Genetic Algorithms with Automatic Accelerated Termination

Abdel-Rahman Hedar, Bun Theang Ong, and Masao Fukushima

Abstract

The standard versions of Evolutionary Algorithms (EAs) have two main drawbacks: unlearned termination criteria and slow convergence. Although several attempts have been made to modify the original versions of Evolutionary Algorithms (EAs), only very few of them have considered the issue of their termination criteria. In general, EAs are not learned with automatic termination criteria, and they cannot decide when or where they can terminate. On the other hand, there are several successful modifications of EAs to overcome their slow convergence. One of the most effective modifications is Memetic Algorithms. In this paper, we modify genetic algorithm (GA), as an example of EAs, with new termination criteria and acceleration elements. The proposed method is called GA with Automatic Accelerated Termination (G3AT). In the G3AT method, *Gene Matrix* (*GM*) is constructed to equip the search process with a self-check to judge how much exploration has been done. Moreover, a special mutation operation called "*Mutagenesis*" is defined to achieve more efficient and faster exploration and exploitation processes. The computational experiments show the efficiency of the G3AT method, especially the proposed termination criteria.

Keywords—Genetic Algorithms, Evolutionary Algorithms Termination Criteria, Mutagenesis, Global optimization,

I. Introduction

Evolutionary Algorithms (EAs) constitute one of the main tools in the computational intelligence area [4], [17]. Developing practical versions of EAs is highly needed to confront the rapid growth of many applications in science, engineering and economics [1], [17]. Nowadays, there is a great interest in improving the evolutionary operators which go further than modifying their genetics to use the estimation of distribution algorithms in generating the offspring [20], [23]. However, EAs still have no automatic termination criteria. Actually, EAs cannot decide when or where they can terminate and usually a user should prespecify a maximum number of generations or function evaluations as termination criteria. The following termination criteria have been widely used in numerical experiments of the recent EA literature.

- Maximum number of generations or function evaluations [18], [20], [22], [28], [29], [31], [32], [34].
- Reaching very close to the known global minima [29], [30], [33], [9].
- Maximum number of consecutive generations without improvement [21].

It is worthwhile to note that the test suites for the methods used in these references are unconstrained test problems with known minima. The above-mentioned measures for termination are generally not applicable in many real-world problems. There are only a few recent works on termination criteria for EAs [5], [15]. In [5], an empirical study is conducted to detect a maximum number of generations using the problem characteristics. In [15], 8 termination criteria have been studied with an interesting idea of using clustering techniques to examine the distribution of individuals in the search space at a given generation. In general, the termination criteria for EAs can be classified as follows:

Abdel-Rahman Hedar is with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, JAPAN (email: hedar@amp.i.kyoto-u.ac.jp). He is also affiliated with the Department of Computer Science, Faculty of Computer and Information Sciences, Assiut University, EGYPT (email: hedar@aun.edu.eg).

Masao Fukushima is with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, JAPAN (phone: +81-75-753-5519; fax: +81-75-753-4756; email: fuku@amp.i.kyoto-u.ac.jp).

This work has been done while Bun Theang Ong was with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, JAPAN.

TABLE I
EAs TERMINATION CRITERIA

Criterion	Measures	Possible Disadvantages
T_{Fit}	Fitness Function Convergence	Premature Convergence
T_{Pop}	Population Convergence	High Costs
T_{Bud}	Max No. of Computational Budget	User Defined
T_{SFB}	Search Feedback	Complexity

- T_{Fit} *Criterion* uses convergence measures of best fitness function values over generations.
- T_{Pop} *Criterion* uses convergence measures of the population over generations.
- T_{Bud} *Criterion* uses maximum numbers of computational budget like the number of generations or function evaluations.
- T_{SFB} *Criterion* uses search feedback measures to check the progress of exploration and exploitation processes.

These termination criteria may face some difficulties as summarized in Table I. EAs can easily be trapped in local minima if only T_{Fit} is applied for termination, especially if the algorithm could fast reach a deep local minimum. Using T_{Pop} for termination is generally improper since making the whole population or a part of it converging is not cheap. In addition, this is not generally needed, and reaching global minima with one individual is enough. Invoking T_{Bud} is problem-dependent and a user should have prior information about the test problem. Finally, although using T_{SFB} seems efficient, it may face the curse of dimensionality. Moreover, saving and checking huge historical search information face a complexity problem.

An intelligent search method should perform wide exploration with deep exploitation. Moreover, it should also have its strategies to check the progress of its main processes; exploration and exploitation. This can equip the search method with automatic termination criteria. This paper introduces a new automatic T_{SFB} termination criterion with low complexity structure. Specifically, a new version of Genetic Algorithms (GAs) called Genetic Algorithm with Automatic Accelerated Termination (G3AT) is proposed. In the G3AT method, the standard GA is equipped with automatic termination criteria and new accelerating elements in order to achieve a better performance of GAs.

Our goal is to construct an intelligent method which seeks optimal or near-optimal solutions of the nonconvex optimization problem:

$$\min_{x \in X} f(x), \tag{$\mathfrak{P}$}$$

where f is a real-valued function defined on the search space $X \subseteq R^n$ with variables $x \in X$. Several versions of EAs have been proposed to deal with this problem, see [8], [10], [20] and reference therein. This problem has also been considered by other heuristics than EAs, see [11], [12] and references therein. Many applications in different areas of computer science, engineering, and economics can be expressed or reformulated as Problem ($\mathfrak{P}$) [1], [6].

The G3AT method is composed by modifying GAs with some directing strategies. First, an exploration and exploitation scheme is invoked to equip the search process with automatic accelerated termination criteria. Specifically, a matrix called *Gene Matrix* (*GM*) is constructed to represent sub-ranges of the possible values of each variable. The role of *GM* is to assist the exploration process in two different ways. First, *GM* can provide the search with new diverse solutions by applying a new type of mutation called "*mutagenesis*". Mutagenesis operator alters some survival individuals in order to accelerate the exploration and exploitation processes. In addition, *GM* is used to let G3AT know how far the exploration process has gone in order to judge a termination point. The mutagenesis operation lets G3AT behave like a so-called "Memetic Algorithm" [26] in order to achieve faster convergence [28], [29]. The numerical results presented later show that mutagenesis is also effective and much cheaper than local search. Then, the final intensification process can be started in order to refine the elite solutions obtained so far. The numerical results shown later indicate that the proposed G3AT method is competitive with some other

versions of GAs. In the next section, we highlight the main components of the G3AT method. In Section III, we discuss numerical experiments with the G3AT method. Finally, the conclusion makes up Section IV.

II. Genetic Algorithm with Automatic Accelerated Termination

In this section, a new modified version of GAs called Genetic Algorithm with Automatic Accelerated Termination (G3AT) is presented. Before presenting the details of G3AT, we highlight some remarks on GAs to motivate the proposed G3AT.

The standard GA selection mechanism is a strict Darwinian selection in which all parents are discarded unless they are good. In general, this is not an optimal selection mechanism since some promising solutions may be lost if all such parents are discarded. On the other hand, there is an elite selection mechanism in which the best individuals out of the parents and offspring can only survive to the next generation. We think this is also not an optimal selection mechanism since it may lead to immature convergence and the diversity may be lost. Therefore, we compose a new operator called "*mutagenesis*" in order to alter some survival individuals which can maintain the diversity and the elitism, and accelerate the diversification and intensification processes.

Termination criteria are the main drawback EAs in general. They are not equipped with automatic termination criteria. Actually, EAs may obtain an optimal or near-optimal solution in an early stage of the search but they are not learned enough to judge whether they can terminate. To counter this weakness, G3AT is built with termination measures which can check the progress of the exploration process through successive generations. Then, an exploitation process starts to refine the best candidates obtained so far in order to achieve faster convergence.

In designing G3AT, we consider the above-mentioned remarks. The standard GA is modified with the below-mentioned components to compose the G3AT method.

A. Gene Matrix and Termination

Each individual x in the search space consists of n variables or genes since G3AT uses the real-coding representation of individuals. The range of each gene is divided into m sub-ranges in order to check the diversity of gene values. The *Gene Matrix GM* is initialized to be the $n \times m$ zero matrix in which each entry of the i-th row refers to a sub-range of the i-th gene. While the search is processing, the entries of *GM* are updated if new values for genes are generated within the corresponding sub-ranges. After having a *GM* full, i.e., with no zero entry, the search is learned that an advanced exploration process has been achieved. Therefore, *GM* is used to equip the search process with practical termination criteria. Moreover, *GM* assists in providing the search with diverse solutions as we will show shortly. We consider two types of *GM* as follows.

1) Simple Gene Matrix (GM^S*): GM*^{S} does not take into account the number of genes lying inside each sub-range. Indeed, during the search, G3AT looks for the value of the considered gene and extracts the number associated with the sub-range, say $v \in \{1, \ldots, m\}$, where this value lies. Let x_i be the representation of the i-th gene, $i = 1, \ldots, n$. Once a gene gets a value corresponding to a non-explored sub-range, the *GM* is updated by flipping the zero into one in the corresponding (i, v) entry.

Figure 1 shows an example of *GM*^{S} in two dimensions, i.e., $n = 2$. In this figure, the range of each gene is divided into ten sub-ranges. We can see that for the first gene x_1, no individual has been generated inside the sub-ranges 1, 7 and 10. Consequently, *GM*^{S}'s values in the $(1, 1)$, $(1, 7)$ and $(1, 10)$ entries are still equal to zero. For the second gene x_2, only the first and the last sub-ranges are still unvisited, hence *GM*^{S}'s values in entries $(2, 1)$ and $(2, 10)$ are null.

2) Advanced Gene Matrix (GM^A_α*): GM*$^A_\alpha$ comes along with a ratio α predefined by the user. Unlike *GM*^{S}, the *GM*$^A_\alpha$ is not immediately updated unless the ratio of the number of individuals that have been generated inside a sub-range so far and m (the total number of gene sub-ranges) exceeds or equals α. Let us note that the counter keeping tracks of the visited sub-ranges is not reinitialized at every iteration.

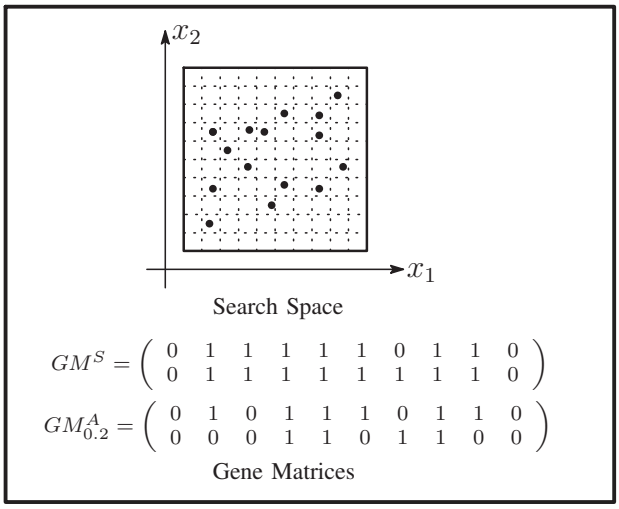

Fig. 1. An example of the *Gene Matrix* in R^2.

An example of GM^A_α with $\alpha = 0.2$ in two dimensions can be found in Figure 1. Like GM^S, no individual has been generated inside the sub-ranges 1, 7 and 10 for the gene x_1. However, unlike GM^S entry $(1,3)$ is equal to 0 in $GM^A_{0.2}$ since there is only one individual lying inside the third sub-range and 1 divided by 10 (the number of sub-ranges) is less than α. For the same reason, x_2 has six zero-entries corresponding to six sub-ranges in which the number of generated individuals divided by m is less than α. This example refers to the first generation of individuals. In a succeeding generation, if one or more individuals are generated inside the third sub-range for x_1 for example, then entry $(1,3)$ will be set equal to one.

B. Artificial Improvement

After computing all children in each generation, G3AT may give a chance to some characteristic children to improve themselves by modifying their genes. This is done by using a more artificial mutation operation called "*mutagenesis*". Specifically, two different types of mutagenesis operation are defined; the *gene matrix mutagenesis* (GM-*Mutagenesis*) and the *best child inspiration mutagenesis* (BCI-*Mutagenesis*).

1) GM-Mutagenesis: In order to accelerate the exploration process, GM-*Mutagenesis* is used to alter N_1 from the N_w worst individuals selected for the next generation. Specifically, a zero-position in *GM* is randomly chosen, say the position (i,j), i.e., the variable x_i has not yet taken any value in the j-th partition of its range. Then a random value for x_i is chosen within this partition to alter one of the chosen individuals for mutagenesis. The formal procedure for GM-*Mutagenesis* is given as follows.

Procedure 2.1: GM$-$*Mutagenesis*(x, GM)

1. If *GM* is full, then return; otherwise, go to Step 2.
2. Choose a zero-position (i,j) in *GM* randomly.
3. Update x by setting $x_i = l_i + (j-r)\frac{u_i - l_i}{m}$, where r is a random number from $(0,1)$, and l_i, u_i are the lower and upper bounds of the variable x_i, respectively.
4. Update *GM* and return.

2) BCI-Mutagenesis: N_2 individuals from the N_w worst children are modified by considering the best child's gene values in the children pool. For each of the N_2 worst children, one gene from the best child is randomly chosen and copied to the same position of the considered bad child as stated formally in the following procedure.

Procedure 2.2: BCI$-$*Mutagenesis*(x, x^{Best})

1. Choose a random gene position i from $\{1, 2, \ldots, n\}$.
2. Alter x by setting $x_i := x_i^{Best}$, and return.

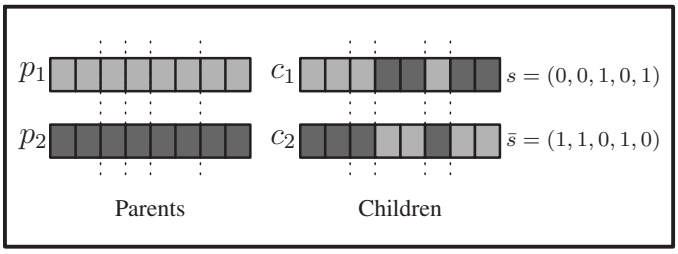

Fig. 2. An example of the crossover operation.

C. Parent Selection

The parent selection mechanism first produces an intermediate population, say P' from the initial population P: $P' \subseteq P$ as in the canonical GA. For each generation, P' has the same size as P but an individual can be present in P' more than once. The individuals in P are ranked with their fitness function values based on the *linear ranking selection mechanism* [2], [10]. Indeed, individuals in P' are copies of individuals in P depending on their fitness ranking: the higher fitness an individual has, the more the probability that it will be copied is. This process is repeated until P' is full while an already chosen individual is not removed from P.

D. Crossover

The crossover operation has an exploration tendency, and therefore it is not applied to all parents. First, for each individual in the intermediate population P', the crossover operation chooses a random number from the interval $(0, 1)$. If the chosen number is less than the crossover probability $p_c \in (0, 1)$, the individual is added to the parent pool. After that, two parents from the parent pool are randomly selected and mated to produce two children c_1 and c_2, which are then placed in the children pool. These procedures are repeated until all parents are mated. A recombined child is calculated to have ρ ($\rho \leq n$) partitions. Both parents genotypes are cut in ρ partitions, and the randomly chosen parts are exchanged to create two children. Let us note that a parent is selected only once, and if the total number of parents inside the parent pool is uneven, then the unfortunate last parent added into the pool is not considered for the mating procedure. As an example, Figure 2 shows two parents p_1 and p_2 partitioned into five partitions at same positions (after the second, third, fourth and sixth genes). Then, a recombined child c_1 is generated to inherit partitions from the two parents according to the random sequence of zeros and ones $(0, 0, 1, 0, 1)$ meaning that its first, second and fourth partitions will be inherited from p_1 and third and fifth partitions from p_2. The other recombined child c_2's partition sequence is the complementary of c_1's sequence, namely $(1, 1, 0, 1, 0)$. The following procedure describes the G3AT crossover operation precisely.

Procedure 2.3: Crossover(p_1, p_2, c_1, c_2)

1. Choose an integer ρ ($2 \leq \rho \leq n$) randomly.
2. Partition each parent into ρ partitions at the same positions, i.e. $p_1 = [X_1^0\ X_2^0\ \ldots\ X_\rho^0]$ and $p_2 = [X_1^1\ X_2^1\ \ldots\ X_\rho^1]$.
3. Choose randomly a sequence of 0 and 1, and let $s = \{o_1, o_2, \ldots, o_\rho\}$, where $o_i \in \{0, 1\}$, $i = 1, \ldots, \rho$.
4. Calculate $\bar{s} = \{\bar{o_1}, \bar{o_2}, \ldots, \bar{o_\rho}\}$, where $\bar{o_i}$ is the complementary of o_i for each i.
5. Calculate the recombined children $c_1 = [X_1^{o_1}\ X_2^{o_2}\ \ldots\ X_\rho^{o_\rho}]$ and $c_2 = [X_1^{\bar{o}_1}\ X_2^{\bar{o}_2}\ \ldots\ X_\rho^{\bar{o}_\rho}]$ and return.

Actually, this type of crossover is chosen to support the G3AT exploration process. Specifically, there is no information related to different sub-ranges of different variables saved in *GM* in order to escape from the complexity of high dimensional problems. Therefore, there is a possibility of having misguided termination of exploration process as in the example shown in Figure 3(a), where the *GM* is already

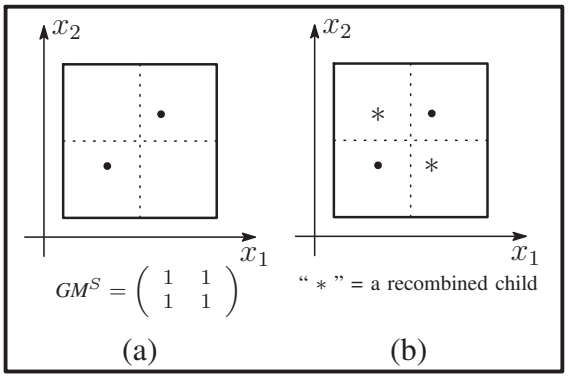

Fig. 3. The role of crossover operation and *GM*.

full although genes x_1 and x_2 have not their search spaces entirely covered. However, invoking such a crossover operation as in Procedure 2.3 can easily overcome such a drawback as shown in Figure 3(b). Moreover, the numerical simulations presented in Section III show that the G3AT can explore the search space widely.

E. Mutation

The mutation operator uses information contained in the *Gene Matrix*. For each individual in the intermediate population P' and for each gene, a random number from the interval $(0, 1)$ is associated. If the associated number is less than the mutation probability p_m, then the individual is copied to the intermediate pool IP_M. The number of times the associated numbers are less than the mutation probability p_m is counted, and let num_m denote this number. Afterward the mutation operation makes sure that the total number num_m of genes to be mutated does not exceed the number of zeros in the *GM*, denoted num_{zeros}. Otherwise the number of genes to be mutated is reduced to num_{zeros}. Finally, a zero from *GM* is randomly selected, say in the position (i, j), and a randomly chosen individual from IP_M has its gene x_i modified by a new value lying inside the j-th partition of its range. The formal procedure for mutation is analogous to Procedure 2.1.

Procedure 2.4: Mutation(x, GM)

1. If *GM* is full, then return; otherwise, go to Step 2.
2. Choose a zero-position (i, j) in *GM* randomly.
3. Update x by setting $x_i = l_i + (j - r)\frac{u_i - l_i}{m}$, where r is a random number from $(0, 1)$, and l_i, u_i are the lower and upper bounds of the variable x_i, respectively.
4. Update *GM* and return.

F. G3AT Formal Algorithm

The formal detailed description of G3AT is given in the following algorithm.

Algorithm 2.5: G3AT Algorithm

1. Initialization. Set values of m, μ, η, $It_{\max}$, and (l_i, u_i), for $i = 1, \ldots, n$. Set the crossover and mutation probabilities $p_c \in (0, 1)$ and $p_m \in (0, 1)$, respectively. Set the generation counter $t := 1$. Initialize *GM* as the $n \times m$ zero matrix, and generate an initial population P_0 of size μ.

2. Parent Selection. Evaluate the fitness function F of all individuals in P_t. Select an intermediate population P'_t from the current population P_t.

3. Crossover. Associate a random number from $(0, 1)$ with each individual in P'_t and add this individual to the parent pool SP_t if the associated number is less than p_c. Repeat the following Steps 3.1 and 3.2 until all parents in SP_t are mated:

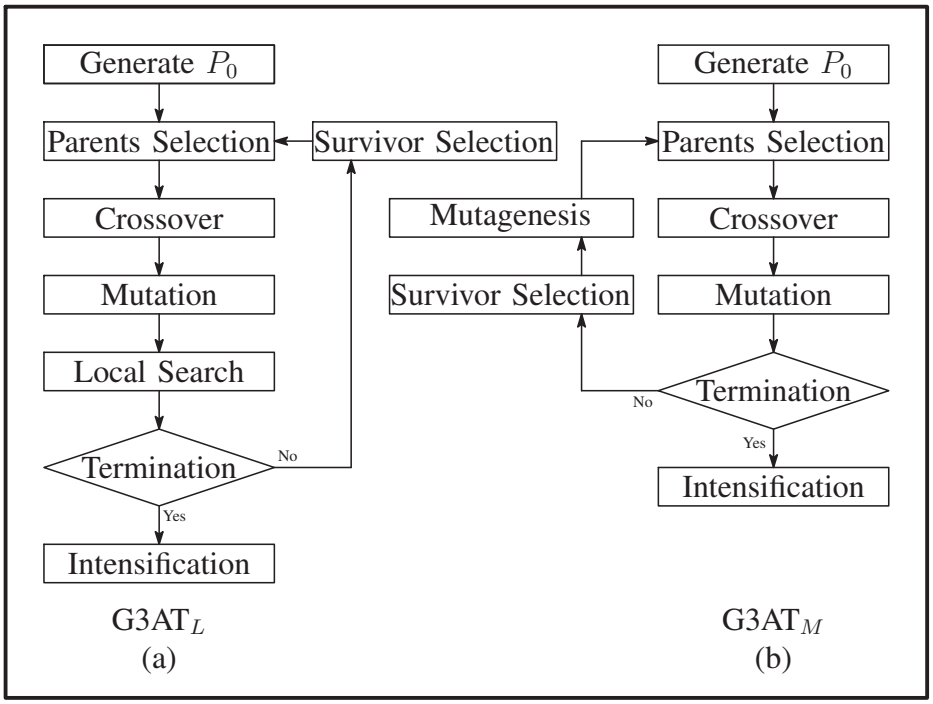

Fig. 4. G3AT Flowcharts.

3.1. Choose two parents p_1 and p_2 from SP_t. Mate p_1 and p_2 using Procedure 2.3 to reproduce children c_1 and c_2.

3.2. Update the children pool SC_t by $SC_t := SC_t \cup \{c_1, c_2\}$ and update SP_t by $SP_t := SP_t \setminus \{p_1, p_2\}$.

4. Mutation. Associate a random number from $(0, 1)$ with each gene in each individual in P'_t. Count the number num_m of genes whose associated number is less than p_m and the number num_{zeros} of zero elements in *GM*. If $num_m \geq num_{zeros}$ then set $num_m := num_{zeros}$. Mutate num_m individuals among those which have an associated number less than p_m by applying Procedure 2.4 and add the mutated individual to the children pool SC_t.

5. Artificial Improvement *(Local Search)*. Apply a local search method to improve the best child (if exists).

6. Stopping Condition. If η generations have passed after getting a full *GM*, then go to Step 9. Otherwise, go to Step 7.

7. Survivor Selection. Evaluate the fitness function of all generated children SC_t, and choose the μ best individuals in $P_t \cup SC_t$ for the next generation P_{t+1}.

8. Artificial Improvement *(Mutagenesis)*. Apply Procedures 2.1 and 2.2 to alter the N_w worst individuals in P_{t+1}, set $t := t + 1$, and go to Step 2.

9. Intensification. Apply a local search method starting from each solution from the N_{elite} elite ones obtained in the previous search stage.

There are two main versions of Algorithm 2.5 as in the flowcharts shown in Figure 4. These versions are composed by considering only one "Artificial Improvement" step (Step 5 or Step 8). G3AT$_L$ version is composed by invoking Local Search as Artificial Improvement and discarding Step 8, while G3AT$_M$ invokes mutagenesis and discards Step 5.

III. Numerical Experiments

Algorithm 2.5 (G3AT) was programmed in MATLAB and applied to 25 well-known test functions [32], [21], listed in the Appendix. These test functions can be classified into two classes; functions involving a

TABLE II
G3AT PARAMETER SETTING

Parameter	Definition	Value
μ	Population size	$\min\{50, 10n\}$
p_c	Crossover Probability	0.6
p_m	Mutation Probability	0.1
m	No. of *GM* columns	$\min\{50n, 200\}$
N_{wm}	No. of individuals used by GM-*Mutagenesis*	2
N_{wb}	No. of individuals used by BCI-*Mutagenesis*	2
N_{elite}	No. of starting points for intensification	1
η_{max}	Value for the parent selection probability computation	1.1
α	Advanced Gene Matrix percentage for $G3AT_M^A$	$3/m$
NM_{maxIt}	No. of maximum iterations used by the Artificial Improvement in G3AT$_L$	$5n$

small number of variables (f_{14}-f_{23}), and functions involving arbitrarily many variables (f_1-f_{13}, f_{24}, f_{25}). For each function, the G3AT MATLAB codes were run 50 times with different initial populations. The results are reported in Tables IV, V, VI, VII and VIII as well as in Figures 5, 6, 7 and 8. Before discussing these results, we summarize the setting of the G3AT parameters and performance.

A. Parameter Setting

First, the search space for Problem $\mathfrak{P}$ is defined as

$$[L, U] = \{x \in R^n : l_i \leq x_i \leq u_i,\ i = 1, \ldots, n\}.$$

In Table II, we summarize all G3AT parameters with their assigned values. These values are based on the common setting in the literature or determined through our preliminary numerical experiments.

We used the scatter search diversification generation method [19], [13] to generate the initial population P_0 of G3AT. In that method, the interval $[l_i, u_i]$ of each variable is divided into four sub-intervals of equal size. New points are generated and added to P_0 as follows.

1) Choose one sub-interval for each variable randomly with a probability inversely proportional to the number of solutions previously chosen in this sub-interval.
2) Choose a random value for each variable that lies in the corresponding selected sub-interval.

We tested three mechanisms for local search in the intensification process based on two methods: Kelley's modification [16] of the Nelder-Mead (NM) method [27], and a MATLAB function called "`fminunc.m`". These mechanisms are

1) *method*$_1$: to apply $10n$ iterations of Kelley-NM method,
2) *method*$_2$: to apply $10n$ iterations of MATLAB `fminunc.m` function, and
3) *method*$_3$: to apply $5n$ iterations of Kelley-NM method, then apply $5n$ iterations of MATLAB `fminunc.m` function.

Five test functions shown in Table III were chosen to test these three mechanisms. One hundred random initial points were generated in the search space of each test function. Then the three local search mechanisms were applied to improve the generated points and the average results are reported in Table III. The last mechanism "*method*$_3$" could improve the initial values significantly, and it is better than the other two mechanisms. Therefore, G3AT employs "*method*$_3$" for the local search in Step 4 of Algorithm 2.5. Moreover, *method*$_3$ was used with a bigger number of iterations as the final intensification method in Step 9 of Algorithm 2.5.

B. G3AT Performance

Extensive numerical experiments were carried out to examine the efficiency of G3AT and to give an idea about how it works. First, preliminary experiments given in Figures 5, 6 and 7 show the role of exploration

TABLE III
RESULTS OF LOCAL SEARCH STRATEGIES

		Initial	Local Search Strategy		
f	n	$\bar{f}$	$\bar{f}_{method_1}$	$\bar{f}_{method_2}$	$\bar{f}_{method_3}$
f_{18}	2	46118	371.3246	23.8734	13.0660
f_{19}	3	–0.9322	–2.6361	–2.8486	–3.6695
f_{15}	4	8.8e+5	1998.5	0.0120	0.0309
f_2	30	2.2e+20	1.2+e19	1.1e+13	340.6061
f_9	30	551.7739	256.0628	91.5412	193.4720

and automatic termination using *GM* within G3AT^S_M version. In these figures, the solid vertical lines "Full GM" refer to the generation number at which a full *GM* is obtained, while the dotted vertical lines "Termination" refer to the generation number at which the exploration process terminates. The G3AT^S_M code continued running after reaching the dotted lines without applying the final intensification in order to check the efficiency of the automatic termination. As we can see, no more significant improvement in terms of function values can be expected after the dotted lines for all 23 test functions. Thus, we can conclude that our automatic termination played a significant role. Even in failure runs for some difficult test problems, genetic operators in G3AT^S_M could not achieve any significant progress after the automatic termination line as shown in Figure 7. On the other hand, letting the final intensification phase in Step 9 of Algorithm 2.5 can accelerate the convergence in the final stage instead of letting the algorithm running for several generations without much significant improvement of the objective value as shown in Figure 8. This figure represents the general performance of G3AT (using G3AT^S_M as an example) on functions f_{12}, f_{15} and f_{23} by plotting the values of the objective function versus the number of function evaluations. We can observe in this figure that the objective value decreases as the number of function evaluations increases. This figure also highlights the efficiency of using a final intensification phase in Step 9 of Algorithm 2.5 in order to accelerate the convergence in the final stage. The behavior in the final intensification phase is represented in Figure 8 by dotted lines, in which a rapid decrease of the function value during the last stage is clearly observed.

Three versions of the G3AT (G3AT$_L$, G3AT^S_M and G3AT^A_M) were compared with each other. The comparison results are reported in Tables IV and V, which contain the mean and the standard deviation (SD) of the best solutions obtained for each function as the measure of solution qualities, and the average number of function evaluations as the measure of solution costs. These results were obtained out of 50 runs. Moreover, t-tests [25] for solution qualities are also reported to show the significant differences between the compared methods.

For the results in Tables IV, we can conclude that solutions obtained by G3AT$_L$ and G3AT^S_M are roughly similar. Nevertheless, we clearly see that the number of function evaluations required by G3AT$_L$ is much larger than that required by G3AT^S_M. We estimate that this relatively high cost in terms of function evaluations required by G3AT$_L$ does not justify the slight advantage in terms of function values. Therefore, we decided to use G3AT^S_M instead of G3AT$_L$ for our further comparisons. At this stage, we may conclude that the mutagenesis operation is as efficient as the local search but the latter is more expensive.

In order to explore sub-ranges of each gene's search space more than once, the parameter α of *GM*$^A_\alpha$ allows G3AT^A_M to revisit sub-ranges until the ratio α is achieved. For our experiments, a sub-range is considered to be visited if it is visited at least three times. We compared these two versions of *GM* methodology and the numerical results are reported in Table V. They indicate that even if G3AT^A_M could be considered more tenacious, the t-tests results show that on the 23 test functions, G3AT^A_M outperformed G3AT^S_M only twice on functions f_{13} and f_{14}. However, the number of function evaluations required by G3AT^A_M is relatively huge compared to that needed by G3AT^S_M for all functions. Consequently, we judge that the best version of G3AT among G3AT$_L$, G3AT^S_M and G3AT^A_M is G3AT^S_M.

TABLE IV

SOLUTION QUALITIES AND COSTS FOR G3AT$_L$ AND G3AT^S_M

		Solution Qualities				t-test for Solution Qualities		Solution Costs	
		G3AT$_L$		G3AT^S_M		t-value	Best Method	G3AT$_L$	G3AT^S_M
f	n	Mean	SD	Mean	SD	G3AT$_L$ – G3AT^S_M	at significance level 0.05	Mean	Mean
f_1	30	0	0	0	0	–	–	3.1e+4	1.4e+4
f_2	30	0	0	0	0	–	–	2.9e+4	1.1e+4
f_3	30	0	0	0	0	–	–	2.7e+4	1.4e+4
f_4	30	0	0	0	0	–	–	2.8e+4	1.2e+4
f_5	30	6.3e–11	2.9e–12	6.4e–11	3.1e–12	1.7	–	3.2e+4	1.4e+4
f_6	30	0	0	0	0	–	–	2.8e+4	1.1e+4
f_7	30	2.0e–4	2.1e–4	2.0e–4	1.6e–4	–	–	3.0e+4	1.2e+4
f_8	30	–1.0e+4	7.8e+2	–1.2e+4	194.04	17.5	G3AT^S_M	2.9e+4	1.3e+4
f_9	30	0	0	0	0	–	–	3.0e+4	1.2e+4
f_{10}	30	8.8e–16	0	8.8e–16	0	–	–	2.9e+4	1.2e+4
f_{11}	30	0	0	0	0	–	–	2.9e+4	1.3e+4
f_{12}	30	3.9e–12	2.9e–12	1.1e–11	7.7e–12	–6.1	G3AT$_L$	3.0e+4	1.3e+4
f_{13}	30	1.36	1.48	1.90	1.37	–1.8	–	3.2e+4	2.1e+4
f_{14}	2	1.37	1.28	2	2.26	–1.7	–	1.6e+3	5.5e+2
f_{15}	4	3.9e–4	2.6e–4	3.3e–4	1.3e–4	1.4	–	3.9e+4	2.1e+3
f_{16}	2	–1.03	8.5e–15	–1.03	3.8e–14	0	–	1.7e+3	5.9e+2
f_{17}	2	0.39	1.5e–13	0.397887	8.8e–14	0	–	1.6e+3	5.9e+2
f_{18}	2	3	7.4e–14	3	2.6e–13	0	–	1.7e+3	6.1e+2
f_{19}	3	–3.86	5.5e–13	–3.86	2.2e–14	0	–	2.6e+3	1.1e+3
f_{20}	6	–3.27	5.7e–2	–3.28	5.5e–2	–0.8	–	4.3e+3	2.5e+3
f_{21}	4	–7.14	3.27	–6.90	3.71	–0.3	–	3.8e+3	2.1e+3
f_{22}	4	–7.64	3.20	–7.86	3.58	0.3	–	3.9e+3	2.1e+3
f_{23}	4	–7.98	3.49	–8.36	3.38	0.5	–	3.8e+3	2.1e+3

TABLE V

SOLUTION QUALITIES AND COSTS FOR G3AT^S_M AND G3AT^A_M

		Solution Qualities				t-test for Solution Qualities		Solution Costs	
		G3AT^S_M		G3AT^A_M		t-value	Best Method	G3AT^S_M	G3AT^A_M
f	n	Mean	SD	Mean	SD	G3AT^S_M – G3AT^A_M	at significance level 0.05	Mean	Mean
f_1	30	0	0	0	0	–	–	1.4e+4	2.6e+4
f_2	30	0	0	0	0	–	–	1.1e+4	2.3e+4
f_3	30	0	0	0	0	–	–	1.4e+4	2.6e+4
f_4	30	0	0	0	0	–	–	1.2e+4	2.4e+4
f_5	30	6.4e–11	3.1e–12	6.5e–11	2.8e–12	–1.6	–	1.4e+4	2.9e+4
f_6	30	0	0	0	0	–	–	1.1e+4	2.3e+4
f_7	30	2.0e–4	1.6e–4	2.1e–4	1.9e–4	-0.3	–	1.2e+4	2.4e+4
f_8	30	–1.2e+4	194.04	–1.2e+4	1.9e–11	0	–	1.3e+4	2.5e+4
f_9	30	0	0	0	0	–	–	1.2e+4	2.4e+4
f_{10}	30	8.8e–16	0	8.8e–16	0	–	–	1.2e+4	2.4e+4
f_{11}	30	0	0	0	0	–	–	1.3e+4	2.5e+4
f_{12}	30	1.1e–11	7.7e–12	2.2e–2	5.2e–2	–2.9	G3AT^S_M	1.3e+4	2.6e+4
f_{13}	30	1.90	1.37	3.2e–1	8.9e–1	6.8	G3AT^A_M	2.1e+4	5.3e+4
f_{14}	2	2	2.26	1.03	1.9e–1	3	G3AT^A_M	5.5e+2	1.4e+3
f_{15}	4	3.3e–4	1.3e–4	3.8e–4	2.4e–4	–1.2	–	2.1e+3	5.2e+3
f_{16}	2	–1.03	3.8e–14	–1.03	9.0e–15	0	–	5.9e+2	1.4e+3
f_{17}	2	0.397887	8.8e–14	0.397887	3.6e–14	0	–	5.9e+2	1.3e+3
f_{18}	2	3	2.6e–13	3	7.4e–14	0	–	6.1e+2	1.4e+3
f_{19}	3	–3.86	2.2e–14	–3.86	3.1e–14	0	–	1.1e+3	2.8e+3
f_{20}	6	–3.28	5.5e–2	–3.27	5.7e–2	0.8	–	2.5e+3	6.3e+3
f_{21}	4	–6.90	3.71	–5.70	3.46	–1.6	–	2.1e+3	5.3e+3
f_{22}	4	–7.86	3.58	–8.23	3.36	0.5	–	2.1e+3	5.3e+3
f_{23}	4	–8.36	3.38	–7.97	3.48	–0.5	–	2.1e+3	5.3e+3

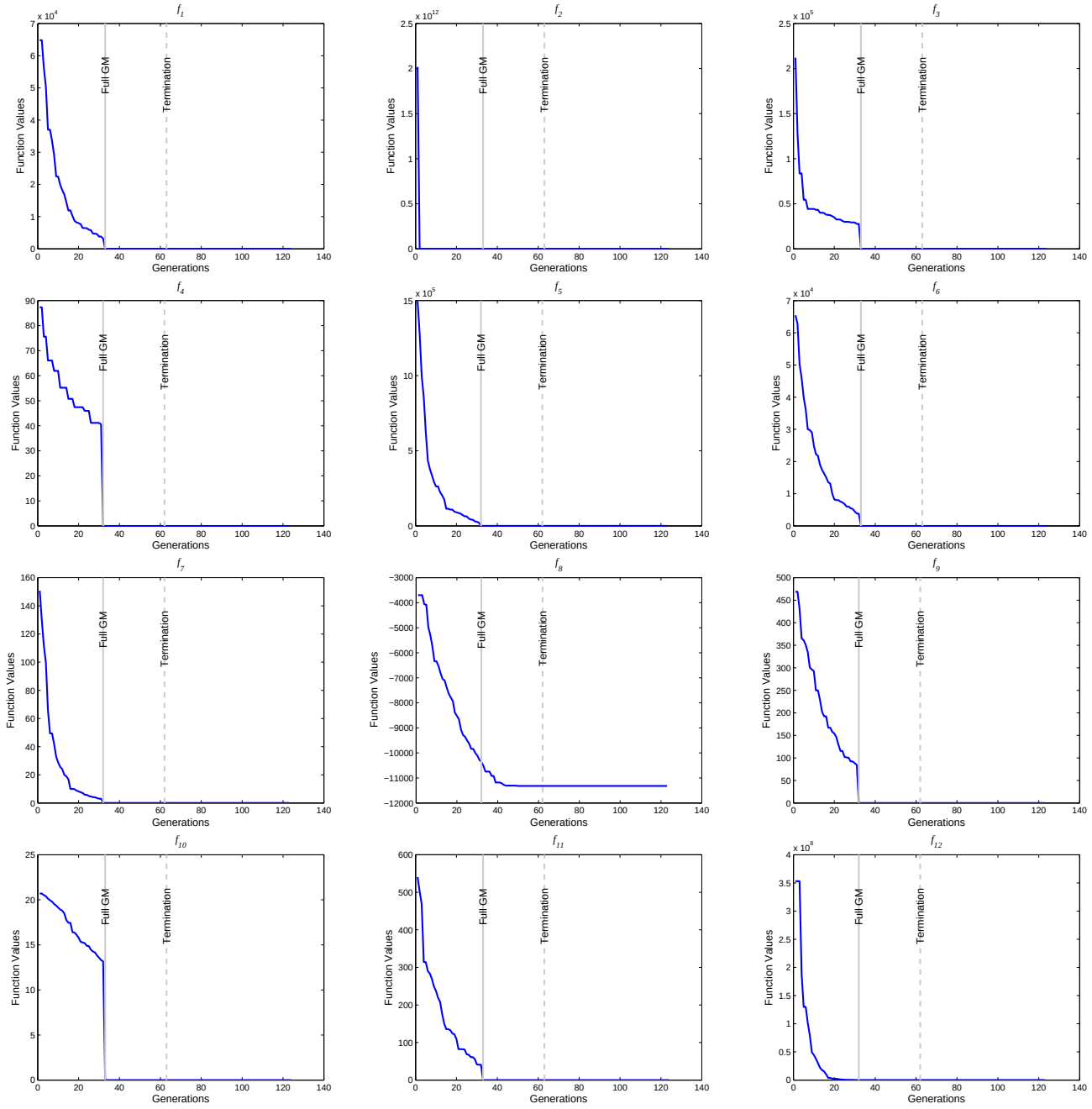

Fig. 5. The Performance of Automatic Termination f_1—f_{12}

C. G3AT Against Other GAs

The best version G3AT^S_M was compared with GA MATLAB toolbox [24] and two other advanced GAs; Orthogonal GA (OGA/Q) [21], and Hybrid Taguchi-Genetic Algorithm (HTGA) [30]. The comparison results are reported in Tables VI, VII and VIII, which contain the mean and the standard deviation (SD) of the best solutions obtained for each function and the average number of function evaluations. These results were obtained out of 50 runs. The results of OGA/Q and HTGA reported in Tables VII and VIII are taken from their original papers [21], [30]. In the tables, t-tests [25] for solution qualities are used to show the significant differences between the compared methods.

The algorithm implemented in GA MATLAB Toolbox is a basic GA. We compared it with our best version of G3AT using the same number of function evaluations and the same GA parameters. To refine

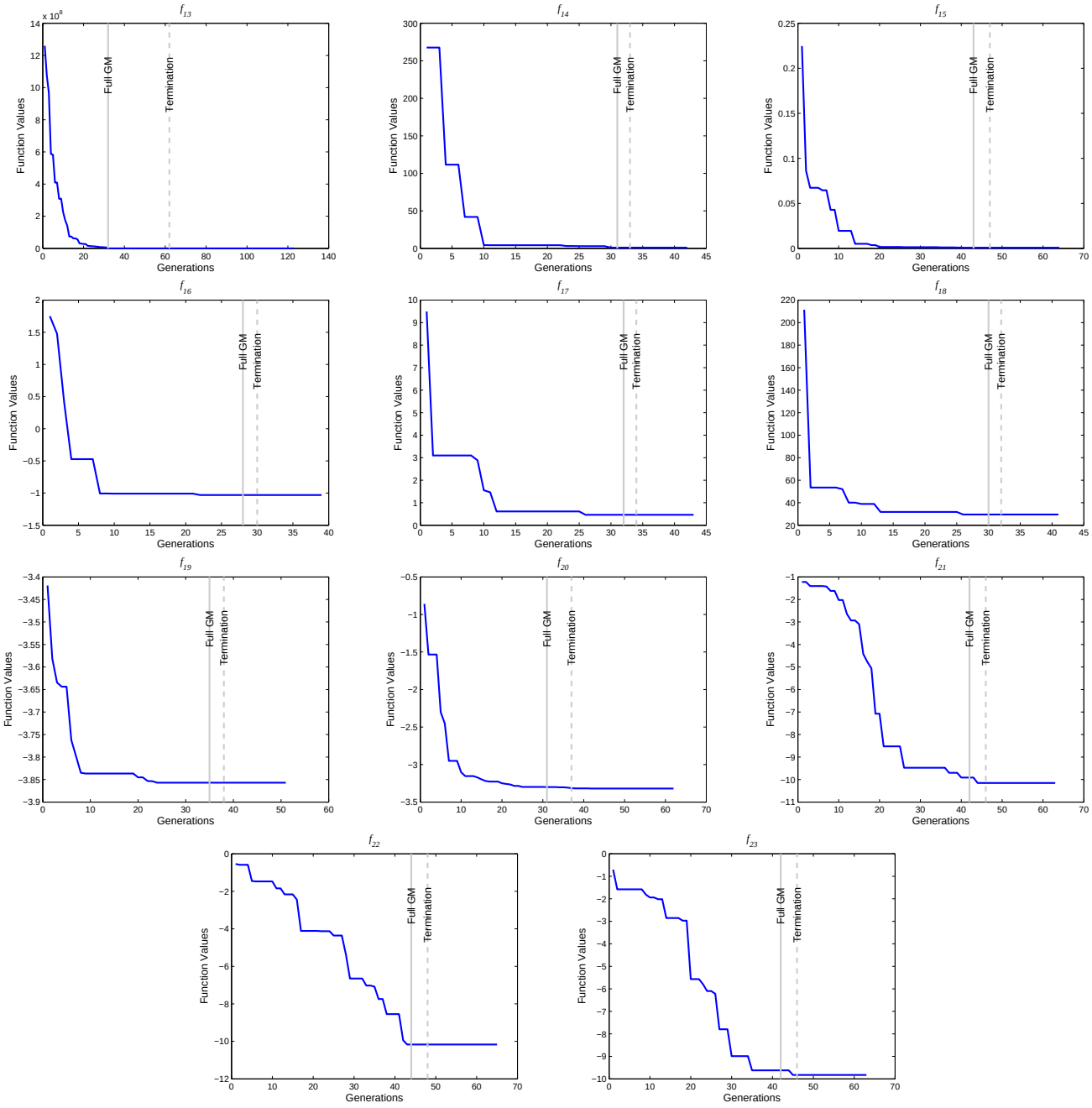

Fig. 6. The Performance of Automatic Termination f_{13}—f_{23}

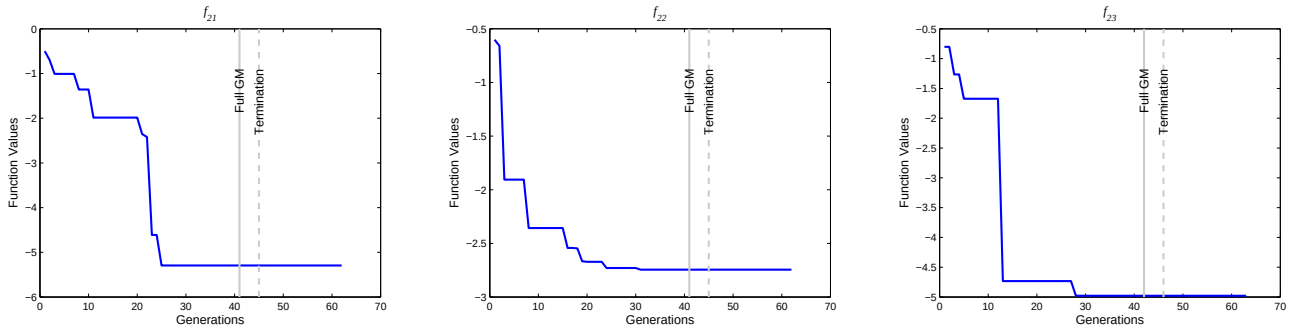

Fig. 7. The Performance of Automatic Termination in Failure Runs

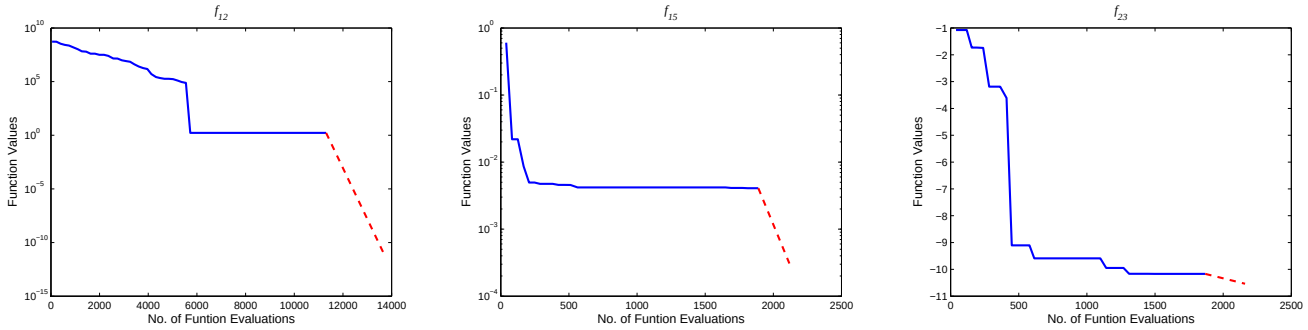

Fig. 8. The Performance of Final Intensification

TABLE VI
SOLUTION QUALITIES FOR G3AT^S_M AND GA

		Solution Qualities				t-test for Solution Qualities	
		G3AT^S_M		GA		t-value	Best Method
f	n	Mean	SD	Mean	SD	G3AT^S_M – GA	at significance level 0.05
f_1	30	0$^{(100\%)}$	0	1.1e–11$^{(100\%)}$	4.9e–11	–1.5	–
f_2	30	0$^{(100\%)}$	0	8.88$^{(0\%)}$	1.78	–35.2	G3AT^S_M
f_3	30	0$^{(100\%)}$	0	1.8e+4$^{(2\%)}$	5.1e+3	–24.9	G3AT^S_M
f_4	30	0$^{(100\%)}$	0	21.62$^{(0\%)}$	11.08	–13.7	G3AT^S_M
f_5	30	6.4e–11$^{(100\%)}$	3.1e–12	1.8e+3$^{(0\%)}$	2.4e+3	–5.3	G3AT^S_M
f_6	30	0$^{(100\%)}$	0	769.34$^{(0\%)}$	225.71	–24.1	G3AT^S_M
f_7	30	2.0e–4$^{(100\%)}$	1.6e–4	0.27$^{(0\%)}$	8.9e–2	–21.4	G3AT^S_M
f_8	30	–1.2e+4$^{(0\%)}$	194.04	–9.3e+3$^{(0\%)}$	7.1e+3	80.6	G3AT^S_M
f_9	30	0$^{(100\%)}$	0	38.90$^{(0\%)}$	10.58	–25.9	G3AT^S_M
f_{10}	30	8.8e–16$^{(100\%)}$	0	5.64$^{(0\%)}$	0.66	–60.4	G3AT^S_M
f_{11}	30	0$^{(100\%)}$	0	1.2e–11$^{(100\%)}$	1.1e–11	–7.7	G3AT^S_M
f_{12}	30	1.1e–11$^{(100\%)}$	7.7e–12	6.55$^{(2\%)}$	2.64	–17.5	G3AT^S_M
f_{13}	30	1.90$^{(4\%)}$	1.37	5.62$^{(2\%)}$	2.23	–10.0	G3AT^S_M
f_{14}	2	2$^{(74\%)}$	2.26	6.04$^{(24\%)}$	5.41	–4.8	G3AT^S_M
f_{15}	4	3.3e–4$^{(100\%)}$	1.3e–4	2.3e–3$^{(42\%)}$	1.8e–3	–7.7	G3AT^S_M
f_{16}	2	–1.03$^{(100\%)}$	3.8e–14	–1.03$^{(100\%)}$	2.3e–14	0	–
f_{17}	2	0.397887$^{(100\%)}$	8.8e–14	0.397887$^{(100\%)}$	3.1e–13	0	–
f_{18}	2	3$^{(100\%)}$	2.6e–13	5.16$^{(92\%)}$	7.40	–2.06	G3AT^S_M
f_{19}	3	–3.86$^{(100\%)}$	2.2e–14	–3.86$^{(100\%)}$	1.5e–14	0	–
f_{20}	6	–3.28$^{(70\%)}$	6.0e–2	–3.26$^{(50\%)}$	6.0e–2	0	–
f_{21}	4	–6.90 $^{(56\%)}$	3.71	–5.19$^{(26\%)}$	3.13	–2.4	G3AT^S_M
f_{22}	4	–7.86$^{(66\%)}$	3.58	–6.13$^{(38\%)}$	3.44	–2.4	G3AT^S_M
f_{23}	4	–8.36$^{(70\%)}$	3.38	–5.99$^{(36\%)}$	3.53	–3.4	G3AT^S_M

the best solution found by GA MATLAB Toolbox, a built-in hybrid Nelder-Mead method is used to keep an equal level of search strategies between GA MATLAB Toolbox and G3AT^S_M. The results are reported in Table VI, which also contains the success rate of a trial judged by means of the condition

$$|f^* - \hat{f}| < \epsilon, \tag{1}$$

where f^* refers to the known exact global minimum value, $\hat{f}$ refers to the best function value obtained by the algorithm, and ϵ is a small positive number, which is set equal to 10^{-3} in our experiments.

Table VI clearly indicates that G3AT^S_M outperforms the GA MATLAB Toolbox in terms of success rates and solution qualities for all functions. This concludes that using the mutagenesis operation can help GA to achieve better results with faster convergence.

As to the results in Tables VII and VIII, all the compared methods (G3AT^S_M, HTGA and OGA/Q) are neutral on 7 out of 14 problems in terms of solution qualities. Moreover, we observe that G3AT^S_M outperforms the other two methods on 4 problems, while HTGA and OGA/Q outperforms G3AT^S_M on 3

TABLE VII
SOLUTION QUALITIES FOR G3AT^S_M, HTGA AND OGA/Q

		Solution Qualities						t-test for Solution Qualities
		G3AT^S_M		HTGA		OGA/Q		Best Method
f	n	Mean	SD	Mean	SD	Mean	SD	at significance level 0.05
f_1	30	0	0	0	0	0	0	–
f_2	30	0	0	0	0	0	0	–
f_3	30	0	0	0	0	0	0	–
f_4	30	0	0	0	0	0	0	–
f_5	100	2.2e–010	0	0.7	0	0.752	0.114	G3AT^S_M
f_7	30	2.0e–4	1.6e–4	1.0e–3	0	6.3e–3	4.1e–4	G3AT^S_M
f_8	30	–12155.34	194.04	–12569.46	0	–12569.45	6.4e–4	HTGA, OGA/Q
f_9	30	0	0	0	0	0	0	–
f_{10}	30	8.8e–16	0	0	0	4.4e–16	4.0e–17	–
f_{11}	30	0	0	0	0	0	0	–
f_{12}	30	1.1e–11	7.7e–12	1.0e–6	0	6.0e–6	1.2e–6	G3AT^S_M
f_{13}	30	1.90	1.37	1.0e–4	0	1.9e–4	2.6e–5	HTGA
f_{24}	100	–93.31	1.14	–92.83	0	–92.83	2.6e–2	G3AT^S_M
f_{25}	100	–72.68	1.46	–78.3	0	–78.3	6.3e–3	HTGA, OGA/Q

TABLE VIII
SOLUTION COSTS FOR G3AT^S_M, HTGA AND OGA/Q

f	n	G3AT^S_M	HTGA	OGA/Q
f_1	30	1.41e+4	1.43e+4	1.13e+5
f_2	30	1.17e+4	1.28e+4	1.13e+5
f_3	30	1.40e+4	1.54e+4	1.13e+5
f_4	30	1.23e+4	1.46e+4	1.13e+5
f_5	100	3.88e+4	3.93e+4	1.68e+5
f_7	30	1.22e+4	1.28e+4	1.13e+5
f_8	30	1.35e+4	1.11e+5	3.02e+5
f_9	30	1.20e+4	1.51e+4	2.25e+5
f_{10}	30	1.20e+4	1.41e+4	1.12e+5
f_{11}	30	1.36e+4	1.61e+4	1.34e+5
f_{12}	30	1.38e+4	3.43e+4	1.35e+5
f_{13}	30	2.05e+4	2.06e+4	1.34e+5
f_{24}	100	4.49e+4	2.19e+5	3.03e+5
f_{25}	100	5.07e+4	2.05e+5	2.46e+5

problems, in terms of solution qualities. On the other hand, G3AT^S_M outperforms the other two methods in terms of solution costs. Actually, G3AT^S_M is cheaper than HTGA and much cheaper than OGA/Q.

IV. CONCLUSIONS

This paper has showed that G3AT is robust and enables us to reach global minima or at least be very close to them in many test problems. The use of *Gene Matrix* effectively assists an algorithm to achieve wide exploration and deep exploitation before stopping the search. This indicates that our main objective to equip evolutionary algorithms with automatic accelerated termination criteria has largely been fulfilled. Moreover, the proposed Artificial Improvement based on the mutagenesis of Gene Matrix and Best Child Inspiration has proved to be more efficient in our experiments than local search based on the Nelder-Mead method. Among the three versions of G3AT, G3AT^S_M is the one showing the best general performance. Finally, the comparison results indicate that G3AT^S_M undoubtedly outperforms the GA MATLAB toolbox, and much cheaper than some advanced versions of GAs.

ACKNOWLEDGMENT

This work was supported in part by the Scientific Research Grant-in-Aid from Japan Society for the Promotion of Science.

APPENDIX

A. Sphere Function (f_1)

Definition: $f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$.
Search space: $-100 \leq x_i \leq 100,\ i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_1(\mathbf{x}^*) = 0.$

B. Schwefel Function (f_2)

Definition: $f_2(\mathbf{x}) = \sum_{i=1}^{n} |x_i| + \Pi_{i=1}^{n} |x_i|$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_2(\mathbf{x}^*) = 0.$

C. Schwefel Function (f_3)

Definition: $f_3(\mathbf{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$.
Search space: $-100 \leq x_i \leq 100,\ i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_3(\mathbf{x}^*) = 0.$

D. Schwefel Function (f_4)

Definition: $f_4(\mathbf{x}) = \max_{i=1,\ldots n}\{|x_i|\}$.
Search space: $-100 \leq x_i \leq 100,\ i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_4(\mathbf{x}^*) = 0.$

E. Rosenbrock Function (f_5)

Definition: $f_5(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right].$
Search space: $-30 \leq x_i \leq 30,\ i = 1, 2, \ldots, n.$
Global minimum: $\mathbf{x}^* = (1, \ldots, 1),\ f_5(\mathbf{x}^*) = 0.$

F. Step Function (f_6)

Definition: $f_6(\mathbf{x}) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$.
Search space: $-100 \leq x_i \leq 100,\ i = 1, 2, \ldots, n.$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0), f_6(\mathbf{x}^*) = 0.$

G. Quartic Function with Noise (f_7)

Definition: $f_7(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$.
Search space: $-1.28 \leq x_i \leq 1.28,\ i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_7(\mathbf{x}^*) = 0.$

H. Schwefel Functions (f_8)

Definition: $f_8(\mathbf{x}) = -\sum_{i=1}^{n} \left(x_i \sin \sqrt{|x_i|}\right).$
Search space: $-500 \leq x_i \leq 500,\ i = 1, 2, \ldots, n.$
Global minimum: $\mathbf{x}^* = (420.9687, \ldots, 420.9687),\ f_8(\mathbf{x}^*) = -418.9829n.$

I. Rastrigin Function (f_9)

Definition: $f_9(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right)\right).$
Search space: $-5.12 \leq x_i \leq 5.12,\ i = 1, \ldots, n.$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0), f_9(\mathbf{x}^*) = 0.$

J. Ackley Function (f_{10})

Definition: $f_{10}(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}.$
Search space: $-32 \leq x_i \leq 32,\ i = 1, 2, \ldots, n.$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0); f_{10}(\mathbf{x}^*) = 0.$

K. Griewank Function (f_{11})

Definition: $f_{11}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1.$
Search space: $-600 \leq x_i \leq 600,\ i = 1, \ldots, n.$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0), f_{11}(\mathbf{x}^*) = 0.$

L. Levy Functions (f_{12}, f_{13})

Definition:
$f_{12}(\mathbf{x}) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}\left[(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))\right] + (y_n - 1)^2\}$
$\quad + \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ y_i = 1 + \frac{x_i - 1}{4},\ i = 1, \ldots, n.$
$f_{13}(\mathbf{x}) = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}\left[(x_i - 1)^2(1 + \sin^2(3\pi x_i + 1))\right] + (x_n - 1)^2(1 + \sin^2(2\pi x_n))$
$\quad + \sum_{i=1}^{n} u(x_i, 5, 100, 4),$
$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a; \\ 0, & -a \leq x_i \leq a; \\ k(-x_i - a)^m, & x_i < a. \end{cases}$$
Search space: $-50 \leq x_i \leq 50,\ i = 1, \ldots, n.$
Global minimum: $\mathbf{x}^* = (1, \ldots, 1),\ f_{12}(\mathbf{x}^*) = f_{13}(\mathbf{x}^*) = 0.$

M. Shekel Foxholes Function (f_{14})

Definition: $f_{14}(\mathbf{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - A_{ij})^6}\right]^{-1},$
$$A = \begin{bmatrix} -32 & -16 & 0 & 16 & 33 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{bmatrix}.$$
Search space: $-65.536 \leq x_i \leq 65.536,\ i = 1, 2.$
Global minimum: $\mathbf{x}^* = (-32, -32);\ f_{14}(\mathbf{x}^*) = 0.998.$

N. Kowalik Function (f_{15})

Definition: $f_{15}(\mathbf{x}) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2,$
$a = (0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246),$
$b = (4, 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16})$.
Search space: $-5 \leq x_i \leq 5,\ i = 1, \ldots, 4.$
Global minimum: $\mathbf{x}^* \approx (0.1928, 0.1908, 0.1231, 0.1358),\ f_{15}(\mathbf{x}^*) \approx 0.0003.75.$

O. Hump Function (f_{16})

Definition: $f_{16}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$
Search space: $-5 \leq x_i \leq 5,\ i = 1, 2.$
Global minima: $\mathbf{x}^* = (0.0898, -0.7126),\ (-0.0898, 0.7126);\ f_{16}(\mathbf{x}^*) = 0.$

P. Branin RCOS Function (f_{17})

Definition: $f_{17}(\mathbf{x}) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10.$
Search space: $-5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15.$
Global minima: $\mathbf{x}^* = (-\pi, 12.275),\ (\pi, 2.275),\ (9.42478, 2.475);\ f_{17}(\mathbf{x}^*) = 0.397887.$

Q. Goldstein & Price Function (f_{18})

Definition: $f_{18}(\mathbf{x}) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))$
$* (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)).$
Search space: $-2 \leq x_i \leq 2,\ i = 1, 2.$
Global minimum: $\mathbf{x}^* = (0, -1);\ f_{18}(\mathbf{x}^*) = 3.$

R. Hartmann Function (f_{19})

Definition: $f_{19}(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right],$
$\alpha = [1, 1.2, 3, 3.2]^T,\ A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},\ P = 10^{-4}\begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$
Search space: $0 \leq x_i \leq 1,\ i = 1, 2, 3.$
Global minimum: $\mathbf{x}^* = (0.114614, 0.555649, 0.852547);\ f_{19}(\mathbf{x}^*) = -3.86278.$

S. Hartmann Function (f_{20})

Definition: $f_{20}(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{6} B_{ij}\left(x_j - Q_{ij}\right)^2\right],$
$\alpha = [1, 1.2, 3, 3.2]^T,\ B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$
$Q = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$
Search space: $0 \leq x_i \leq 1,\ i = 1, \ldots, 6.$
Global minimum: $\mathbf{x}^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300);$
$f_{20}(\mathbf{x}^*) = -3.32237.$

T. Shekel Functions (f_{21}, f_{22}, f_{23})

Definition: $f_{21}(\mathbf{x}) = S_{4,5}(\mathbf{x}), f_{22}(\mathbf{x}) = S_{4,7}(\mathbf{x}), f_{23}(\mathbf{x}) = S_{4,10}(\mathbf{x}),$
where $S_{4,m}(\mathbf{x}) = -\sum_{j=1}^{m}\left[\sum_{i=1}^{4}\left(x_i - C_{ij}\right)^2 + \beta_j\right]^{-1},\ m = 5, 7, 10,$
$\beta = \frac{1}{10}[1, 2, 2, 4, 4, 6, 3, 7, 5, 5]^T,\ C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}.$
Search space: $0 \leq x_i \leq 10,\ i = 1, \ldots, 4.$
Global minimum: $\mathbf{x}^* = (4, 4, 4, 4);\ f_{21}(\mathbf{x}^*) = -10.1532,\ f_{22}(\mathbf{x}^*) = -10.4029,\ f_{23}(\mathbf{x}^*) = -10.5364.$

U. Function (f_{24})

Definition: $f_{24}(\mathbf{x}) = -\sum_{i=1}^{n} \sin(x_i) \sin^{20}(\frac{ix_i^2}{\pi})$.
Search space: $0 \leq x_i \leq \pi,\ i = 1, \ldots, n$.
Global minimum: $f_{24}(\mathbf{x}^*) = -99.2784$.

V. Function (f_{25})

Definition: $f_{25}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i)$.
Search space: $-5 \leq x_i \leq 5,\ i = 1, \ldots, n$.
Global minimum: $f_{25}(\mathbf{x}^*) = -78.33236$.

REFERENCES

[1] T. Back, D.B. Fogel and Z. Michalewicz. Handbook of Evolutionary Computation. IOP Publishing Ltd. Bristol, UK, 1997.

[2] J. E. Baker, Adaptive selection methods for genetic algorithms. In: J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, MA, (1985) 101–111.

[3] A.E. Eiben and J.E. Smith. Introduction to Evolutionary Computing. Springer, Berlin, 2003.

[4] A.P. Engelbrecht. Computational Intelligence: An Introduction. John Wiley & Sons, England, 2003.

[5] M.S. Giggs, H.R. Maier, G.C. Dandy and J. B. Nixon. Minimum number of generations required for convergence of genetic algorithms. *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada (2006) 2580–2587.

[6] F. Glover and G.A. Kochenberger (Eds.). Handbook of Metaheursitics. Kluwer Academic Publishers, Boston, MA, 2003.

[7] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[8] N. Hansen. "The CMA evolution strategy: A comparing review," in *Towards a new evolutionary computation,* Edited by J.A. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea (Eds.), Springer-Verlag, Berlin, 2006.

[9] N. Hansen, and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. *Proceedings of Eighth International Conference on Parallel Problem Solving from Nature PPSN VIII*, 82–291, 2004.

[10] A. Hedar, and M. Fukushima. Minimizing multimodal functions by simplex coding genetic algorithm. *Optimization Methods and Software*, 18: 265–282, 2003.

[11] A. Hedar, and M. Fukushima. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optimization Methods and Software* 19:291–308, 2004.

[12] A. Hedar, and M. Fukushima. Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170:329–349, 2006.

[13] A. Hedar, and M. Fukushima. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, 35(4):521–549, 2006.

[14] F. Herrera, M. Lozano and J.L. Verdegay, Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis, Artificial Intelligence Review, 12 (1998) 265-319.

[15] B.J. Jain, H. Pohlheim and J. Wegener. On termination criteria of evolutionary algorithms, in L. Spector, *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers, p. 768, 2001.

[16] C. T. Kelley. Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition, *SIAM J. Optim.*, 10:43–55, 1999.

[17] A. Konar. Computational Intelligence : Principles, Techniques and Applications. Springer-Verlag, Berlin, 2005.

[18] V.K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable popultion size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, February 2006.

[19] M. Laguna and R. Marti. Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers, Boston, 2003.

[20] C.Y. Lee and X. Yao. Evolutionary programming using the mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8:1-13, 2004.

[21] Y.-W. Leung, and Y. Wang. An orthogonal genetic algorithm with quantization for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 5:41-53, 2001.

[22] D. Lim, Y.-S. Ong, Y. Jin and B. Sendhoff. Trusted evolutionary alorithm. *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada (2006) 456–463.

[23] J.A. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea (Eds.). Towards a new evolutionary computation. Springer-Verlag, Berlin, 2006.

[24] MATLAB Genetic Algorithm and Direct Search Toolbox Users Guide, The MathWorks, Inc. http://www.mathworks.com/access/helpdesk/help/toolbox/gads/

[25] D.C. Montgomery, and G.C. Runger. Applied Statistics and Probability for Engineers. Thrid Edition. John Wiley & Sons, Inc, 2003.

[26] P. Moscato. Memetic algorithms: An introduction, in *New Ideas in Optimization* Edited by D. Corne, M. Dorigo and F. Glover. McGraw-Hill, London, UK, (1999).

[27] J.A. Nelder and R. Mead. A simplex method for function minimization, *Comput. J.*, 7:308-313, 1965.

[28] Y.-S. Ong and A. Keane. Meta-Lamarckian learning in memetic alorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, April 2004.

[29] Y.-S. Ong, M.-H. Lim, N. Zhu and K.W. Wong. Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 36(1):141–152, February 2006.

[30] J.-T. Tsai, T.-K. Liu and J.-H. Chou. Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(2):365–377, April 2004.

[31] Z. Tu and Y. Lu. A robust stochastic genetic algorithm (StGA) for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(5):456–470, October 2004.

[32] X. Yao, Y. Liu and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82-102, July 1999.

[33] W. Zhong, J. Liu, M. Xue and L. Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 34(2):1128–1141, April 2004.

[34] Z. Zhou, Y.-S. Ong, P.B. Nair, A.J. Keane and K.Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 37(1):66–76, January 2007.