

Effect of epistasis on the performance of genetic algorithms^{*}

Sajad JAFARI¹, Tomasz KAPITANIAK², Karthikeyan RAJAGOPAL³,

Viet-Thanh PHAM^{†‡4}, Fawaz E. ALSAADI⁵

¹Biomedical Engineering Department, Amirkabir University of Technology, Tehran 15875-4413, Iran

²Division of Dynamics, Lodz University of Technology, 90-924 Lodz, Poland

³Center for Nonlinear Dynamics, Defence University, 6020 Bishoftu, Ethiopia

⁴Modeling Evolutionary Algorithms Simulation and Artificial Intelligence, Faculty of Electrical & Electronics Engineering,
Ton Duc Thang University, Ho Chi Minh City, Vietnam

⁵Department of Information Technology, Faculty of Computing and IT, King Abdulaziz University, Jeddah, Saudi Arabia

[†]E-mail: phamvietthanh@tdt.edu.vn

Received May 28, 2018; Revision accepted Sept. 11, 2018; Crosschecked Oct. 15, 2018; Published online Nov. 10, 2018

Abstract: In the field of genetics, it is well known that a specific genetic behavior may be influenced by more than one gene. There is a similar concept in genetic algorithms (GAs), called epistasis, which is the interaction between genes. This study demonstrates that, in spite of what is generally assumed, GAs are not an efficient optimization tool. This is because the main operator, mating (crossover), cannot function properly in epistatic optimization problems. In non-epistatic problems, although a GA can possibly provide a correct solution, it is an inefficient and time-consuming algorithm. As proof, we used conventional test functions and introduced new ones and confirmed our claim with simulation results.

Key words: Genetic algorithm (GA); Epistasis; Crossover; Superposition; Optimization; Cost function
<https://doi.org/10.1631/jzus.A1800399>

CLC number: O31

1 Introduction

Genetic algorithms (GAs) are an optimization method that have been widely used for many applications, including vehicle routing problems (Karakatić and Podgorelec, 2015), sequence planning (Tseng et al., 2018), train-set circulation plan problems (Zhou et al., 2017), genetic code adaptability (de Oliveira et al., 2018), feature selection (Dong et al., 2018), business process monitoring (di Francescomarino et al., 2018), damage identification (Greco et al., 2018), cryptosystems (Jain and Chaudhari,


2017), and detection systems of public security events (Wang et al., 2017).

Epistasis is a phenomenon where the function of one gene is influenced by one or several other genes. According to William Batson (Steinberg and Cosloy, 2009), in genetics, a gene is epistatic if it influences the operation of other genes. In GA literature, epistasis is used with a similar definition: the interaction between genes (Haupt and Haupt, 2004). Although the interaction between the parameters of a problem is a serious issue, it has been ignored or neglected in many optimization and GA studies. In this paper, it is shown that in optimization problems with epistasis, the GA algorithm faces some critical challenges and if there is no epistasis, the GA is not efficient.

According to GA studies, some of the essential advantages of GAs are as follows (Haupt and Haupt, 2004; Sivanandam and Deepa, 2008):

[‡] Corresponding author

^{*} Project supported by the Polish National Science Centre, MAESTRO Programme (No. 2013/327 08/A/ST8/00/780)

 ORCID: Sajad JAFARI, <https://orcid.org/0000-0002-6845-7539>

© Zhejiang University and Springer-Verlag GmbH Germany, part of Springer Nature 2018

1. It can deal with a large number of variables (it can overcome the curse of dimensionality).

2. It is capable of optimizing functions with extremely complex surfaces because of the ability to escape from local optima. In other words, it has the ability of finding the global optimum of objective functions.

3. It requires no knowledge of the objective function surface or its gradient. It only needs the value of the objective function at some points.

It also has some disadvantages (Haupt and Haupt, 2004; Sivanandam and Deepa, 2008):

1. It requires a large number of fitness function evaluations.

2. It is not effective for smooth unimodal cost functions. Note that this does not mean that a GA cannot find the optimum of these surfaces. It means that considering the time-consuming way in which GAs solve simple problems, it is better to use simple methods such as gradient-based algorithms or line search in these cases.

As mentioned before, one significant advantage of GAs compared to other methods is its capability to overcome the so-called curse of dimensionality. In fact, in the real world, a GA is not being used to solve 1D problems, for example, because according to the first disadvantage, it would be a waste of time and resources. Even problems that are used to educate evolutionary algorithms with the purpose of visualization usually have two or three variables. In every nontrivial optimization problem, there should be at least a minor degree of epistasis; otherwise, the problem will be trivial (Davis et al., 2012). The reason is that, if there is no epistasis, the N -variable non-epistatic function can be divided into N separate functions, each with one variable. A GA is undoubtedly one of the last choices for optimizing such functions. Considering this issue, why there is no investigation on epistasis in almost all of the references that use a GA? Should a GA be accepted as an efficient and useful algorithm just because it has been used in the last four decades? This work attempts to answer these questions.

2 Role of epistasis in benchmarks

In this section, some important benchmarks in the GA field are investigated. There is an example in

(Haupt and Haupt, 2004), named “Word Guess”, in which a game has been designed to examine GAs. The following is a brief explanation of this example: suppose we are going to find a specific 8-letter word using a GA. The number of letters in a word is given to the GA and it starts guessing different combinations of letters which form the word until it finds the right solution. Therefore, every chromosome is an 8-letter array. The fitness of each chromosome is equal to the number of letters which are in the correct place. Thus, the cost function for each solution (chromosome) is defined as follows:

$$\text{cost} = \sum_{n=1}^{\text{\#letters}} [1 - \text{sgn}(\text{guess}_n - \text{answer}_n)], \quad (1)$$

where #letters is the number of letters in the word, guess_n is the n th letter in the guess chromosome, and answer_n is the n th letter in the answer.

In this example, the word being tested is “colorado”. To find the word, a GA with 32 chromosomes was used, and the word was discovered in the 17th iteration. This means that $32 \times 17 = 544$ evaluations of the cost function have been done.

Haupt and Haupt (2004) stated that since the total number of possible combinations is $26^8 \approx 2.08 \times 10^{11}$ (8 places for letters and 26 possible letters for each place), this is a notable accomplishment for a GA and it proves its efficiency (544 compared to 2.08×10^{11}). However, an important missing point is that there is no epistasis between variables to find the unknown word. Therefore, we just need to find the correct letter for each place separately. In this case, the maximum number of required evaluations of cost function is equal to $8 \times (26 - 1) = 200$. Although GAs are usually successful in finding the optimum for such problems, it would be very costly and time-consuming because of the large number of cost function evaluations required. Thus, there is no reason to use a GA for this type of problem.

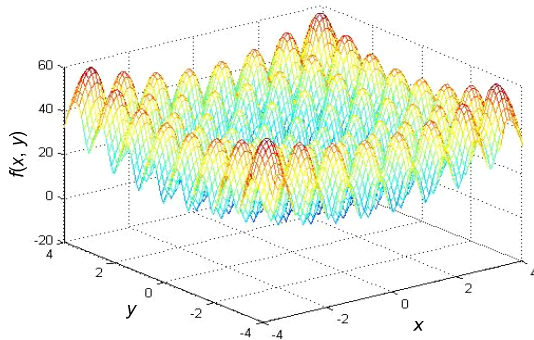
In (Haupt and Haupt, 2004), some test functions have been introduced to examine the efficiency of various optimization algorithms, especially GAs. These functions are among the most famous test functions commonly used in the literature (Guo et al., 2014; Teimouri et al., 2014; Qu et al., 2016). However, an interesting point about these functions is that many of them have zero degree of epistasis (e.g. functions in Table 1, where x and y are optimization variables).

Table 1 Functions with zero degree of epistasis

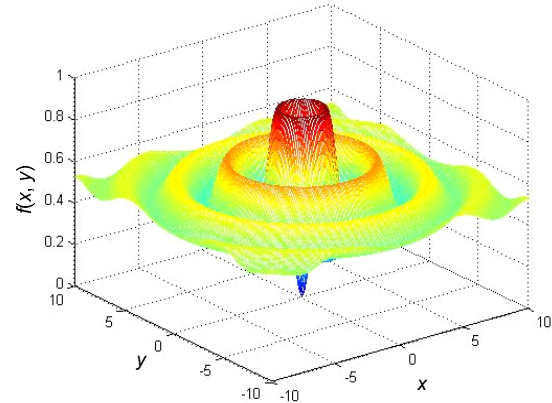
Function	Formula
F_1	$\sum_{n=1}^N x_n^2$
F_2	$\sum_{n=1}^N x_n - 10 \cos(\sqrt{10x_n})$
F_3	$x \sin(4x) + 1.1y \sin(2y)$
F_4	$10N + \sum_{n=1}^N [x_n^2 - 10 \cos(2\pi x_n)]$

For example, we can write $F_1 = x_1^2 + x_2^2 + \dots + x_N^2$. It is obvious that to minimize F_1 , we can minimize N separate functions which are not related to each other. Although some of the test functions (e.g. F_4 , shown in Fig. 1) have complex surfaces with many local optima (here minima) in N -dimensional space, these kinds of problems are very easy to solve because of their non-epistatic nature. In other words, there is no need to use a GA, since it tries to solve problems in N -dimensional space and encounters the curse of dimensionality. Instead, by breaking the N -variable function into N one-variable functions, N 1D problems can be solved without encountering the curse of dimensionality.

On the other hand, some test functions are epistatic, such as those mentioned in Table 2. As an example, the surface of F_6 is shown in Fig. 2.

**Fig. 1 Surface of function F_4 for $N=2$** **Table 2 Epistatic functions**

Function	Formula
F_5	$\sum_{n=1}^N [100(x_{n+1} - x_n^2)^2 + (1 - x_n)^2]$
F_6	$0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{1 + 0.1(x^2 + y^2)}$

**Fig. 2 Surface of function F_6**

In case of F_6 , it can be seen that it is a highly epistatic function and the optimum of none of the variables could be evaluated separately. Thus, how do GAs deal with such a problem? This question will be answered in the next section.

3 How does a GA work?

It should be noted that binary genetic algorithm (BGA) and continuous genetic algorithm (CGA) do not have conceptual differences. The only difference is how they encode and decode a problem. Considering the nature of the problem, one of the two methods will lead to a better result. For example, when the variables are naturally quantized, the BGA fits better, and when the variables are continuous, it is mostly better to use a CGA. We continue the discussion using a CGA as it has a more sensible demonstration and is easier to understand. The flowchart of this algorithm is shown in Fig. 3.

The steps by which an optimization problem is solved are analyzed in this section, based on the CGA flowchart. Although defining appropriate parameters and a suitable cost function definitely improves the efficiency of any optimization algorithm, it is clear that the final solution is not achieved at the step. Surely a suitable initial population of chromosomes increases the chance of finding the correct solution. However, obviously it is not the core of GAs. Then, the cost function (or fitness function) for each chromosome is calculated. This is only an evaluation of existing solutions. The next step (the fourth box) is

natural selection. In this step, it is decided which chromosomes should survive and be transferred to the next level or be used to generate new offspring. The selection process is important and influences the operation of the algorithm; however, a new solution is still not generated. The fifth box, which is essential to this study, is discussed in more detail in the next paragraph. The sixth box, which belongs to mutation, is important as it increases the variety (diversity) of the chromosomes by making random changes in some solutions. Although in this step some new solutions, which did not exist before, are generated, it should not be considered as the main operator. Since increasing the mutation rate in a GA leads to a random search, the rate of mutation is very low in most cases. Finally, the last box is related to the conditions of the algorithm's completion and has no effect on the solution.

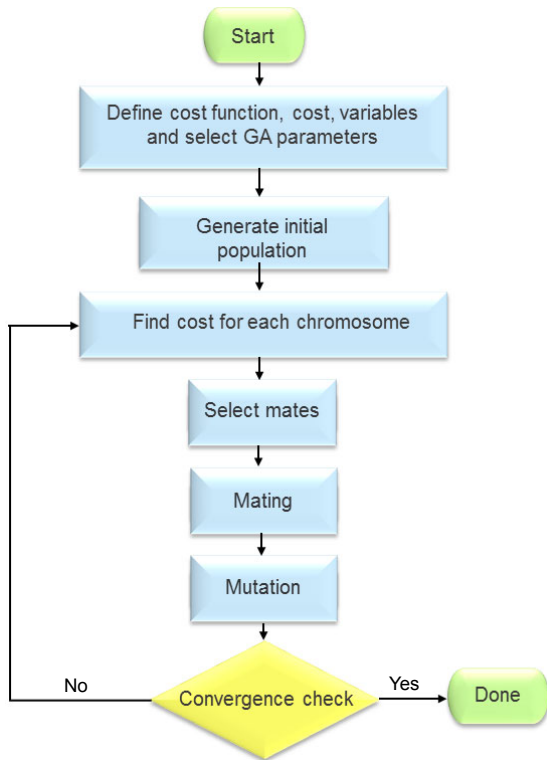


Fig. 3 Flowchart of CGA

By inspecting the GA closely, it is clear that it is the mating (crossover) operator that plays the main role in finding the optimum solution. With the hope of finding new and better solutions, the mating operator generates new children from current parents. Although several main strategies and secondary

methods have been introduced for mating in different studies, the main operation of mating is based on the random replacement of variables between the existing good solutions, with the hope of generating a new solution with “better genes” than the parents. By using this operator, we automatically accept that the current value of each variable can be good, normal, bad, or negligible. This means that we suppose a set of variables in which the superposition theorem applies (while in the real world this is mostly not the case). In other words, the total is more than the sum of components and the difference between them is the extra information which exists in the structure.

To present a proper example for the above discussion, a new test is introduced in Section 4.

4 Clover: a function to challenge GA

Consider the following objective function (Eq. (2)) in which θ is a definite parameter. We have named this function as “Clover” because of its special shape.

$$F(x, y) = -(e^{-(x \cos \theta - y \sin \theta)^2} + e^{-(x \cos \theta + y \sin \theta)^2}). \quad (2)$$

Changing θ does not make any changes to the surface shape of this function and just rotates it around the F axis. In fact, it gives a θ -radian rotation to the function around the F axis. Figs. 4a–4d show this function's schema for $\theta=0$, $\theta=\pi/12$, $\theta=\pi/6$, and $\theta=\pi/4$, respectively.

This function is analytical, continuous, and unimodal. It has just one minimum (Eq. (3)) which can be obtained easily by differentiation:

$$\min(F(x, y)) = F(0, 0) = -2. \quad (3)$$

The function can be rewritten for $\theta=0$ as

$$F(x, y) = -(e^{-x^2} + e^{-y^2}) = f_1(x) + f_2(y). \quad (4)$$

In this case, there is no epistasis between x and y variables; so Eq. (5) is true:

$$\text{opt}(F(x, y)) = \text{opt}(f_1(x)) + \text{opt}(f_2(y)). \quad (5)$$

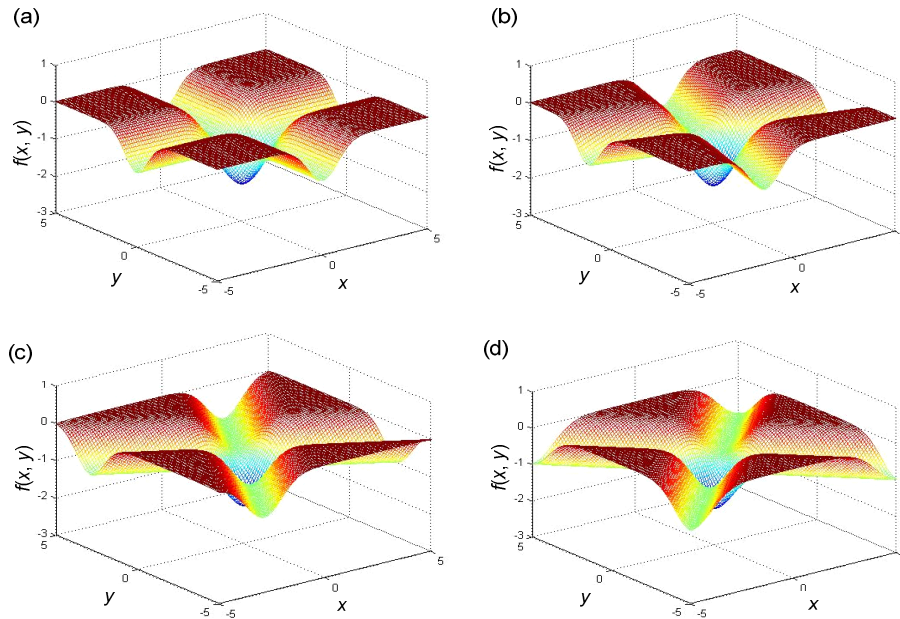


Fig. 4 Clover function for $\theta=0$ (a), $\theta=\pi/12$ (b), $\theta=\pi/6$ (c), and $\theta=\pi/4$ (d)

To solve this problem using a GA for a determined range of variables, for example, $-L/2 \leq x, y \leq L/2$, the search space is equivalent to L^2 . On the other hand, to find the optimum set of variables in this example, $F(x, y) = f_1(x) + f_2(y)$, just the optimum of each variable separately in its own range needs to be found. In fact, the search space for this case is equivalent to $2L$, the same as the word search example in Section 2 in which the real search space was eight 1D spaces not one 8-dimensional space. Therefore, if there is no epistasis between variables, it is clear that a GA or other algorithms, which search the N -dimensional space, are useless, especially when the number of variables or their ranges increase.

A simple way to find the optimum of such functions is to start from an arbitrary initial condition. Then, by choosing the variables one by one and keeping the other variables frozen, we can search the defined range for each variable separately with any desired resolution and find the optimum value for each variable independently (generalized form of line search).

Now, suppose that in the Clover function for $\theta=\pi/6$:

$$F(x, y) = -\left(e^{-(x-y)^2/2} + e^{-(x+y)^2/2}\right). \quad (6)$$

Eq. (7) should then be true:

$$F(x, y) \neq f_1(x) + f_2(y). \quad (7)$$

In this case, the optimum cannot be obtained by finding each variable's optimum separately. Thus, finding an optimum for a variable is not meaningful. In the previous status ($\theta=0$), we saw that using a GA is not efficient. However, if the use of a GA is insisted upon, it can find the optimum solution (in a very inefficient way). However, when $\theta=\pi/4$, GAs usually cannot arrive at the correct solution. To make this clearer, assume we have the chromosomes in Table 3.

Table 3 Selected chromosomes for clarifying the mating problem

Chromosome	x	y	Chromosome	x	y
1	0.8	0.75	4	7	0
2	2	-2	5	0	3
3	-4	-4			

Chromosomes Nos. 1, 2, and 3 are quite not-bad solutions for this problem. However, it can be seen that by mating them, some chromosomes with lower fitness are generated. On the other hand, despite not being suitable solutions, chromosomes Nos. 4 and 5 can generate the global minimum (0, 0) by mating, which is the global optimum. This issue is against the theory of mating which says that parents with high

fitness have a greater chance of generating better offspring in the next generations. In epistatic cases, parents with good interactions between their variables (variables with appropriate order) have higher fitness values than the other parents. In such cases, the mating operator destroys these orders.

One of the points that the supporters of GAs mention is that “This algorithm works”. Table 4 shows the simulation results of applying a GA on Clover function for $\theta=0$ and $\theta=\pi/4$. To avoid possible programming mistakes, MATLAB Optimization Toolbox has been used for these simulations. In this case, an initial population with 20 chromosomes is formed and a random and monotonous initial distribution of variables is done between $-D$ and $+D$ (where D is an arbitrary range for optimization variable, $D=20$). Other parameters are the default parameters of the toolbox. The algorithm was run for 50 repeats and the results are represented in Table 4.

Table 4 Simulation results of applying a GA on Clover function for $\theta=0$ and $\theta=\pi/4$

θ	Number of successes in finding the optimum	Number of failures in finding the optimum	Average of solutions
0	50	0	-2
$\pi/4$	26	24	-1.543

A simple comparison between simulation results shows that applying GAs on the Clover function for two different values of θ leads to two different results with a significant difference in the efficiency of the algorithm. In Section 1, where some of the main advantages of GAs were listed, it was mentioned that one of the important advantages of GAs is that they do not require the system’s formula, and the values of the objective function in some points are enough for them to work. If so, why does the GA’s efficiency notably decrease by such a simple change of a parameter which only leads to a rotation of the cost function surface? Bear in mind that this is a very simple 2D problem without any complexity, discontinuity, or local minima.

5 Other examples

Suppose an objective function as shown in Eq. (8) in which $r = \sqrt{x^2 + y^2}$ and $\theta = \arctan(y/x)$.

$$F(x, y) = -r \times e^{-(r-\theta)^2}. \quad (8)$$

We have named this the “Spiral Channel” function. Like the Clover function, this is an epistatic function. Although it is not differentiable for the positive values on the x axis, starting from any arbitrary point and using a gradient would easily lead to its only minimum at $(2\pi, 0)$ with the cost equal to -2π . The function surface is shown in Fig. 5 and the result of applying the GA on this function is presented in Table 5.

Now, consider the following new objective function (named here as the “Zagros” function):

$$F(x, y) = -\frac{\cos(ax + by)}{[(ax + by)^2]^m + k} - \frac{\cos(cx + dy)}{[(cx + dy)^2]^m + k}, \quad (9)$$

where $a=2$, $b=1$, $c=2$, $d=-1$, $m=0.2$, and $k=1$. This is a highly epistatic function in which the variables are not important separately. In addition, it has a lot of local minima that make its optimization very difficult. The function is depicted in Fig. 6. By applying a GA on this function, the results in Table 6 are obtained.

Then, we define two new variables as shown in Eq. (10) to eliminate epistasis of this function:

$$\begin{cases} u = ax + by, \\ v = cx + dy. \end{cases} \quad (10)$$

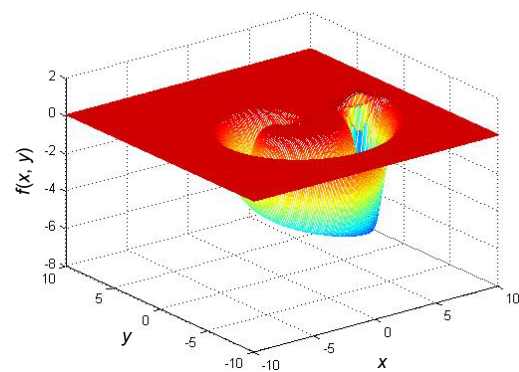


Fig. 5 Surface of the Spiral Channel function

Table 5 Simulation results of applying a GA on the Spiral Channel function

Number of successes in finding the optimum	Number of failures in finding the optimum	Average of solutions
21	29	-4.66

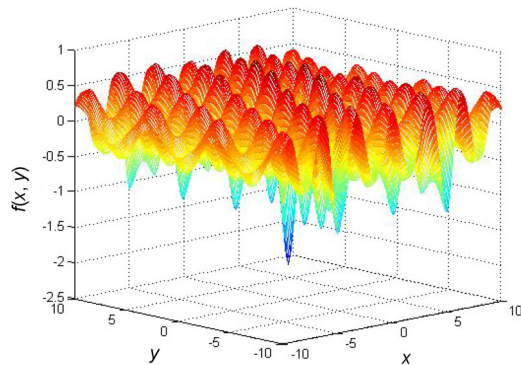


Fig. 6 Surface of the Zagros function

Table 6 Simulation results of applying a GA on the Zagros function

Number of successes in finding the optimum	Number of failures in finding the optimum	Average of solutions
1	49	-1.57

By this change of variables, the problem is converted to Eq. (11), which is clearly non-epistatic:

$$F(u, v) = -\frac{\cos(u)}{(u^2)^m + k} - \frac{\cos(v)}{(v^2)^m + k}. \quad (11)$$

By applying a GA on the $F(u, v)$, the results in Table 7 are obtained.

Comparing the results in Table 6 and Table 7, as expected, the GA shows a very poor performance in a highly epistatic condition, but on removing the epistasis, the GA does find the solution (though it does not do it in an optimized way). As mentioned before, in non-epistatic conditions, simple algorithms can lead to the correct solution with much less computational cost compared to GAs.

Table 7 Simulation results of applying a GA on the Zagros function with new variables

Number of successes in finding the optimum	Number of failures in finding the optimum	Average of solutions
50	0	-2

6 Conclusions

In this paper, the genetic algorithm technique was challenged and criticized. We claim that, in spite

of the generally positive assumption, a GA is not an efficient and useful optimization algorithm. It seems that this algorithm cannot converge successfully in the real-life problems and if it converges to an acceptable solution, it is not computationally efficient. The cost functions can be classified into two groups: epistatic and non-epistatic. In this paper, it was shown that if the cost function is non-epistatic, the problem can be solved using GAs. However, it would be more efficient to use simple algorithms instead of GAs, since the N -dimensional function can be divided into N 1D functions. Further, in most epistatic cost functions, GAs cannot find the correct solution. In addition to some commonly used benchmarks, our claim was evaluated and confirmed by three new test functions which were designed in this study. These test functions were designed in a way that could be employed in both epistatic and non-epistatic conditions. Closer inspection shows that the main obstacle in a GA is the main operator, crossover. Disconnecting the suitable relations between variables by destructing the existing appropriate structures, crossover plays a negative role in the GA evolution. We encourage interested readers, especially those who use GAs, to investigate our claim in some real-world optimization problems. In this paper, our discussion on the degrees of epistasis is qualitative. We encourage those readers who are interested in this area to propose a proper method for measuring the degrees of epistasis.

Acknowledgements

The authors would like to thank Dr. Mansour Rasoulzadeh DARABAD and Dr. Pegah T. HOSSEINI (Department of Electronic and Computer Sciences, University of Southampton, UK) for their kind help and support in enhancing the quality of this paper.

References

- Davis LD, de Jong K, Vose MD, et al., 2012. *Evolutionary Algorithms*. Springer, New York, USA.
<https://doi.org/10.1007/978-1-4612-1542-4>
- de Oliveira LL, Freitas AA, Tinós R, 2018. Multi-objective genetic algorithms in the study of the genetic code's adaptability. *Information Sciences*, 425:48-61.
<https://doi.org/10.1016/j.ins.2017.10.022>
- di Francescomarino C, Dumas M, Federici M, et al., 2018. Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, 74:67-83.
<https://doi.org/10.1016/j.is.2018.01.003>

- Dong HB, Li T, Ding R, et al., 2018. A novel hybrid genetic algorithm with granular information for feature selection and optimization. *Applied Soft Computing*, 65:33-46. <https://doi.org/10.1016/j.asoc.2017.12.048>
- Greco A, D'Urso D, Cannizzaro F, et al., 2018. Damage identification on spatial Timoshenko arches by means of genetic algorithms. *Mechanical Systems and Signal Processing*, 105:51-67. <https://doi.org/10.1016/j.ymssp.2017.11.040>
- Guo LH, Wang GG, Gandomi AH, et al., 2014. A new improved krill herd algorithm for global numerical optimization. *Neurocomputing*, 138:392-402. <https://doi.org/10.1016/j.neucom.2014.01.023>
- Haupt RL, Haupt SE, 2004. *Practical Genetic Algorithms*. John Wiley & Sons, Hoboken, New Jersey, USA.
- Jain A, Chaudhari NS, 2017. An improved genetic algorithm for developing deterministic OTP key generator. *Complexity*, 2017:7436709. <https://doi.org/10.1155/2017/7436709>
- Karakatič S, Podgorelec V, 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519-532. <https://doi.org/10.1016/j.asoc.2014.11.005>
- Qu BY, Liang JJ, Wang ZY, et al., 2016. Novel benchmark functions for continuous multimodal optimization with comparative results. *Swarm and Evolutionary Computation*, 26:23-34. <https://doi.org/10.1016/j.swevo.2015.07.003>
- Sivanandam SN, Deepa SN, 2008. *Introduction to Genetic Algorithms*. Springer, Berlin, Heidelberg, Germany. <https://doi.org/10.1007/978-3-540-73190-0>
- Steinberg ML, Cosloy SD, 2009. *Biotechnology and Genetic Engineering*. Infobase Publishing, New York, USA.
- Teimouri R, Baseri H, Rahmani B, et al., 2014. Modeling and optimization of spring-back in bending process using multiple regression analysis and neural computation. *International Journal of Material Forming*, 7(2):167-178. <https://doi.org/10.1007/s12289-012-1117-4>
- Tseng HE, Chang CC, Lee SC, et al., 2018. A block-based genetic algorithm for disassembly sequence planning. *Expert Systems with Applications*, 96:492-505. <https://doi.org/10.1016/j.eswa.2017.11.004>
- Wang H, Zhao ZZ, Guo ZW, et al., 2017. An improved clustering method for detection system of public security events based on genetic algorithm and semisupervised learning. *Complexity*, 2017:8130961. <https://doi.org/10.1155/2017/8130961>

- Zhou Y, Zhou LS, Wang Y, et al., 2017. Application of multiple-population genetic algorithm in optimizing the train-set circulation plan problem. *Complexity*, 2017:3717654. <https://doi.org/10.1155/2017/3717654>

中文概要

题目：上位效应对遗传算法可靠性的影响

目的：探讨遗传算法的局限性和实用性，并分析基于相互作用产生的上位效应对遗传算法可靠性的影响。

创新点：1. 指出遗传算法缺陷的根源；2. 基于测试样本函数定义目标函数，以判断遗传算法的适用性。

方法：1. 基于非上位效应函数（表 1）和上位效应函数（表 2），以及非上位效应函数 F_4 和上位效应函数 F_6 的结构图来验证遗传算法可靠性；2. 通过计算样本函数（公式（1））和遗传算法流程（图 3）表达遗传算法的工作原理。3. 利用克洛弗函数（公式（2））和计算不同结构角下的函数分布（图 4），进一步判断匹配度（表 3）和计算效率（表 4）；定义新的目标函数（公式（9））和一组新的变量（公式（10））来实现变量相关性解离。

结论：1. 对当前遗传算法存在的不足给出了独到见解，并认为正定性的假设并非可以保证遗传算法实际的有效性和优化性。2. 定义成本代价函数用以判断遗传算法可靠性，并分别考虑上位性和非上位性效应两种情形。当成本代价函数在非上位性效应下时，遗传算法是有效的；否则，可以把 N 维函数降级为 N 个一维函数，从而采用更简单的算法来判断。基于一些通用的基准，进一步设计三类样本函数来证实以上判断，且这些样本函数适合于上位性效应情形和非上位性效应情形。3. 遗传算法的瓶颈在于主算子和相干匹配性；可以通过破坏某些结构来实现变量关系的解离，从而抑制相干匹配性对遗传算法的影响。希望相关读者在处理实际优化问题时能验证作者关于上位效应的定性结论，并给出更可靠的方法来表征这种效应。

关键词：上位性效应；遗传算法；相干匹配性；叠加性；优化；成本代价函数