Annales Univ. Sci. Budapest., Sect. Comp. 37 (2012) 175–194

MAINTAINING GENETIC DIVERSITY IN BACTERIAL EVOLUTIONARY ALGORITHM

Miklós F. Hatwágner and András Horváth
(Győr, Hungary)

Communicated by András Benczúr

(Received January 15, 2012; revised March 5, 2012;
accepted March 10, 2012)

Abstract. The Bacterial Evolutionary Algorithm (BEA) is a relatively new type of evolutionary algorithm and shows the typical phenomena of stochastic optimization methods. Two of these phenomena: premature convergence and low convergence speed near the optimum are often in connection with the low genetic diversity of the population. Variation of genetic diversity in the original BEA and in its three parallel variants have been examined and documented. Several possible ways of increasing the diversity have been also studied. In this paper the authors present an effective method called forced mutation to maintain the genetic diversity of the population and to speed up the convergence. With forced mutation a significant improvement has been achieved in all test cases, but the parameter setting is problem-dependent. Therefore a possible way of adaptive parameter setting for forced mutation is also proposed.

Key words and phrases: Optimization, heuristic methods, Bacterial Evolutionary Algorithm, gene transfer, parallel computation, genetic diversity.
2010 Mathematics Subject Classification: 68T20, 65K10.
1998 CR Categories and Descriptors: I.2.8, D.1.3, F.1.2, G.4.
The authors' research is supported by the National Development Agency and the European Union within the frame of the project TAMOP 4.2.2-08/1-2008-0021 at the Széchenyi István University entitled "Simulation and Optimization — basic research in numerical mathematics".

1. Introduction

In the late 90's Norberto Eiji Nawa and Takeshi Furuhashi suggested a new algorithm for the optimization of the parameters of fuzzy systems. They labelled their algorithm Bacterial Evolutionary Algorithm (BEA) [12][11], because it imitates the evolution process of bacteria. The algorithm itself is an advanced version of the Pseudo-Bacterial Genetic Algorithm (PBGA) [13] and the original Genetic Algorithm (GA) [4]. BEA inherited the modified mutation operator of PBGA, and introduced the new gene transfer operator instead of crossover. Gene transfer automatically realizes the function of elitism as well. These are the two main operators of the algorithm, because selection is not needed anymore. Partially due to its straightforward and robust nature has this algorithm become more and more popular among researchers.

BEA is a kind of evolutionary algorithm, and its properties are similar to those of the GA's: it is also a global optimization technique, and provides a near-optimal, approximate solution for the problem. It is useful even if the objective function is non-linear, non-continuous, multimodal or high-dimensional. BEA does not use the derivatives of the objective function, thus it does not cause a problem, if they are not known or do not exist.

These properties make it possible to use the BEA in engineering applications as well. One of the problems is that complex optimization problems such as computational fluid dynamics (CFD) or finite element models (FEM) need huge computational power. In these cases, the evaluation of the objective function is a complete simulation process, and can take 0.1 to 5 hours on an average CPU core. To solve a typical industrial problem, thousands of objective function evaluations are needed. That is why parallel execution is so important: this way the optimization time could be a fraction of the sequential optimization time. Unfortunately, only the bacterial mutation operator of the BEA can be executed in a parallel way. Gene transfer in its original form is inherently sequential. The authors already proposed [9] three new, slightly modified gene transfer versions to make the whole algorithm parallel. The properties of these operators were also investigated. The authors used five well-known test functions (K. A. De Jong's 1^{st} and 3^{rd}, Step, Rastrigin, Keane) with different properties. The empirical results showed that there are no significant differences between the three modified operators. All of them have good scaling properties, and in most cases they are more efficient than the original gene transfer even with one CPU.

In order to understand the following topics, the authors feel important to briefly summarize the operation of the BEA, including the original version of the gene transfer, and at least one of the modified gene transfers. The exhaustive details of BEA can be read in [12].

BEA works with a collection of possible solutions. These solutions are often called individuals or bacteria. The collection is called a population. The algorithm creates newer and newer versions (i.e. generations) of the population during the optimization using the two main operators: "bacterial mutation" and "gene transfer". This process is repeated until some kind of termination condition is fulfilled. This condition can be a limit on the maximum number of objective function evaluations, or on the maximum number of generations, etc. The result is the best bacterium of the last population.

Bacterial mutation optimizes the bacteria individually. For every bacterium of the population, the operator does the following. First, it creates K copies, called clones of the original bacterium. Then it chooses one from the genes of the bacterium, and it randomly modifies the value of the selected gene in the clones. The next step is the evaluation of the modified bacteria. If one of the clones proved to be better than the original bacterium, the new value of the gene is copied back to the original bacterium and to all the clones. This process is repeated until all the genes of the original bacteria are mutated. At the end of the mutation, all the clones are dropped.

The gene transfer operator combines the genetic data of the bacteria in the hope that it can produce better bacteria. The original version of the gene transfer operator divides the population into two, equally sized parts. The bacteria with better objective function values get into the superior part, the others into the inferior part. Next, the operator repeats T times the following. It chooses one bacterium from the superior part and another bacterium from the inferior part. After that, it copies some randomly selected genes of the superior bacterium into the other. Naturally, after such a modification, the objective function has to be evaluated again. Before the next iteration of the gene transfer, the population has to be sorted again according to the objective function values. If the modification of the inferior bacteria was successful enough, it can migrate into the superior part. (See Fig. 1.)

As it can be seen, gene transfer has a sequential nature: all the bacteria have to be evaluated immediately after the modification, but only one can be evaluated at a time. This property slows down the optimization processes. That is why the authors proposed three modified, parallel gene transfer versions in [9]. Because of the similar behaviour of these gene transfer versions, only one of them, the gene transfer inspired by the Microbial Genetic Algorithm with auxiliary population (pMGA Aux.) will be presented here. (The details of MGA can be found in [7].)

pMGA Aux. (see Fig. 2) uses an auxiliary population, which can contain A pieces of bacteria, but is initially empty. The operator fills it in the following way. It chooses two bacteria of the population in a random manner and creates a new bacterium in the auxiliary population. The genes of the new bacterium come from the bacterium with the worse objective function value, but some of

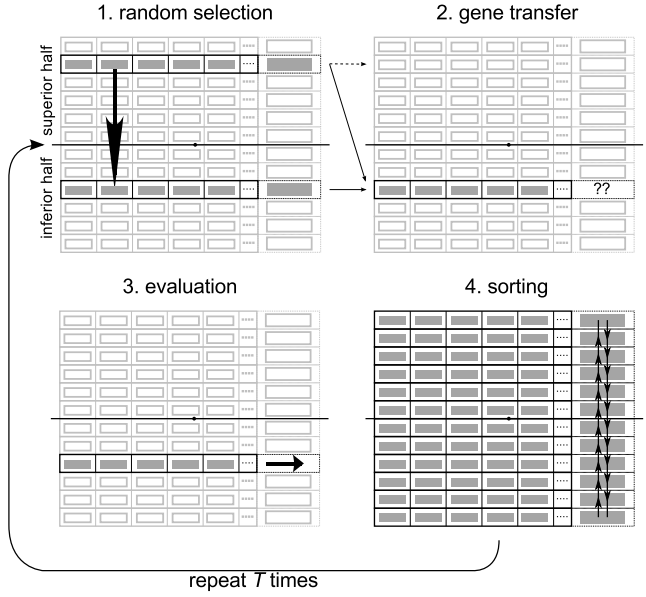


Figure 1: Schematic view of the gene transfer operator

its genes are overwritten with the genes of the other bacterium in some places. (The two selected bacteria remain untouched.) This process has to be repeated until the auxiliary population is full with new bacteria. All of the new bacteria can be evaluated simultaneously. In the next, step the two populations are merged and A number of the least fit bacteria are dropped. If the desired number of gene transfers (T) is more than the size of the auxiliary population, the operator has to be executed again.

It has to be added that the version of the gene transfer operator influences the optimization process. In the case of the original gene transfer, the inferior bacterium is always overwritten, and it does not matter how good or bad it was. pMGA Aux. gives a chance to the inferior bacterium to survive, if the members of the auxiliary populations are not fit enough. This nature of the operator can be regarded as a kind of elitism, but it can decrease the genetic diversity of the population.

Another side-effect of pMGA Aux. is that during the original gene transfer the modified genes of the bacteria have $T-1$ possibilities to infiltrate into other bacteria as well. In the case of using an auxiliary population, this number drops to $(T/A) - 1$. This phenomenon also decreases the genetic diversity, but the possibility of parallel objective function evaluations helps to speed up computations.

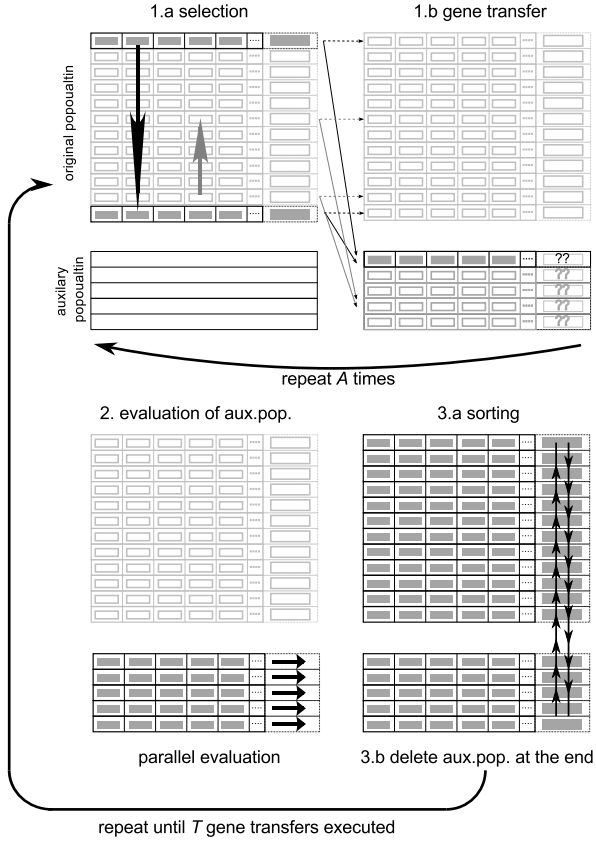


Figure 2: Schematic view of the gene transfer inspired by MGA with auxiliary population (pMGA Aux.)

The authors already measured genetic diversity in [8]. The measurements showed that the genetic diversity is practically not lower in the case of using the modified gene transfers, but its rate is very low in general. This phenomenon slows down the convergence speed and makes it harder to find the multiple optima of multimodal functions. It is obvious that some kind of diversity increasing technique can be very useful in practice. Several algorithms were developed for genetic algorithms, but only a few of them can be applied to BEA. Another way of speeding up optimization is the use of a local search algorithm included in BEA to find local optima faster and more accurately. Unfortunately, these hybrid or memetic algorithms [1] have some drawbacks as well. The local optimizer often has requirements regarding the objective function (e.g. continuity, needs gradients, etc.), and can be computationally complex.

In the next sections, the genetic diversity during the optimization will be further analysed and some techniques made to increase genetic diversity will be discussed. The authors will examine their applicability in the case of BEA, and present the results of test function optimizations. Finally, a new method will be introduced that is similar to the Directed Random Search (DRS) methods. A possible way of auto-tuning the parameters of this method will also be presented.

2. Measuring genetic diversity

The authors used four well-known artificial test functions to study the effect of gene transfers on genetic diversity. The test functions were selected on the basis of computational complexity and several other properties. Table 1 summarizes their most important properties. Equations 2.1-2.4 show the equations of the test functions.

	Linearity	Modality	Continuity	Plateaux / Shoulder
De Jong's 1st (sphere)	no	unimodal	yes	no
Ackley	no	multimodal	yes	yes
Rastrigin	no	multimodal	yes	no
Keane	no	multimodal	no	no

Table 1: Properties of the test functions

$$f_{DeJong1} = \sum_{i=1}^{n} x_i^2, \quad -5.12 \leq x_i \leq 5.12 \tag{2.1}$$

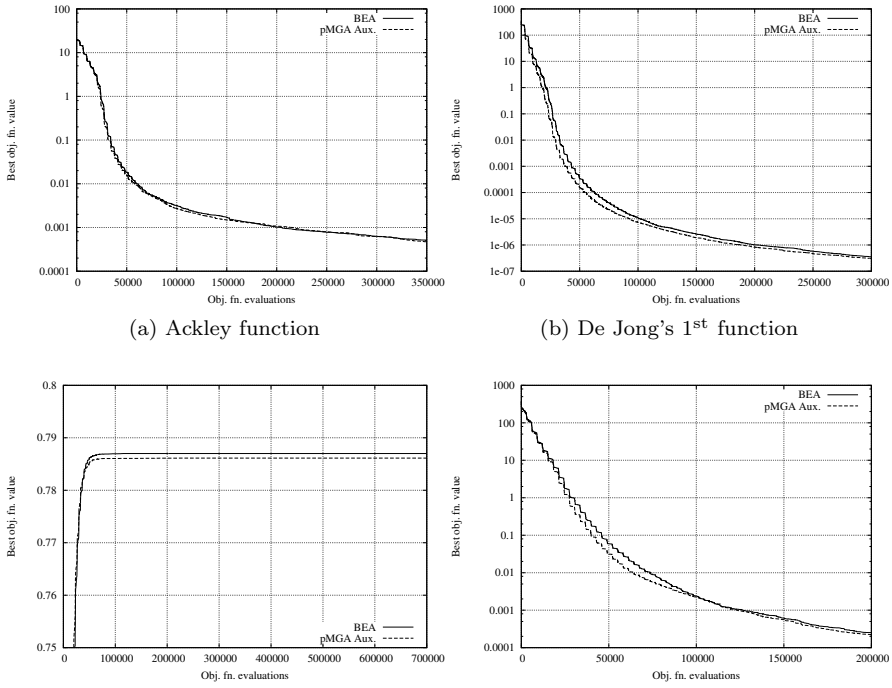
$$f_{Ackley} = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e \tag{2.2}$$
$$-20 \leq x_i \leq 30$$

$$f_{Rastrigin} = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right), \quad -5.12 \leq x_i \leq 5.12 \tag{2.3}$$

$$f_{Keane} = \left|\frac{\left(\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)\right)}{\sqrt{\sum_{i=1}^{n} i x_i}}\right| \tag{2.4}$$
$$0 \leq x_i \leq 10, \quad \prod_{i=1}^{n} x_i \geq 0.75, \quad \sum_{i=1}^{n} x_i \leq 7.5n$$

Fig. 3 shows the value of the best bacterium during different test function optimizations as a function of the number of objective function evaluations.

This approach is very practical in the sense that during the optimization process of a real, engineering problem BEA spends most of the wall clock time on the evaluation of the objective function. The results are the average of 20 repeated optimizations. For extent reasons, mostly the results of Rastrigin function optimization will be presented in the following figures.



(a) Ackley function

(b) De Jong's 1st function

(c) Keane function

(d) Rastrigin function

Figure 3: Optimization of test functions

The authors used formula 2.5 to measure the "genetic distance" (d_{ij}) between bacteria i and j. This function was proposed by Goldberg to realise niching with [4]. Here g represents the number of genes, x_{ik} is the k^{th} gene of the bacterium, $X_{k,max}$ and $X_{k,min}$ are the possible maximal and minimal values of the k^{th} gene. Using this formula, the genetic diversity of the whole population can be expressed with formula 2.6. Here P is the size of the population. The meaning of formula 2.6 is very straightforward: the value of it is always between 0 and 1, and it denotes the average relative difference between the values of the genes. Fig. 4 shows the genetic diversity during the optimization of the Rastrigin function. The main parameters of the optimization were:

$P = 128, K = 1, g = 20, T = 512, A = 64$, where K is the number of clones and g is the number of genes.

$$d_{ij} = \sqrt{\frac{\sum_{k=1}^{g} \left(\frac{x_{ik} - x_{jk}}{X_{k,max} - X_{k,min}} \right)^2}{g}} \tag{2.5}$$

$$D = \frac{1}{P-1} \sum_{i=1}^{P} d_{i,best} \tag{2.6}$$

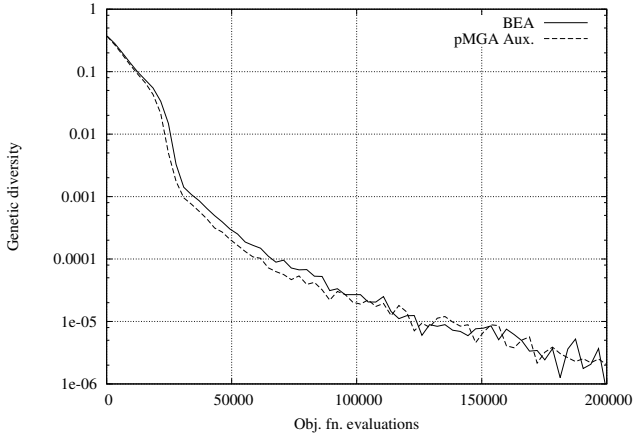


Figure 4: Genetic diversity during the optimization of Rastrigin function

As it can be seen in the figure, both gene transfer operators produce similar but very low genetic diversity especially at the end of the optimization. In such a situation most of the bacteria in the population are almost the same and only the bacterial mutation is able to produce new values. In this case BEA does not perform better than a simple random search and it causes very slow convergence. This phenomenon is not surprising, because none of the above mentioned gene transfer methods have protection against evolving similar or identical bacteria.

3. Classical methods to maintain genetic diversity

It is obvious that some kind of technique would be useful to maintain the originally existing genetic diversity of the population. This technique would be

especially useful to find multiple solutions to multimodal problems, but it can also be used to do multi-objective optimization or simulate complex, adaptive systems. Even if the objective function has multiple optima, the standard evolutionary algorithms focus on finding a single solution, because the individuals gather around a single solution in the course of consecutive generations. Several methods were developed through the years, but most of them were originally developed for GA, and it is not always trivial to adapt these techniques to BEA. The two best known methods are niching and speciation. Both of them are inspired by nature. The details of these techniques can be found for example in [4], [5] and [10], and here only a short description will be given.

In nature, a lot of species exist at the same time and they occupy their own ecological niche. Like in nature, niching makes it possible for the GA to maintain a population with diverse solutions through time. It means that multiple optimal solutions can be obtained in a single population. The two main methods to attain niching are fitness sharing and crowding.

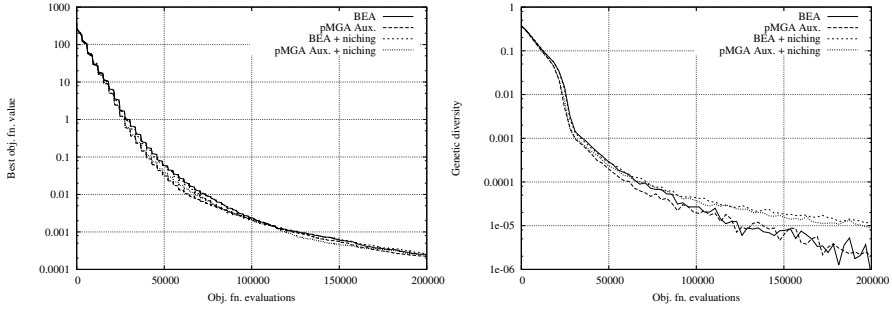
The idea of fitness sharing is that the resources in nature are limited, and the individuals living in the same niche have to share these resources. This necessity inhibits the uncontrolled growth of species in the population.

In GA, if some individuals are similar (and thus occupy the same niche), their fitness is divided by a value depending on the number of similar individuals. This value is what we call niche count, which is the sum of the sharing function values. The sharing function is a function of the genetic distance between two individuals and its value depends on the degree of similarity between these individuals. The value of the sharing function is 1 if the individuals are identical and 0 if the distance is above a given threshold (σ). Otherwise it returns a value in the [0; 1] range depending on the level of similarity. Formula 3.1 shows one of the widely used simple sharing functions. The derated fitness values can be obtained with formula 3.2. Figure 5 depicts the optimization process of the Rastrigin function with ($\sigma = 10^{-6}$) and without niching.

$$s(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma}\right), & \text{if } d < \sigma \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d_{ij})} \tag{3.2}$$

The attentive reader has surely noticed that BEA does not use fitness values because these values are not necessary for the operators. If niching is required for some reason, fitness values have to be recorded also. Naturally, bacteria must be classified in the superior and inferior parts of the population using the



(a) Best objective fn. value (b) Genetic diversity

Figure 5: Optimization of Rastrigin function with and without niching

fitness values, which are derived from the objective function values. Another kind of practical problem arises if the value of the objective function can be negative as well. In this case, formula 3.2 does not work correctly. This problem can be solved for example with a simple linear transformation of the objective function value into the [0; 1] range.

Unfortunately, the use of niching did not increase the genetic diversity in the measure hoped for. Obviously, the goal of niching is not to increase the genetic diversity for its own sake but to maintain an appropriate diversity that helps creating good individuals in the vicinity of equally good solutions. It means that the primary goal of this technique is not to speed up the convergence, so other techniques have to be applied.

The second major realization of niching is crowding. It was first proposed by De Jong [3]. He also tried to maintain the originally existing diversity of the population, but he used another method to achieve it. This method also measures the similarity, but instead of modifying the fitness values the new individuals simply overwrite the most similar individuals in the population.

There are several possible implementations, but only De Jong's original method will be presented here. In this algorithm only a fraction of the population ("generation gap", generally 0.1) reproduces and dies. The members of the generation gap are chosen with fitness-proportional selection. Only these individuals participate in crossover and mutation. Every new individual replaces the most similar one of the "crowding factor" individuals. The amount of crowding factor individuals is generally between 2 and 5 and the individuals are chosen in a random manner.

Deb and Goldberg compared crowding and fitness sharing with some test functions in [2], and fitness sharing performed better than crowding in finding multiple peeks. This is one reason why the authors of this article used fitness

sharing instead of crowding. The other reason is that it is not easy to use the concept of crowding in BEA. In BEA, all of the bacteria participate in the creation of newer bacteria and the "crowding factor" is half of the population. This value is generally much more than 5.

In nature, the individuals of different species do not mate with each other. The method of speciation imitates this law to find multiple solutions of the objective functions. Niching does not prevent an individual of a niche to mate with an individual of a different niche. This kind of mating often results in "lethal solutions", representing none of the two niches. This problem can be solved using speciation methods. Deb and Goldberg reviewed the following speciation method (but other techniques are also known, e.g. tag bits [14]). The two selected parent individuals participate in crossover only if the genetic distance between them is under a specified parameter (σ_{mating}). Otherwise another parent must be chosen in a random manner until the condition is fulfilled. Deb and Goldberg implemented this restriction on mating in the crossover operator, and niching in selection as well. Using both methods the individuals found their way near the peeks. This technique is undoubtedly useful if the goal is to find all the equally good solutions of the problem, but the idea is entirely contrary to the concept of gene transfer. In the case of gene transfer the goal was to transfer some genes from the superior bacterium into an inferior bacterium. In many cases the genetic distance between these bacteria is very large, thus this method is not applicable in BEA.

The applicability of Restricted Random Search (RTS, [6]) was also investigated. This method is similar to crowding, but performs much better and it is simpler than fitness sharing. Unfortunately, its adaptation has the same difficulties as crowding and speciation.

4. The method of "forced mutation"

In the previous section classical methods that could help to maintain genetic diversity have been examined. Niching appeared to be moderately useful, but other methods were not suitable in bacterial optimization problems. In this section a new method that directly prevents the population from the low levels of diversity is presented.

The main idea of this method is that the members of the population that are too similar to an another member with better object function value do not improve convergence, therefore it is better to spread them in a wider area around their original position with a mutation operator. This method will be referred to as "forced mutation".

Forced mutation has no direct analogy in real genetics, however, it is similar in a way to possible conscious breeding tactics where the governing person

intervenes in the process if there are too similar individuals in the population and forces them to change slightly, e.g. with radioactive irradiation. As far the authors know this kind of forced mutation is not applied in plant or animal breeding.

Another biological (but not genetic) analogy can be found in the rivalry for limited resources, for example for water resources in a dry territory. Individuals of an animal population strive towards the water resource and the most successful or fittest ones will occupy the best areas. The "second best" entities try to get close to the resource but the locally strongest animals can drive them out from the immediate surroundings of the best known water resource. This way the good but not best individuals are forced to look for an another water resource in the neighbourhood, and they may find an even better one than the original. Without their being driven out this better source could remain unknown.

Forced mutation shows similarities with directed random search methods, therefore, one could consider this method as a hybrid bacterial (BEA, pMGA Aux) and local search (directed random search) method. In contrast to the most hybrid methods, the proposed technique is well integrated in the sense that the two types of steps work together and the user does not have to take care of switching between bacterial and local search steps. Another useful characteristic of forced mutation is that it does not require the derivatives of the object function and will work even for discontinuous problems, therefore, it can be used for every problem that is manageable by bacterial type optimization.

Forced mutation steps are applied after every full bacterial generation is finished. The algorithm is the following: let $x[i]$ denote the i-th element of the sorted population, i.e. $f(x[i]) \leq f(x[i+1])$ for every $i = 1, 2, ..., N-1$, where f is the objective function to be minimized.

```
Loop for i=2, ..., N
  Loop for j=1, ..., i-1
    distance=d(x[i], x[j])
    if distance<sigma:
      mutate x[i]
      mark x[i] for re-evaluation
      jump out from inner loop
    end if
  End loop (j)
End loop (i)

Loop for i=2, ..., N
  if x[i] is marked for re-evaluation:
     calculate f(x[i])
End loop (i)
```

Here "mutation" means that every gene will be perturbed with a normal distribution random number with $\sigma \cdot \Delta X_k$ deviation, where X_k is the difference between the maximal and minimal value of the k-th gene.

This forced mutation is significantly different from the "classical" mutation operator in the following aspects:

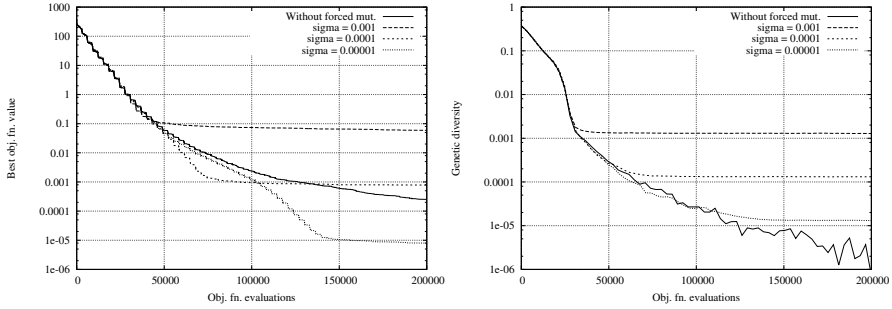
- In a population with large genetic diversity, forced mutation has no effect, while classical mutation always changes a constant number of individuals. On the other hand, forced mutation becomes active when the population begins to degenerate. This kind of adaptivity to the actual features of the population is a desirable property.
- Forced mutation places the modified individuals in the close environment of the original state since the original state is presumably close to a local optimum, therefore, it is beneficial to search in the close regions of the phase space.
- This type of mutation has a "local elitism" property: it preserves the individuals that are the best among similar (in σ radius) individuals, therefore it does not change the most promising ones.

The control parameter of forced mutation is σ, which shows the radius of the undesirable environment of entities. It is obvious that a large σ will protect the calculation process from premature convergence, but it has a negative effect at the end of the optimization, when the refining of the found solutions is happening. On the other hand, a very small σ will help at the refining phase, but will not prevent premature convergence.

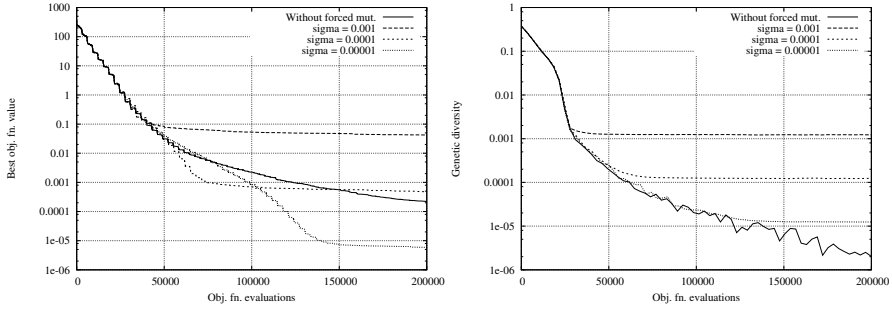
In Fig. 6 calculation results with the Rastrigin test function are presented. According to theoretical expectations, with forced mutation the genetic diversity will always be higher than σ. The convergence to the optimal value (0) is affected by forced mutation as written above:

- at the very beginning of the optimization process forced mutation has only a small effect on the convergence speed
- for a high (10^{-3}) σ value convergence is slowed down near the optimum
- for a medium σ value there is significant boost at the middle of the process, but refining is slowed down
- for small σ (10^{-5}) there is a moderate speed up at the middle, but in the refining phase the best individuals approach the optimal value well.

Other test problems showed the same qualitative behaviour, however, the optimal range for σ is problem-dependent. To demonstrate this, an objective



(a) Best objective fn. value, BEA (b) Genetic diversity, BEA



(c) Best objective fn. value, pMGA Aux. (d) Genetic diversity, pMGA Aux.

Figure 6: Optimization of Rastrigin function with and without forced mutation, using different σ values

function value for every test problem was fixed ("target value") and the number of object function evaluations needed to achieve this target value is presented in Tab. 2 and 3. The target value was 10^{-6} for De Jong's 1$^{\text{st}}$ problem (optimal value is 0), 10^{-3} for the Ackley function and Rastrigin function (optimal value is 0) and 0.79 for the Keane function (maximization problem with 0.80 optimal value).

In the presented calculations a maximal number of objective function evaluations was chosen to prevent a quasi-infinite calculation time. The calculation is considered "successful" if the optimization process can reach the target value within this limit. For every test problem the ratio of the successful cases was logged during the repeated calculations. If this "success rate" is 0, then no value can be presented. When the success rate is less than 100%, the average of the needed number of objective function evaluations to reach the target objective value is shown in successful cases; the success rate is also given in parentheses.

σ	Test functions			
	De Jong's 1st	Ackley	Rastrigin	Keane
10^{-2}	-	-	-	56047
10^{-3}	-	-	-	(95%) 104625
10^{-4}	-	-	98524	141065
10^{-5}	107365	76944	101807	(80%) 276272
10^{-6}	165365	121247	123930	(55%) 167852
without f. m.	202846	204746	127586	(40%) 50970

Table 2: Needed number of objective function evaluations to reach the target objective value with BEA and forced mutation. (Success rates are shown in parentheses.)

σ	Test functions			
	De Jong's 1st	Ackley	Rastrigin	Keane
10^{-2}	-	-	(75%) 808365	62454
10^{-3}	-	-	-	(90%) 72344
10^{-4}	(60%) 395597	-	76639	(85%) 275413
10^{-5}	92029	74021	96716	(45%) 176429
10^{-6}	148673	117743	120703	(35%) 75851
without f. m.	186041	215102	120903	(25%) 41660

Table 3: Needed number of objective function evaluations to reach the target objective value with pMGA Aux. and forced mutation. (Success rates are shown in parentheses.)

Tab. 2 and 3 show the similar behaviour of two bacterial algorithm variants. For large σ the success rate is usually 0 for the first three problems. Keane shows the opposite: large σ helps to find the good solution. This is probably due to the extremely high number of local maximums in this problem and premature convergence is the main obstacle in the way of the optimization process. As a short conclusion one can state that forced mutation with $\sigma = 10^{-5}$ and 10^{-6} always has a positive effect on the optimization process.

Note that due to the limited size of the paper, calculation results with other BEA variants are not presented here quantitatively, however, they show the same behaviour as BEA and pMGA Aux.

5. Adaptive parameter setting in forced mutation

The previous section proved that this type of forced mutation can improve the success rate and convergence speed. The optimal σ parameter is problem-

dependent, which is a serious drawback in practical applications. As it has been written above, higher values of σ are good for preventing premature convergence, but are bad in refinement of the extrema, while smaller σ has no effect in the beginning and does not work against premature convergence, but is useful at the end phase of optimization.

It is obvious that changing σ in the course of optimization can help, but the optimal way to do it can be highly problem-dependent. A simple idea is considered in this paper: let σ be a simple, monotonically increasing function of the actual value of genetic diversity. This way the value scope of forced mutation corresponds to the stage of the optimization process: at the beginning, when potentially optimal members of the population are scattered in a large region of the search space, we get high σ values, which decreases the probability of premature convergence, but after a few iterations the entities will gather in interesting domains, which decreases the diversity, which results in a smaller σ that helps in refinement.

The simplest way of this parameter-setting is a linear relationship between genetic diversity of the whole population (D) and σ with a lower bound:

$$\sigma = \max(bD, \sigma_0). \tag{5.1}$$

Based on the results in the previous section, $\sigma_0 = 10^{-5}$ or $\sigma_0 = 10^{-6}$ can be a good value.

In a typical calculation, genetic diversity starts from 0.1–0.5 (random entities). In this stage $\sigma \approx 0.01$ is a good choice. Genetic diversity starts to decrease rapidly as the entities start to converge to interesting optimum areas in the search space. At this phase much smaller values of σ are needed. Based on these considerations, $b = 0.1$–0.5 seems to be a good choice.

The proposed way of adaptive parameter setting is promising in the way that the actual value of σ will be adopted both to the problem and the stage of the optimisation process. The numerical experiments validate these hopes.

Table 4 and 5 show the numerical results with $b = 0.1$, 0.2 and 0.5. Comparing these values to the ones in Table 2 and 3 one can observe that the adaptive setting of σ works well. In the case of $b = 0.2$ we get a 100% success rate in every test problem, and the needed number of objective function evaluations is always significantly lower than that of the ones in the original methods without forced mutation. Furthermore, with $b = 0.2$, optimal parameter setting is almost always better than the forced mutation with any fixed σ, which shows that this simple method for σ setting is beneficial. However, in further work more sophisticated methods for setting σ should be examined.

Figure 7 shows the best objective function value and genetic diversity during the optimization process for the Rastrigin test problem. These graphs show that adaptive forced mutation works as expected. Other test problems show

the same behaviour, i.e. with $b = 0.2$ the actual objective value of the best individual is always better than that of the original version and the genetic diversity does not decrease to an undesirably low value.

b	Test functions			
	De Jong's 1st	Ackley	Rastrigin	Keane
0.1	104227	76082	96623	(85%) 109154
0.2	104869	74881	91276	58988
0.5	91790	60391	(50%) 134901	241600

Table 4: Needed number of objective function evaluations to reach the target objective value with BEA and adaptive forced mutation. $\sigma_0 = 10^{-5}$.

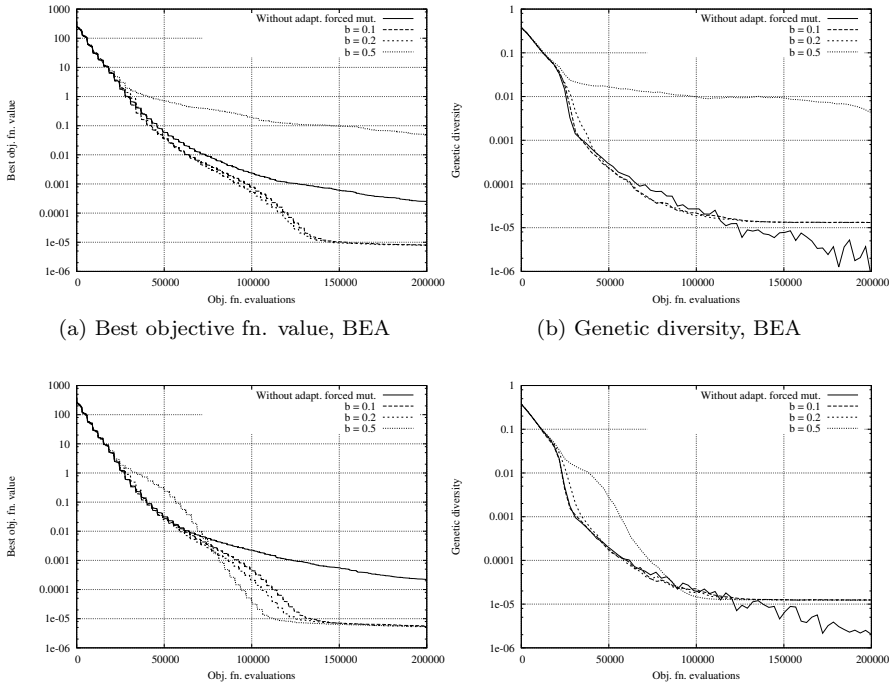
b	Test functions			
	De Jong's 1st	Ackley	Rastrigin	Keane
0.1	90880	73925	90905	(65%) 186437
0.2	90647	69449	85889	53005
0.5	79765	60011	77042	188214

Table 5: Needed number of objective function evaluations to reach the target objective value with pMGA Aux. and adaptive forced mutation. $\sigma_0 = 10^{-5}$.

6. Summary

Genetic diversity in the original BEA and a parallel variant, "pMGA Aux." was investigated. It was shown that genetic diversity decreases very rapidly in the examined problems, which results in a slow convergence rate especially at the final stage of the optimization, where refining of the approximate optimum is happening. Classical methods of maintaining genetic diversity were considered and Goldberg's type "niching" was found to be applicable for bacterial type methods without fundamental conceptual changes. This method was moderately useful in controlling the diversity and improving the speed of optimization.

The method of "forced mutation" was proposed in this paper to avoid the pile-up of too similar entities. It was shown that this method works well, but different parameters are optimal in different problems and in the different stages of the optimization process. To overcome this problem a simple model



(a) Best objective fn. value, BEA

(b) Genetic diversity, BEA

(c) Best objective fn. value, pMGA Aux.

(d) Genetic diversity, pMGA Aux.

Figure 7: Optimization of Rastrigin function with and without adaptive forced mutation, using different values for b

for adaptive setting of forced mutation parameter was investigated. The final version of this method appeared to be useful: it improved the success rate and the speed of the convergence in every test problem.

Adaptive forced mutation is commendable in every bacterial type method since it does not require any special property of the objective function and it improves the convergence speed significantly in all the examined cases.

References

[1] **Botzheim, J., C. Cabrita, L.T. Kóczy and A.E.B. Ruano,** Fuzzy rule extraction by bacterial memetic algorithms, in: *Proceedings of Int. J. Intell. Syst.*, 2009, 312–339.

[2] **Deb, K.and D.E. Goldberg,** An investigation of niche and species formation in genetic function optimization, in: *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, 42–50.

[3] **De Jong, K.A.,** *An Analysis of the Behavior of a Class of Genetic Adaptive Systems* (dissertation), Michigan, USA, (1975)

[4] **Goldberg, D.E.,** *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., USA, 1989.

[5] **Goldberg, D.E. and J. Richardson,** Genetic algorithms with sharing for multimodal function optimization, in: *Proceedings of the Second International Conference on Genetic Algorithms*, 41–49.

[6] **Harik, G.,** Finding multimodal solutions using restricted tournament selection, in: *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1995, 24–31.

[7] **Harvey, I.,** The microbial genetic algorithm, in: G. Kampis et al (Ed.) *Proceedings of the Tenth European Conference on Artificial Life*, Springer LNCS., Heidelberg, 2009.

[8] **Hatwagner, M. and A. Horvath,** Comparative analysis of the parallel gene transfer operators of the bacterial evolutionary algorithm, *Acta Polytechnica Hungarica*, accepted (2012).

[9] **Hatwagner, M. and A. Horvath,** Parallel gene transfer operations for the bacterial evolutionary algorithm, *Acta Technica Jaurinensis*, **4(1)** (2011), 89–113.

[10] **Mahfoud, S.W.,** *Niching Methods for Genetic Algorithms* (dissertation), Urbana, Illinois, USA, 1995.

[11] **Nawa, N.E. and T. Furuhashi,** A study on the effect of transfer of genes for the bacterial evolutionary algorithm, in: Jain, L. C., Jain, R. K. (Ed.) *Second Intenational Conference on Knowledge-Based Intelligent Electronic System*, Adelaide, Australia, 21-23 April 1998, 585–590.

[12] **Nawa, N.E. and T. Furuhashi,** Fuzzy system parameters discovery by bacterial evolutionary algorithm, *IEEE Transactions on Fuzzy Systems*, **7(5)** (1999), 608–616.

[13] **Nawa, N.E., T. Hashiyama, T. Furuhashi and Y. Uchikawa,** A study on fuzzy rules discovery using pseudo-bacterial genetic algorithm with adaptive operator, *Proceedings of IEEE Int. Conf. on Evolutionary Computation*, ICEC'97, 1997.

[14] **Spears, W.M.,** Speciation using tag bits, *Handbook of Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press, 1995.

M.F. Hatwágner and A. Horváth
Széchenyi István University
H-9026 Győr, Egyetem tér 1.
Hungary
{miklos.hatwagner, horvatha}@sze.hu