A POTENTIAL WEAKNESS OF THE GENETIC ALGORITHM IN MULTIMODAL OPTIMIZATIONS

Roman Knobloch and Jaroslav Mlynek

Technical University of Liberec, Department of Mathematics, e-mail: roman.knobloch@tul.cz, jaroslav.mlynek@tul.cz

KEYWORDS

Optimization, genetic algorithm, generation, crossover, mutation, selection, multimodal fitness function, random search.

ABSTRACT

The genetic algorithms are robust and stable evolutionary computation techniques in the sense that they mostly provide a solution. This fact is undisputedly an important positive feature, particularly in the situation when no feasible solution is available. On the other hand, this robustness and the fact that living nature utilizes analogous mechanisms for its development and propagation sometimes lead to unrealistic expectations and exaggerated faith in their capabilities. In particular, when they are used as function optimizers they sometimes tend to demonstrate also their weak points. In this article, we demonstrate the limits of the genetic algorithm in a multimodal model task. The task consists in the identification of a de Bruijn sequence of a given length between general binary sequences. Generally, the multimodality of the fitness function can lead to the stagnation of the genetic algorithm well below the desirable fitness function values which hinders the genetic algorithm from finding the optimized solution of the task under consideration.

INTRODUCTION

During the last decades, evolutionary computation techniques have become an important tool for solving diverse and wide-ranging tasks in science, industry, technology, and economics - see for instance Mitsuo and Runwei (2000), Goldberg (2000). The steady and fast progress in the power of computational facilities contributed to the wider use of these methods. Parallel computing technology and processes made these procedures even more of a practical interest.

A group of genetic algorithms is probably the most known representative of evolutionary computation techniques. These algorithms use the principles of natural evolution as postulated by Charles Darwin and Johann Gregor Mendel in the 19th century. They are stochastic methods that incorporate the endeavour for survival and genetic inheritance. These algorithms form a population consisting of a certain number of independent individuals. Each of these individuals represents a potential solution to the task under consideration.

The operation of genetic algorithms in their standard original alternative is controlled by three characteristic operators. These are: the selection of individuals for another generation based on the value of the fitness function, cross-over, and mutation. It should also be pointed out that the first two operators - that is the selection and cross-over form the principal part of the genetic algorithm functioning. On the other hand, the role of the mutation is rather subtle and substantially different. It is a minor mechanism and its role is to keep the population of potential solutions from stagnation.

The fundamental idea standing behind the cross-over operator is that if you combine two good potential solution candidates for the given task, you can reasonably expect to obtain again good, preferably even better solution candidates. This simple concept proves prospective and fruitful in many circumstances. Unfortunately, there also exist situations, where this assumption fails or at least leads to questionable results or unconvincing substandard algorithm behaviour.

To demonstrate this potential weakness of the genetic algorithm, we choose a model task of binary vector optimization. Specifically, the task is to identify a de Bruijn sequence of a given length between general binary sequences. The task of finding a de Bruijn sequence serves exclusively as a model optimization problem. There are several diverse and more efficient methods to form de Bruijn sequences. The interested reader can consult for instance Sawada et al. (2016) or Levine (2011). This means that de Bruijn sequences themselves are not our principal aim. The primary intention is to investigate how the genetic algorithm behaves in the process of a relatively simple binary optimization problem.

We should also mention the fact that the term genetic algorithms includes nowadays a large family of similar algorithms inspired by natural selection and hereditary mechanisms. These algorithms differ in their specific implementations, selection mechanisms, cross-over, and mutation procedures. They can also include various additional mechanisms contributing to their efficiency or are useful under special circumstances. These include for instance elitism feature (keeping the best up to now found individual and contingently returning it into the subsequent

generations). All these improvements and modifications complicate in certain ways the functioning of the original genetic algorithm and additionally can prove themselves beneficial only in special circumstances.

To keep things simple and clear we used primarily the genetic algorithm in its pure original form. We utilized the conventional selection mechanisms - roulette wheel selection and tournament selection. We implemented the classical one-point crossover operator controlled by one probabilistic parameter and simple mutation also guided by one probabilistic parameter. In principle, it would not be a problem to program more sophisticated and complex features into our version of the genetic algorithm (in fact we even did it in some cases - e.g. elitism). Nevertheless, finally, we decided to keep our version of the genetic algorithm in its standard and simple form. The main reason is that our investigations, conclusions, ideas, and comments have then a broader impact and greater applicability.

OPTIMIZATION

There is a wide class of important tasks or problems for which no suitable analytic solution strategy or algorithm exists. Many of these tasks are of great importance for science, industry, technology, or economics - see for example Mitsuo and Runwei (2000), Goldberg (2000). In general, almost any task can be formulated as an optimization problem. The search for a solution can then be interpreted as a search through a space of potential solutions. This space is sometimes called the search space. For reasonably small search spaces exhaustive techniques are applicable. For large search spaces, the exhaustive techniques are not adequate and special stochastic searching methods have to be utilized. Genetic algorithms belong to such randomized techniques.

Let us consider fitness function f defined on the set of binary vectors of length λ, which assigns a value from the set of natural numbers $\mathbb{N}$ to each binary vector. Function f can be symbolically expressed as

$$f : \{0,1\}^{\lambda} \mapsto \mathbb{N}.$$

Let

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{\lambda}) \tag{1}$$

is a binary vector of length λ, $f(\alpha) \in \mathbb{N}$. The cardinality of the set of such binary vectors is determined by relation

$$|\{0,1\}^{\lambda}| = 2^{\lambda}. \tag{2}$$

In the following parts of the article we will concentrate on the identification of $\alpha^{max} \in L^*$, where

$$L^* = \{\alpha^* : f(\alpha^*) = max\{f(\alpha) : \alpha \in \{0,1\}^{\lambda}\}\}.$$

Vector α^{max} can be identified by successive calculations of the function values $f(\alpha)$ at the systematic search of all possible vectors α from relation (1) or by their random choice. For a large search space defined by relation (2) this procedure is not suitable and therefore we use a genetic algorithm to identify the maximum of the fitness function f.

GENETIC ALGORITHM

Below we present the operation scheme of a standard genetic algorithm. In this algorithm, we call the binary vector α defined by relation (1) an individual.

Genetic algorithm operation scheme

1. Create the initial generation of individuals randomly (or alternatively according to a prescribed scheme). We denote the number of individuals by symbol n_p.
2. Evaluate all individuals by means of the fitness function f.
3. Repeat until the terminating criterion is met (usually the number of generations denoted by symbol n_g).
   (a) Select the individuals for the next generation by means of a roulette wheel or by another mechanism favouring individuals with higher values of the objective function f.
   (b) Select randomly couples of individuals and perform the cross-over operation with them - see Figure 1. The cross-over procedure is controlled by cross-over probability denoted by symbol P_{cr}.
   (c) Submit the resulting individuals to mutation - see Figure 2. The mutation procedure is controlled by mutation probability denoted by symbol P_m.
   (d) Evaluate all individuals in the generation by means of the fitness function f.
   (e) Store the best individual in the current generation.

   Endrepeat.
4. Show the best individual.

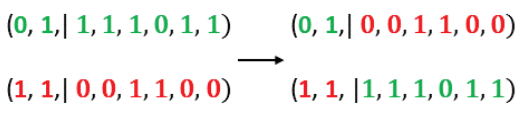

Figure 1: Principle of the one-point cross-over operation. The cross-over point is denoted by a vertical line.

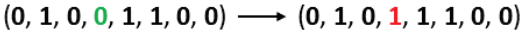

Figure 2: Principle of the mutation operation. The mutation took place at the 4th position of the binary vector.

The fitness proportionate selection, sometimes called the roulette wheel, is a standard selection mechanism used by genetic algorithms. This selection scheme, as well as some more advanced selection procedures, are described in standard introductory books on genetic algorithms, for instance Michalewicz (1999) or Simon (2013).

The part 3.(e) forms a usual part even of the standard genetic algorithm. Without it, the algorithm can lose and often loses the best-found solution candidate during the transition to the next generation.

BINARY DE BRUIJN SEQUENCES

To test the operation of the genetic algorithm we need a suitable model task. This task should be easily scalable, so that we can test the genetic algorithm at different levels of difficulty. As a model task we chose a search for binary de Bruijn sequences of a given length.

De Bruijn sequences are interesting objects from the area of discrete mathematics relating to combinatorics and coding. As simple examples of binary de Bruijn sequences we can present for instance

$$0011$$

$$00010111.$$

The first case exhibits a cyclic sequence containing all possible variations of 2nd order composed of two symbols 0, 1. Specifically, codes 00, 01, 11, and 10. The second case gives a cycle sequence containing all possible variations of 3rd order. Specifically, codes 000, 001, 010, 101, 011, 111, 110, and 100. In the de Bruijn sequence, all codes are used just once and all possible codes are used. We should point out once more that all de Bruijn sequences are considered cyclic. This means that the codes corresponding to the 6th, 7th and 8th positions are respectively: 111, 110 and 100.

An interested reader can find more detailed information concerning de Bruijn sequences either in the original article by de Bruijn, de Bruijn (1946) or consult more up-to-date sources, for instance Sawada et al. (2016) or Levine (2011).

Definition. A binary de Bruijn sequence b of order κ is a string of bits $b_i \in \{0,1\}$, $b = \{b_1, \ldots, b_{2^\kappa}\}$ such that each string of length κ, $\{a_1, \ldots, a_\kappa\} \in \{0,1\}^\kappa$ occurs exactly once in b.

The following well-known theorem is taken from Levine (2011).

Theorem. Binary de Bruijn sequence of order κ exists for all κ, and there are exactly $2^{(2^{\kappa-1})}$ of them.

Here we use the following notation concerning the de Bruijn sequences. By symbol κ we denote the number of positions in the codes considered in de Bruijn sequences. For instance, if we are interested in de Bruijn sequences with 4-positional binary codes (specifically 0000, 0001, ..., 1111) the value of κ equals 4. For brevity, the quantity κ will further be termed as the code width. The symbol

Table 1: Length of de Bruijn sequences λ and numbers of binary β and de Bruijn δ sequences in dependence on the code width κ

Code width κ	Length of de Bruijn seq. λ	Number of de Bruijn seq. δ	Number of binary seq. β
1	2	2	4
2	4	4	16
3	8	16	256
4	16	256	65536
5	32	65536	$4.295 \cdot 10^9$
6	64	$4.295 \cdot 10^9$	$1.845 \cdot 10^{19}$
7	128	$1.845 \cdot 10^{19}$	$3.403 \cdot 10^{38}$
8	256	$3.403 \cdot 10^{38}$	$1.158 \cdot 10^{77}$
9	512	$1.158 \cdot 10^{77}$	$1.341 \cdot 10^{154}$
10	1024	$1.341 \cdot 10^{154}$	$1.798 \cdot 10^{308}$

λ stands for the length of the corresponding de Bruijn sequences. Obviously, we have

$$\lambda = 2^\kappa, \tag{3}$$

since there are exactly 2^κ different binary codes with κ positions.

By β we denote the number of all possible binary sequences of a specified length.The length will always be the same as the length of the corresponding de Bruijn sequences. Because we know that the length of a de Bruijn sequence is 2^κ, we automatically have for the length of the de Bruijn sequence $\lambda = 2^\kappa$. This means we can immediately determine the value of β,

$$\beta = 2^\lambda = 2^{(2^\kappa)}. \tag{4}$$

Symbol δ we use for the number of de Bruijn sequences characterized by the same value of κ. In conformity with the preceding theorem, we have for the value δ the following relation

$$\delta = 2^{(2^{\kappa-1})}. \tag{5}$$

The straightforward comparison of expressions (4) and (5) gives us a surprisingly simple relation between quantities β and δ

$$\delta = 2^{(2^{\kappa-1})} = 2^{\frac{2^\kappa}{2}} = (2^{(2^\kappa)})^{\frac{1}{2}} = \beta^{\frac{1}{2}} = \sqrt{\beta}.$$

To provide the reader with a more quantitative idea about the numbers of de Bruijn sequences with respect to the numbers of all possible binary sequences of the same length we include these quantities for several small values of κ in the following Table 1.

RANDOM SEARCH PROBABILITIES

For our further considerations it will be useful to have a clear idea about probabilities of identification of one specific unique de Bruijn sequence and of finding an arbitrary

de Bruijn sequence by a random choice. By a random choice, we mean filling up a sequence with 2^κ positions by symbols 0 and 1 randomly. That is the probability of putting 0 or 1 into any position is exactly equal to $\frac{1}{2}$. It is, of course, to expect that hitting the specific (fully determined) de Bruijn sequence is much less probable than hitting an arbitrary de Bruijn sequence available, since there are many different de Bruijn sequences, specifically according to (5) there are exactly $2^{(2^{\kappa-1})} = \sqrt{2^\lambda}$ of them.

Probability of identifiing the specific de Bruijn sequence

We choose one specific de Bruijn sequence. This is a unique specimen we want to find. The probability of getting this specific de Bruijn sequence by one random choice is given by relation

$$\frac{1}{2^\lambda}.$$

This fact indicates that the probability that we miss this specific individual at one random try is

$$1 - \frac{1}{2^\lambda}.$$

The probability of missing the specific individual at n random tries is

$$(1 - \frac{1}{2^\lambda})^n.$$

The probability of hitting the specific individual at n random tries is then

$$1 - (1 - \frac{1}{2^\lambda})^n.$$

Now, let us suppose that we want this probability to be bigger than a fixed quantity ρ,

$$1 - (1 - \frac{1}{2^\lambda})^n > \rho.$$

Using simple modifications we can easily express n

$$n > \frac{\ln(1-\rho)}{\ln(1 - \frac{1}{2^\lambda})}. \tag{6}$$

The quantity n determines the number of random tries ensuring to hit the specific bit sequence with probability bigger than ρ. The relation (6) represents the lower estimate for n. The formula (6) can be further simplified using a Taylor expansion of the natural logarithm function in the vicinity of 1 - see Rudin (1976). The Taylor expansion gives

$$\ln(1+x) \approx x - \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \dots. \tag{7}$$

This means that for x small we have an approximation $\ln(1+x) \approx x$ and for $x < 0$ we even have relation $|x| < |\ln(1+x)|$. This indicates that the approximation does not spoil the lower estimate property. That is

$$n > \frac{\ln(1-\rho)}{\ln(1 - \frac{1}{2^\lambda})} \approx -2^\lambda \ln(1-\rho). \tag{8}$$

To illustrate the meaning of the formula (8) let us suppose the probability value $\rho = \frac{1}{2}$. This assumption gives the lower estimate of the quantity $n_{s,0.5}$ which is the number of random choices that guarantees to achieve the specific de Bruijn sequence with probability greater than $\frac{1}{2}$

$$n_{s,0.5} > 2^\lambda \ln 2. \tag{9}$$

This means that to get the specific individual by random choices with probability $\rho = \frac{1}{2}$, we have to generate at least $2^\lambda \ln 2$ individuals, where 2^λ is the total number of different bit sequences. The relation (9) represents a rigorous lower estimate. The value of $\ln 2 \simeq 0.693148$. In practical terms, we should generate at least 70% of the total number of all possible bit sequences to have the probability of hitting the specific bit sequence bigger than 50%.

Probability of getting an arbitrary de Bruijn sequence

This case is analogous to the previous part. For the definite code width κ we have according to (5) the following number of various de Bruijn sequences $\delta = 2^{(2^{\kappa-1})}$. This number determines the relative frequency of de Bruijn sequences among general binary sequences. The total number of all binary sequences with a length 2^κ is according to (4) $\beta = 2^{(2^\kappa)}$. To keep formulas in a simpler form, we use the relation (3) $\lambda = 2^\kappa$. The probability of getting an arbitrary de Bruijn sequence at a random choice is then equal to the ratio of positive cases to all possible cases. In our circumstances, it induces that the probability of getting an arbitrary de Bruijn sequence is

$$\rho = \frac{2^{\lambda^{\frac{1}{2}}}}{2^\lambda} = \frac{1}{2^{\frac{\lambda}{2}}}.$$

We can carry out the same considerations as in the previous part to obtain the number n of random choices that secure finding an arbitrary de Bruijn sequence with probability greater than ρ. So, we get

$$n > \frac{\ln(1-\rho)}{\ln(1 - \frac{1}{2^{\frac{\lambda}{2}}})}.$$

Using the Taylor expansion (7) gives

$$n > -2^{\frac{\lambda}{2}} \ln(1-\rho)$$

and putting $\rho = \frac{1}{2}$, we get the final formula

$$n_{a,0.5} > 2^{\frac{\lambda}{2}} \ln 2, \tag{10}$$

where the symbol $n_{a,0.5}$ denotes he minimal number of random choices ensuring to identify an arbitrary de Bruijn sequence with probability greater than $\frac{1}{2}$. The values $n_{a,0.5}$ and $n_{s,0.5}$ given by relations (9) and (10) for small values of κ and λ are summarized in Table 2. The notation $\lceil x \rceil$ of a real number x is defined as $\lceil x \rceil = min\{p \in \mathbb{Z}, p \geq x\}$, where $\mathbb{Z}$ denotes the set of all integer numbers.

Table 2: Numbers of random choices ensuring identifying any de Bruijn sequence $\lceil n_{a,0.5} \rceil$ and the specific de Bruijn sequence $\lceil \mathrm{n}_{s,0.5} \rceil$ with probability greater than $\frac{1}{2}$ in dependence on the code width κ

Code width κ	de Bruijn seq. length λ	Number of rand. choices $\lceil n_{a,0.5} \rceil$	Number of rand. choices $\lceil n_{s,0.5} \rceil$
1	2	2	3
2	4	3	12
3	8	12	178
4	16	178	45427
5	32	45427	$2.098 \cdot 10^{9}$
6	64	$2.098 \cdot 10^{9}$	$1.279 \cdot 10^{19}$
7	128	$1.279 \cdot 10^{19}$	$2.359 \cdot 10^{38}$
8	256	$2.359 \cdot 10^{38}$	$8.026 \cdot 10^{76}$
9	512	$8.026 \cdot 10^{76}$	$9.294 \cdot 10^{153}$
10	1024	$9.294 \cdot 10^{153}$	$1.246 \cdot 10^{308}$

GENETIC ALGORITHM TESTING

We tested the operation of the standard genetic algorithm in two configurations:

1. **Identification of a specific fixed de Bruijn sequence.**

   In this case, we select one specific de Bruijn sequence and subsequently try to identify this specific de Bruijn sequence by the genetic algorithm. Under such circumstances, it is obvious that this task has a unique solution. Technically, we compare binary individuals with the selected de Bruijn sequence that is further termed as the specimen. If the individual has the same binary value at a given position, the value of the fitness function is incremented by 1.

   Fitness function f_1 is defined by the following prescription. Let $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_\lambda)$ is the given fixed de Bruijn sequence of length λ and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_\lambda)$ an arbitrary binary vector of length λ. Then

   $$f_1(\alpha) = \sum_{i=1}^{\lambda} p_i,$$

   where $p_i = 1$ for $\alpha_i = \gamma_i$ and $p_i = 0$ for $\alpha_i \neq \gamma_i$.

   In this way, individuals similar to the specimen are considered better (have a higher value of the fitness function) and they have higher chances to proceed to subsequent generations. In this way, the genetic algorithm population evolves relatively quickly and successfully and without any apparent obstacles. The specimen is identified relatively fast, of course, the actual number of generations necessary to identify the specimen is dependent on the length of the specimen. This is natural since the length determines the general difficulty of this optimization task.

2. **Identification of an arbitrary de Bruijn sequence.**

   In this case, we seek for an arbitrary de Bruijn sequence of the given length. The solution to this task is evidently not unique since according to (5) there are exactly $\delta = 2^{(2^{\kappa-1})}$ of such de Bruijn sequences. This indicates that from the perspective of random search, this task should be much easier since there are plenty of solutions waiting to be found. The fitness function is constructed in the following way. We traverse cyclically through the given bit sequence representing an individual. Each group of κ digits is interpreted as a number expressed in the binary system. That is each individual obtains a sequence of numbers. We check this sequence and each number is counted only once that is we remove all duplicities. The fitness function value is then set to the quantity of unique numbers in the sequence.

   Fitness function f_2 is defined by the following prescription. Let us have binary vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_\lambda)$, where $\lambda = 2^\kappa$. Then

   $$f_2(\alpha) = s,$$

   where the value s is determined by the number of different binary codes of length κ contained in vector α. According to the Theorem above and relation (5) the number of different de Bruijn sequences is $\delta = 2^{(2^{\kappa-1})}$.

   When we seek for a de Bruijn sequence of length $\lambda = 2^\kappa$, the maximal possible fitness is equal to the length λ of the binary sequence and it is obvious that such an individual corresponds definitely to a certain de Bruijn sequence. This confirms that the fitness function is constructed correctly.

EXPERIMENTAL RESULTS

We present, in Tables 3, 4, and 5, a limited selection from the large pool of experimental results. The intention is to provide the reader with quantitative information regarding the behaviour of the genetic algorithm in the model task with different sizes. For each size of the task, the genetic algorithm was run 10 times. We used two selection strategies, standard roulette wheel selection and tournament selection with tournament size 4. The average result and standard deviation are stated in the last two rows of the tables.

Table 3 illustrates the identification of any de Bruijn sequence with length $\lambda = 16$. In the computation the following algorithm parameters were used: number of individuals $n_p = 20$, number of generations $n_g = 40$, cross-over probability $P_{cr} = 0.99$, mutation probability $P_m = 0.06$.

The algorithm attained the target and solved the task without problems even with small values $n_p = 20$ and $n_g = 40$.

Table 4 summarizes the identification of any de Bruijn sequence with length $\lambda = 64$. In the computation the following algorithm parameters were used: number of

Table 3: Search for a de Bruijn sequence with 16 positions. The maximal value of the fitness function is 16.

Exp. No.	Roulette selection	Tournament selection
1	16	16
2	16	16
3	16	16
4	16	16
5	16	16
6	16	16
7	16	16
8	16	16
9	16	16
10	16	16
Average	16.0	16.0
Stand. deviation	0.0	0.0

Table 4: Search for a de Bruijn sequence with 64 positions. The maximal value of the fitness function is 64.

Exp. No.	Roulette selection	Tournament selection
1	56	64
2	58	64
3	56	63
4	56	64
5	57	64
6	58	64
7	56	63
8	57	64
9	57	64
10	56	61
Average	56.7	63.7
Stand. deviation	0.823	0.972

individuals $n_p = 200$, number of generations $n_g = 800$, cross-over probability $P_{cr} = 0.99$, mutation probability $P_m = 0.02$.

Despite higher values n_p and n_g, the roulette wheel selection gives evidently insufficient results. The target value is 64. The tournament selection gives much better and acceptable results. The reason is that the tournament selection exerts generally much higher selection pressure in comparison with the roulette wheel selection. Another advantage of the tournament selection is that it is easily and suitably quantifiable by the tournament size parameter.

Table 5 illustrates the identification of any de Bruijn sequence with length $\lambda = 256$. In the computation the following algorithm parameters were used: number of individuals $n_p = 200$, number of generations $n_g = 800$, cross-over probability $P_{cr} = 0.99$, mutation probability $P_m = 0.004$.

From Table 5 it is apparent that the algorithm gives much better results using tournament selection. The target value now is 256 and the results of the algorithm with the roulette wheel selection are extremely low. But even the results of the algorithm using tournament selection are

Table 5: Search for a de Bruijn sequence with 256 positions. The maximal value of the fitness function is 256.

Exp. No.	Roulette selection	Tournament selection
1	195	236
2	198	233
3	196	234
4	196	238
5	193	236
6	197	236
7	193	240
8	197	236
9	194	240
10	195	235
Average	195.4	236.4
Stand. deviation	0.823	0.972

well below the target value. Additionally, from the course of the computation, it was apparent that the algorithm exhibited symptoms of stagnation. The best-attained values did not even change when the parameters n_p and n_g were increased.

The behaviour of the genetic algorithm was completely different in the situation when the task was to identify one specific in advance chosen de Bruijn sequence. In this case, the specific target de Bruijn sequence was always identified without problems, even with both the selection mechanisms.

GENETIC ALGORITHM PARADOX

In general, we face two diametrically different sorts of behaviour of the genetic algorithm. When in the first case we look for the specific in advance given de Bruijn sequence, the genetic algorithm behaves efficiently and finds the solution without any problems up to $\kappa = 10$ inducing $\lambda = 1024$.

On the other hand, in the second task, we look for any of the de Bruijn sequences out of the number $2^{(2^{\kappa-1})}$, the algorithm solves efficiently only relatively small tasks corresponding to $\kappa = 4$ and $\kappa = 5$ corresponding to $\lambda = 16$ and $\lambda = 32$. For bigger tasks $\kappa = 6$ and $\kappa = 7$ corresponding to $\lambda = 64$ and $\lambda = 128$ the algorithm becomes inefficient and without special modifications (e.g. tournament selection) does not attain the maximum of the fitness function. Using the genetic algorithm for tasks with values of $\kappa >= 8$ corresponding to $\lambda >= 256$ in these circumstances seems rather questionable even with special modifications - see Table 5. We recorded obvious stagnation of the algorithm well below the target value. It is also apparent that the situation gets worse with the increasing length of the de Bruijn sequence.

The reasons standing behind this curious behaviour are of great importance. We came to the conclusion that the source of this behaviour is the multimodality of the fitness function. When we consider a multimodal task, the usual

assumption of the genetic algorithm that two prospective potential solutions combine by the cross-over operation to other prospective potential solutions fails. Combining these individuals can lead to individuals with much worse value of the fitness function. This fact finally leads to the stagnation of the genetic algorithm far from the maxima of the fitness function.

CONCLUSIONS

In this article, we handle the topic of convergence of the genetic algorithm to the maximum of the fitness function. This theme is important from theoretical as well as practical reasons. If a scientist considers using the genetic algorithm, he should have a concrete idea regarding what he can expect. The topic is even more urgent since genetic algorithms are relatively demanding concerning computational resources.

In the next part, we describe the identification of de Bruijn sequences by the genetic algorithm. We present a limited selection of numerical results from the large pool of experimental data - see Tables 3, 4, and 5. We come to an interesting result that for the genetic algorithm is much easier to find the fixed de Bruijn sequence than to identify an arbitrary de Bruijn sequence. This leads to the conclusion that the genetic algorithm is not suitable for the optimization of multimodal tasks without any further modifications of the algorithm or the fitness function. The reason for this pathological behaviour are the fluctuations of the algorithm between maxima of the fitness function caused by the cross-over feature of the algorithm. These fluctuations finally lead to the stagnation of the algorithm far from maxima of the fitness function.

There are three principal points ensuing from our investigations concerning the binary optimizations.

1. In general, the standard genetic algorithm can be used as a robust solver providing solutions to diverse intricate binary optimization tasks. Nevertheless, the quality of the solution depends considerably on the modality of the fitness function of the task.

2. For unimodal fitness functions the genetic algorithm can be considered a relatively good optimizer providing good results and converging relatively reliably to the maximum of the fitness function.

3. For multimodal fitness functions the situation is different. The genetic algorithm can be negatively influenced by many fitness function maxima. The convergence process can stagnate because of the algorithm fluctuations between fitness function maxima.

Since we tested the genetic algorithm for this article exclusively on the tasks concerning the de Bruijn sequences identification, future research should continue and focus on other binary and also non-binary optimization tasks.

References

N. G. de Bruijn. A combinatorial problem. *Indag. Math. (N.S.)*, 49:758–764, 1946.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 2000.

L. Levine. Algebraic combinatorics. Lecture Notes, 21 18.312, Cornell University, Ithaca, NY, US, 2011.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Springer-Verlag Berlin Heidelberg New York, 1999.

G. Mitsuo and Ch. Runwei. *Genetic Algorithms and Engineering Optimization*. Joh Wiley Sons, Inc., 605 Third Avenue, New York, 2000.

W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill Education, London, 1976.

J. Sawada, A. Williams, and D. Wong. A surprisingly simple de Bruijn sequence construction. *Discrete Mathematics*, 339:127–131, 2016.

D. Simon. *Evolutionary Optimization Algorithms, Biologically-Inspired and Population-Based Approaches to Computer Intelligence*. Willey, Published by John Wiley Sons, Inc., Hoboken, New Jersey, 2013.

AUTHOR BIOGRAPHIES

ROMAN KNOBLOCH was born in Turnov, Czechoslovakia. After high school, he studied mathematics and physics at the Faculty of Mathematics and Physics, Charles University in Prague. He is occupied as an assistant professor at the Department of Mathematics of the Technical University in Liberec, the Czech Republic. His principal areas of interest are: mathematical modeling, up-to-date optimization techniques, continuum mechanics, and bit fields generation. His e-mail address is: roman.knobloch@tul.cz
ORCID: 0000-0003-2427-3479

JAROSLAV MLYNEK was born in Trnava, Czechoslovakia and he studied numerical mathematics at Charles University in Prague at the Faculty of Mathematics and Physics. His professional activities are focused on the study of convergence of evolutionary algorithms, especially differential algorithms. He also deals with the optimization of fiber winding on polymer composite frames and the optimization of heating of thin-walled metal molds in the production of artificial leather. He currently holds the position of an associate professor at the Technical University of Liberec. His e-mail address is: jaroslav.mlynek@tul.cz
ORCID: 0000-0002-3386-6738