimization with regards to $\boldsymbol{x}$ and the line parameters, $\{t_i,\ i = 1, \ldots, n\}$.

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

If we let $\boldsymbol{x}$ be fixed, t_i determines the nearest point on the line i fromv$\boldsymbol{x}$. Since this is an orthogonal projection from $\boldsymbol{x}$ to the line i, we can obtain the below orthogonal condition:

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

From this equation, Eq.(5) is obtained.

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

Note that it is written using ($\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{y}$.)

Let Eq.(5) be put into the $\|\cdot\|$ part of the Eq. (3) to obtain the below Eq.6.

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

where I_d is an identity matrix. Here, let us define H_i as:

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

Next, we can obtain the below objective function from Eq.(2) where we have eliminated the term $\{t_i\}$.

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

Our goal is obtained by minimizing $\mathrm{J}(\boldsymbol{x})$ with regard to $\boldsymbol{x}$. Estimation of $\boldsymbol{x}$, i.e. $\hat{\boldsymbol{x}}$, is obtained by partially differentiating each element of $\boldsymbol{x}$ and setting them equal 0.

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

Thus, the estimation of Eq.(10) is obtained.

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

Since H_i has the characteristic of $H_i^{\mathrm{T}} H_i = H_i^2 = H_i$, i.e. a projection matrix, the Eq.(10) can be rewritten as the Eq.(11).

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

C. Estimation method 2: Approximated solution without matrix operations

Matrix operations are required to solve the Eq. (11) in the Estimation method 1. Although we expect that moving vectors aim toward the global optimum, in practice they do not aim towards the exact point of the global optimum. Based on this fact, one idea would be to quickly calculate $\hat{\boldsymbol{x}}$ and use it for the optimization, even if it is rough estimation.

Here, let's expand Eq.(11) as a Neumann series and approximate it by using only the lower order terms. Eq.(11) is expanded as shown in Eq.(12).

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

When only the 0th order term of this series expansion, i.e. the I_d part, is used, this equation becomes Eq. (13), which does not require the calculation of an inverse matrix, which is thus expected to shorten calculation time. We can increase the precision by increasing the order of the terms used in the calculations according to computer performance.

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

$\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}$ is a matrix and needs two-dimensional memory size. However, it can be calculated using vector sized memory by changing its form as follows.

$$\begin{aligned}\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{a}_i &= \boldsymbol{b}_{0i}(\boldsymbol{b}_{0i}^{\mathrm{T}}\boldsymbol{a}_i) \\ &= \boldsymbol{b}_{0i}(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{b}_{0i}) \ \ (() \text{ is a scalar}) \\ &= (\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{b}_{0i})\boldsymbol{b}_{0i} \end{aligned} \tag{14}$$

By inserting this into Eq.(13), we obtain Eq.(15), which does not use matrix sized of memory.

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

Note that the estimation value is not location equivalent because $\frac{1}{n}\sum_{i=1}\boldsymbol{b}_{0i}\boldsymbol{b}_{0i}^{\mathrm{T}} \neq 0$ in general in Eq.(13).

D. Estimation method 3: Sequential calculation without matrix operations

Let us consider a method for alternately minimizing $\boldsymbol{x}$ and $\{t_i\}$ in Eq.(3). As the Neumann series expansion is performed at the convergence point of this method, we can say that this method finds an approximated solution by expanding the Neumann series in the section II-C successively.

When a point $\boldsymbol{x}$ is given, let $\boldsymbol{y}_i(\boldsymbol{x})$ be the nearest point on the i-th line to $\boldsymbol{x}$, i.e. the projection of $\boldsymbol{x}$ on the i-th line.

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

where $t_i(\boldsymbol{x})$ is given in Eq.(17) from Eq.(5).

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

On the other hand, when $\boldsymbol{y}_i(\boldsymbol{x}),\ (i = 1, 2, ..., n)$ is given, let $\boldsymbol{x}'$ be the point for which the total sum of distances to them is the shortest. Then, $\boldsymbol{x}'$ is the gravity point of $\boldsymbol{y}_i(\boldsymbol{x})$ and is given by the Eq.(18).

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

According to the above, the following iterative method is obtained:

Iterative Method

Step 1	Initialize $\boldsymbol{x}$. For example, $\boldsymbol{x} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{c}_i$.
Step 2	Obtain $t_i(\boldsymbol{x})$ by the Eq.(17) and then $\boldsymbol{y}_i(\boldsymbol{x})$ by Eq.(16) which is the projection of $\boldsymbol{x}$ onto the i-th line.
Step 3	Replace $\boldsymbol{x}$ with $\boldsymbol{x}'$, which is the gravity point obtained by Eq.(18).
Step 4	Go to Step 2 until convergence is reached.

It is guaranteed that the iterative method in this section converges. By updating Eq.(18), the below equation is clearly established.

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

and due to the update of Eq.(16), the below equation is also established.

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

Since $\mathrm{J}(\boldsymbol{x}') \leq \mathrm{J}(\boldsymbol{x})$ in Eq.(8), convergence of this iteration method is guaranteed.

III. Evaluations of the Effectiveness of the Estimated Convergence Point

$\boldsymbol{x}$, which is the convergence coordinate for the moving vectors calculated in Section II, can be a powerful elite individual. We show its potential using the benchmark functions in Table I. Concretely, firstly, we use differential evolution and calculate the convergence point, $\boldsymbol{x}$, of the moving vectors between generations that are generated when a target vector in the previous generation and that in the current generation are different. Secondly, we calculate fitness values for the $\boldsymbol{x}$ and all individuals, show the rank of the $\boldsymbol{x}$ among them in the Table II, and compare it with ranks of other individuals. The confidence intervals for ($p < 0.05$) are shown in Table III.

TABLE I. CEC2005 benchmark fnctions[9] used in our evaluation experiment.(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	√				$[-100, 100]$	-450
f_2	Schwefel 1.2		√			√		-450
f_3	Elliptic		√	√		√		-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

Experimental conditions are: differential evolution (DE/rand, F=1, CR=1, 40 individuals), runs for 100 generations, and benchmark functions of 2-, 5-, and 10-dimensions. These experiments are conducted using Matlab R2011b running on a PC (Intel(R) Core(T) i7-4500 U CPU@ 1.80GHz 2.39GHz, 4GB RAM) under Windows 8.1 (x84).

$\boldsymbol{x}$ obtained in the estimation method 3 is obtained after 10 times iterations in the Section II-D.

We can use the obtained $\boldsymbol{x}$ for speeding up evolutionary search in several different ways. The typical way is to use it to replace the worst individual [7], [8], [10]. However, we will evaluate the effect of speeding-up evolutionary search in our future work and evaluate the $\boldsymbol{x}$ by focusing on its potential to be a powerful elite candidate in this section.

To evaluate the increased computational costs due to implementation of the proposed estimation methods, we measure the CPU time required for the 14 benchmark functions for the first 100 generations. The ratio of average CPU time with/without implementing the proposed methods in differential evolution is shown in the Table IV.

TABLE IV. RATIO OF INCREASED CPU TIME = (DE + ESTIMATION METHOD)/(DE ONLY) OF 14 BENCHMARK FUNCTIONS (2-D, 5-D, AND 10-D) FOR THE FIRST 100 GENERATIONS.

dimension of functions	estimation method 1	estimation method 2	estimation method 3
2-D	1.09	2.27	2.35
5-D	1.37	1.99	1.81
10-D	1.30	1.88	1.53

The significance level for the confidential intervals is $p < 0.05$ and the distribution of rank data is under a normalization hypothesis, i.e., $\overline{X} \in N(\mu, \sqrt{\frac{\sigma^2}{n}})$, $U_{\alpha/2} = 1.96$, $n = 100$, when $\alpha = 0.05$.

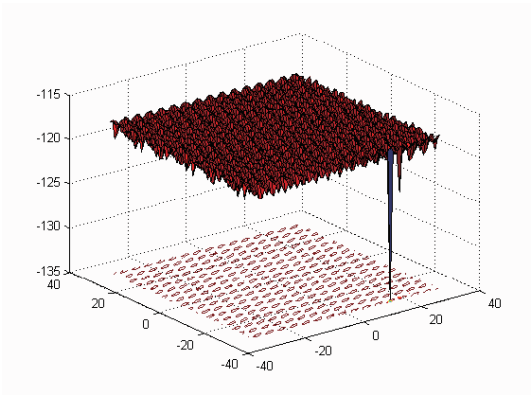
IV. DISCUSSIONS

As expected, $\boldsymbol{x}$ estimated by our proposed method can be a powerful individual for unimodal functions (f_1 – f_5). We can expect to accelerate evolutionary computation searches by using it as an elite individual.

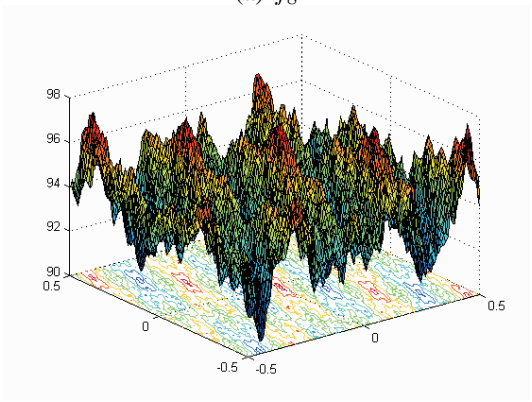
The proposed methods did not work well for 4 of 14 functions: f_8, f_{11}, f_{12}, f_{14}. However, they are not fitness landscapes where an evolutionary search gradually approaches to the global optimum (Fig. 3), so this is natural.

The second discussion point is the approximated calculation. We expanded the estimation method 1 in the Neumann series and approximated it with only its 0-th order term in the evaluation section in this paper. Its performance differed with estimation methods 1 and 3 for lower dimensional unimodal functions (f_1 – f_5). Although the convergence point $\boldsymbol{x}$ is not the global optimum itself in any of estimation methods, approximation with only the 0-th order term must be too rough for optimization. Since the performance of the estimation method 3 with 10 iterations is almost identical to that of estimation 1, that of the estimation method 2 must be increased by calculating some higher order terms. However, taking into account the CPU time comparison, the merit of calculating several higher order terms with estimation method 2 seems little.

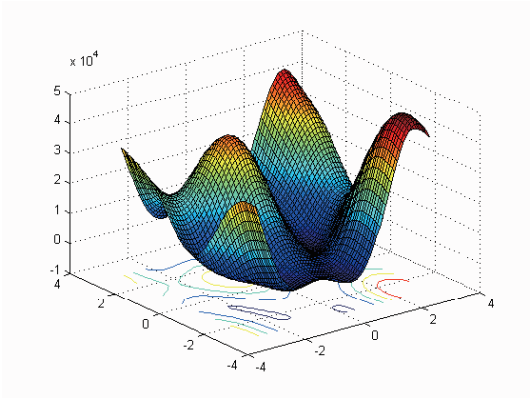
Estimation method 1, which requires the calculation of an inverse matrix and can be thought to be most CPU time demanding among the three methods, was the fastest. As it is said that matrix calculations in Matlab are fast in general, this CPU time is due to Matlab. We have not compared these three



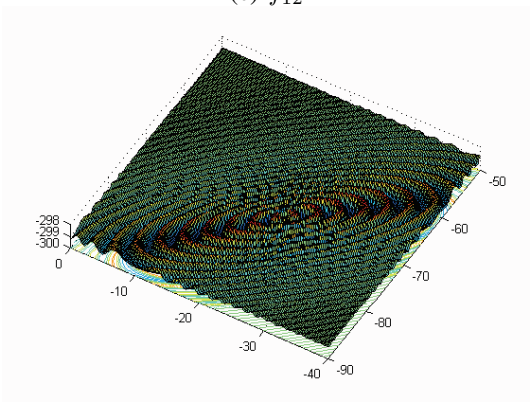
(a) f_8



(b) f_{11}



(c) f_{12}



(d) f_{14}

Fig. 3. Four functions that proposed methods did not work well: f_8, f_{11}, f_{12}, and f_{14}. Figures are reused from the reference [9].

TABLE II. FITNESS RANKS (THE BEST TO THE 41ST) OF $\boldsymbol{x}$ ESTIMATED BY MOVING VECTORS BETWEEN GENERATIONS. AVERAGE RANKS IN THE 2ND–25TH, THE 26TH–50TH, THE 51ST–75TH, AND THE 76TH–100T.

(a) Average rank for 25 generations of the convergence point $\boldsymbol{x}$ obtained by the estimation method 1.

generations	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
functions	Average rank for 2-D functions				Average rank for 5-D functions				Average rank for 10-D functions			
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) Average rank for 25 generations of the convergence point $\boldsymbol{x}$ obtained by the estimation method 2.

generations	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
functions	Average rank for 2-D functions				Average rank for 5-D functions				Average rank for 10-D functions			
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) Average rank for 25 generations of the convergence point $\boldsymbol{x}$ obtained by the estimation method 3.

generations	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100	2 - 25	26 - 50	51 - 75	76 - 100
functions	Average rank for 2-D functions				Average rank for 5-D functions				Average rank for 10-D functions			
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

TABLE III. CONFIDENCE INTERVALS OF EACH ESTIMATION METHOD'S RANK FOR 2-D, 5-D, AND 10-D BENCHMARK FUNCTIONS. WHEN THE LOWER INTERVAL VALUE IS LESS THAN 1 OR THE UPPER ONE IS MORE THAN 41, REPLACE IT WITH 1 AND 41, RESPECTIVELY.

functions	2-D			5-D			10-D		
	method 1	method 2	method 3	method 1	method 2	method 3	method 1	method 2	method 3
f_1	[-2.5, 21.1]	[37.2, 43.1]	[-1.7, 19.7]	[-6.1, 14.2]	[-26.0, 83.0]	[-5.9, 14.0]	[-29.3, 45.3]	[-20.8, 33.5]	[-31.1, 48.3]
f_2	[-4.6, 24.1]	[36.1, 43.7]	[-4.6, 23.5]	[-2.2, 8.8]	[-21.7, 80.7]	[-1.6, 8.0]	[-22.4, 34.6]	[-26.4, 44.7]	[-15.3, 25.7]
f_3	[-11.3, 45.6]	[26.3, 49.1]	[-18.0, 57.8]	[-17.0, 35.2]	[-8.8, 23.6]	[-15.3, 32.9]	[-21.5, 47.1]	[-14.1, 34.8]	[-19.4, 43.8]
f_4	[-0.8, 24.8]	[36.9, 43.3]	[-1.1, 19.7]	[-5.7, 17.7]	[-8.4, 70.9]	[-11.8, 22.6]	[-37.1, 60.9]	[-22.1, 38.7]	[-41.3, 66.8]
f_5	[18.7, 42.3]	[40.0, 41.2]	[23.0, 40.0]	[8.2, 23.9]	[32.0, 45.5]	[8.2, 23.9]	[-15.7, 38.1]	[-19.5, 51.8]	[-10.6, 31.7]
f_6	[10.0, 58.0]	[38.0, 42.0]	[17.8, 52.3]	[-20.5, 50.3]	[-30.1, 82.7]	[-20.4, 49.9]	[-19.2, 32.0]	[-6.2, 14.2]	[-15.7, 27.6]
f_7	[-19.9, 76.1]	[21.8, 54.3]	[-13.1, 71.2]	[-12.2, 26.3]	[-30.3, 77.7]	[-9.7, 23.2]	[-37.2, 57.5]	[-28.7, 44.5]	[-32.6, 52.5]
f_8	[33.7, 45.9]	[34.6, 44.8]	[31.1, 46.6]	[37.0, 43.4]	[31.3, 47.3]	[27.3, 47.5]	[32.0, 46.8]	[37.2, 43.1]	[28.9, 46.2]
f_9	[-26.3, 65.5]	[29.0, 48.9]	[-28.1, 66.9]	[8.7, 62.1]	[12.7, 59.4]	[11.1, 58.8]	[-16.0, 75.7]	[-21.0, 78.1]	[-14.6, 70.6]
f_{10}	[-22.3, 63.4]	[31.2, 46.3]	[-22.3, 63.8]	[-7.5, 72.3]	[-9.7, 71.9]	[-10.2, 69.6]	[-23.9, 76.3]	[-24.5, 74.0]	[-23.3, 73.8]
f_{11}	[30.7, 46.9]	[37.1, 43.3]	[31.3, 47.1]	[30.6, 47.0]	[36.4, 43.7]	[27.2, 47.6]	[30.3, 47.8]	[34.9, 44.3]	[24.0, 50.0]
f_{12}	[36.6, 43.9]	[39.7, 41.5]	[29.4, 48.3]	[33.2, 45.2]	[37.7, 42.5]	[26.8, 47.0]	[39.0, 41.8]	[37.5, 42.7]	[27.9, 46.8]
f_{13}	[-24.3, 74.7]	[28.2, 48.4]	[-21.1, 71.7]	[-26.5, 81.0]	[-20.5, 76.6]	[-24.5, 75.6]	[-43.8, 78.0]	[-32.3, 56.4]	[-37.7, 69.5]
f_{14}	[8.3, 61.8]	[40.3, 41.1]	[-2.3, 67.3]	[28.6, 47.8]	[14.9, 56.5]	[17.6, 52.7]	[21.0, 53.3]	[26.0, 50.6]	[18.9, 52.7]

estimation methods using a non-matrix-base language such as C language, so we do not know if estimation methods 2 and 3 would become faster than estimation method 1.

V. Conclusion and Future Works

We proposed three methods for calculating the convergence point of moving vectors of individuals between generations. We could estimate the convergence point near the global optimum for unimodal functions such that whole individuals would converge to the global optimum, which shows the possibility that the estimated convergence point can be used to accelerate EC search. There are several ways to use the estimated point to accelerate EC search, and we will evaluate them in our future works.

When fitness landscape of EC tasks is multimodal, individuals converge to not only the global optimum but also local optima and the mentioned moving vectors would thus surely aim in multiple directions. If we can categorize the moving vectors based on the convergence points, this clustering is useful for not only accelerating EC search but also developing a new niche method. This is one of major research topics in our future works.

References

[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).

[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] Takagi, H., Ingu, T. and Ohnishi, K., "Accelerating a GA convergence by fitting a single-peak function," J. of Japan Society for Fuzzy Theory and Intelligent Informatics, vol.15, no.2, pp.219–229 (2003) (in Japanese).