BioSystems 150 (2016) 61–72

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

The role of crossover operator in evolutionary-based approach to the problem of genetic code optimization

Paweł Błażej, Małgorzata Wnętrzak, Paweł Mackiewicz*

Department of Genomics, Faculty of Biotechnology, University of Wrocław, ul. Joliot-Curie 14a, Wrocław, Poland

ARTICLE INFO

Article history:
Received 16 March 2016
Received in revised form 20 May 2016
Accepted 11 August 2016
Available online 20 August 2016

Keywords:
Amino acid
Crossover
Evolutionary algorithms
Genetic code
Mutation
Polarity

ABSTRACT

One of theories explaining the present structure of canonical genetic code assumes that it was optimized to minimize harmful effects of amino acid replacements resulting from nucleotide substitutions and translational errors. A way to testify this concept is to find the optimal code under given criteria and compare it with the canonical genetic code. Unfortunately, the huge number of possible alternatives makes it impossible to find the optimal code using exhaustive methods in sensible time. Therefore, heuristic methods should be applied to search the space of possible solutions. Evolutionary algorithms (EA) seem to be ones of such promising approaches. This class of methods is founded both on mutation and crossover operators, which are responsible for creating and maintaining the diversity of candidate solutions. These operators possess dissimilar characteristics and consequently play different roles in the process of finding the best solutions under given criteria. Therefore, the effective searching for the potential solutions can be improved by applying both of them, especially when these operators are devised specifically for a given problem. To study this subject, we analyze the effectiveness of algorithms for various combinations of mutation and crossover probabilities under three models of the genetic code assuming different restrictions on its structure. To achieve that, we adapt the position based crossover operator for the most restricted model and develop a new type of crossover operator for the more general models. The applied fitness function describes costs of amino acid replacement regarding their polarity. Our results indicate that the usage of crossover operators can significantly improve the quality of the solutions. Moreover, the simulations with the crossover operator optimize the fitness function in the smaller number of generations than simulations without this operator. The optimal genetic codes without restrictions on their structure minimize the costs about 2.7 times better than the canonical genetic code. Interestingly, the optimal codes are dominated by amino acids characterized by polarity close to its average value for all amino acids.

1. Introduction

It is worth mentioning that if we take into account the structure of the canonical genetic code with 61 possible codons encoding 20 amino acids and three stop translation codons, then we obtain a huge number of potential alternatives, about 1.51×10^{84}. It makes the question about the 'frozen' canonical genetic code among such enormous number of other possibilities very intriguing (Crick, 1968). There are three main theories trying to explain the origin and structure of the genetic code (see DiGiulio, 2005 for detailed review). However, none of them is unambiguously supported.

The first theory, called stereo-chemical, claims that some structural relationships and interactions between coded amino acids and stretches of RNA (e.g., codons, anticodons and reversed codons) (Dunnill, 1966; Pelc and Welton, 1966) were responsible for the present structure of the genetic code. So far, well confirmed such relationships were found for seven amino acids (see for review: Yarus et al., 2005). According to the physico-chemical (adaptive) theory (Freeland and Hurst, 1998; Gilis et al., 2001; Freeland et al., 2003), the canonical genetic code is optimized to minimize deleterious effects of mutations and errors occurring during protein synthesis (translation). The level of its adaptation can be measured by harmful effects of the replacement of one amino acid to another (Haig and Hurst, 1991). The coevolution hypothesis states that codons in the ancestral genetic code encoded only a small subset of amino acids and later, along with the evolution of biochemical organization of primary cells, newly synthesized amino acids took

* Corresponding author.
E-mail address: pamac@smorfland.uni.wroc.pl (P. Mackiewicz).

http://dx.doi.org/10.1016/j.biosystems.2016.08.008
0303-2647/

over the codons from the amino acids to which they were related in the biosynthetic pathways (Wong, 1975, 2005; Taylor and Coates, 1989; Di Giulio, 1991, 1989, 2016). Since the newly emerged amino acids as well as the taken codons were similar to their precursors this concept also explains why the genetic code can reflect an optimization in respect to translational errors.

The problem of genetic code optimization was investigated by many authors using two approaches: the statistical one (Freeland et al., 2000; Mackiewicz et al., 2008), which compares the canonical genetic code with many randomly generated alternatives, and the engineering method (Di Giulio, 2000), which compares the canonical code with the computationally optimized alternative. However, the large number of possible genetic codes makes it difficult to search the space of potential genetic codes. Therefore, the idea of applying adapted evolutionary-based algorithms (EA) seems very useful in solving this problem and is promising in a further research on general properties of the genetic code (Santos and Monteagudo, 2010, 2011). This proposal allowed for better location of the canonical genetic code in the fitness landscape and calculation of its distance to the optimized code.

The EA approaches are based on mutation and crossover operators. The mutation operator is indispensable in every evolutionary based algorithm because it is responsible mainly for introducing new information into the population of candidate solutions. The effectiveness of this algorithm can be improved by applying a crossover operator. This operator is used to create new individuals (offspring) based on existing solutions (parents). As a result, newly created individuals can often inherit good parts of their parents and therefore can be better and quicker adapted. It results from the fact that parent individuals are not random because they are examined by a selection process in the preceding simulation step. Consequently, the crossover mutation operators are jointly responsible for random changes in the population of candidate solutions and drive the computational evolution.

The different properties of these operators makes that each of them introduces its own variation. Therefore, it seems that the inclusion both of them should generally enhance the effectiveness of searching for possible solutions. However, potential benefits of using crossover operator depend on the kind of optimization problem (Fogel and Atmar, 1990; Spears, 1992, 1994; Park and Carter, 1995; Kokosiński, 2005). Thus, it is reasonable to test the influence of crossover operator on the effectiveness of evolutionary algorithm in every considered model and develop operators that are specific for a given problem.

Therefore, in this work, we adapted the position based crossover operator for two models of the genetic code and proposed a new operator for another model. We studied the performance of the algorithms for different combinations of mutation and crossover probabilities. Based on this large item of data, we were able to test with statistical significance the potential impact of these parameters' values on the quality of the optimization process. Thanks to this extensive search we were also able to evaluate the most optimal genetic codes found in these simulations and compare them with the canonical one.

2. Methods

2.1. Mutation and crossover operators

In the previous attempts to solve the problem of the genetic code optimality, different types of mutation operators were used (see Santos and Monteagudo, 2010, 2011 for details). Their usage depended on restrictions on the genetic code structure. However, the authors did not use any type of crossover operator. Furthermore, they emphasized that the classical crossover operators do not guarantee that all amino acids are always represented in the derived genetic codes (offspring) (Santos and Monteagudo, 2010). To deal with this problem, we adapted an already known crossover operator and also proposed a new one. We tested their quality under three restrictions (models) in searching the space of genetic codes:

1. Canonical structure 1 (CS1), which preserves the characteristic structure of codon blocks and degeneracy of the canonical genetic code. To generate potential codes, we permuted the assignment of amino acids between the codon blocks.
2. Canonical structure 2 (CS2), which preserves the number of codons per amino acid as in the canonical code. To generate potential codes, we permuted the assignment of codons to amino acids disregarding the codon blocks structure. By comparing results obtained for CS2 and CS1, we can test the importance of the characteristic codon blocks' structure with maintained degeneracy of the canonical genetic code.
3. Unrestricted structure (US), which has no constraints on the genetic code structure but assumes that every amino acid should be coded by at least one codon. To generate potential codes, we randomly divided 61 codons into 20 non-overlapping sets.

For all the described models, we claimed that stop codons remained invariant during all simulations and stayed the same as in the canonical code.

In the case of CS1, we adapted the position based crossover (POS) operator (Syswerda, 1991). A similar procedure is used in an evolutionary-based approach to the travelling salesman problem (Larrañaga et al., 1999). The POS draws amino acids from the parental codes at random and assigns them to the corresponding codon blocks in the offspring (Fig. 1A). The remaining codon blocks have amino acids assigned in the order of the other parent. When an amino acid is already present in the offspring, the other one is selected according to its position in the vector of amino acids (Fig. 1B). It ensures that every amino acid in the offspring is assigned only to one codon block.

However, this operator cannot be directly used in the CS2 and US models because the possible offspring might not inherit the proper structure of its parents. In this case, the generated genetic codes might not code all 20 amino acids. Therefore, we had to introduce another version of crossover operator (Fig. 2), according to the following procedure:

1. We create offspring O_1 and O_2, which are identical to their parents P_1 and P_2.
2. We select randomly an amino acid a_i, the same for the two parents, coded by parental codon blocks C_1 and C_2, respectively.
3. We compare the blocks and recognize the set of codons present in both parents, i.e., $U = C_1 \cap C_2$ as well as sets of codons present in one parent and absent in the other, i.e., $S_1 = C_1 \setminus U$ and $S_2 = C_2 \setminus U$ such that the condition $S_1 \cap S_2 = \emptyset$ is fulfilled.
4. The codons that are the same in the two parental codon blocks, i.e., $c_i \in U$, are not exchanged (Fig. 2A).
5. In the case of the sets S_1 and S_2, we choose at random codons $c_i \in S_1$ and $c_j \in S_2$ and exchange them between offspring O_1 and O_2 (Fig. 2B). To keep the original set of all codons represented by only one item, the codon exchange is realized by the swap of corresponding codons within a given offspring. Thanks to that, the individual that donated a codon does not loose it, whereas the offspring obtaining the codon has it assigned only once. The exchanged codons c_i and c_j are then removed from S_1 and S_2. This procedure is repeated until there are no codons left in S_1 or S_2 for selection.
6. When, for example, $S_1 = \emptyset$ and $S_2 \neq \emptyset$, there are no codons for mutual exchange. Then a codon, here $c_j \in S_2$, is moved to the

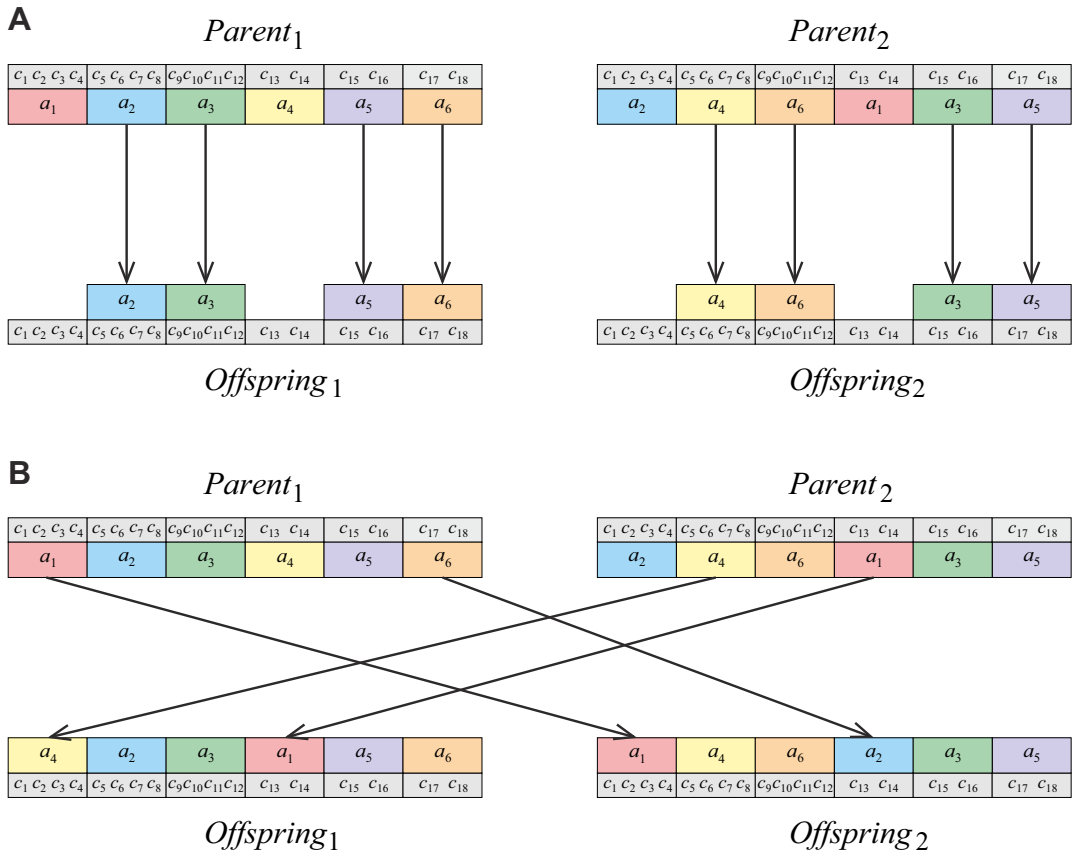

Fig. 1. The schema of the crossover operator for the CS1 model. A. Some amino acids (a_i) are randomly chosen from parents and assigned to blocks consisting of codons (c_i) in the corresponding offspring. B. The still empty codon blocks inherit the remaining amino acids from the second parent according to their order in its code.

offspring O_1 and deleted from the offspring O_2 (Fig. 2C). To maintain the original set of all codons without repetitions, the obtained codon, here by the offspring O_1, is taken from an amino acid of this individual in which it is coded by this codon, whereas in the offspring O_2, the codon is shifted to the same amino acid in its own set. Thereby, it is possible to change the number of codons for amino acids. This exchange is not realized when an amino acid from which a codon should be taken is coded by only one codon (Fig. 2D).

In our simulations, we applied various mutation operators in dependence on the used model. In the case of CS1, the mutation is realized by a random selection of two different amino acids and an interchange of their codon groups. In the CS2 model, we used a swap operator that interchanges randomly selected codons between their original amino acids. In the case of the US model, the mutation process is realized by two operators. The first one chooses one codon at random and assigns it, if it is possible, to another randomly selected amino acid. The second mutation operator is the same as the mutation operator used in the CS2 model. They are both used at random with the joint probability of mutation equal to a fixed value. We applied two operators in the US model because the usage of only the first one did not guarantee a satisfactory convergence to the final solution.

2.2. Fitness function

As it was mentioned, we considered three restrictions on the genetic code structure and searched for the optimized possible alternative that minimizes a fitness function. Similarly to other authors (Haig and Hurst, 1991; Santos and Monteagudo, 2010, 2011), we used the following fitness function:

$$F = \sum_{\langle i,j\rangle \in D} [p(i) - p(j)]^2,$$

where D is a set of pairs of codons $\langle i, j\rangle$ which differ in one codon position, whereas $p(i)$ and $p(j)$ are the polarity values of amino acids coded by the codons i and j, respectively. We chose this amino acid property (Woese, 1973) because it was shown that the standard genetic code has achieved 68%-minimization of this property (Di Giulio, 1989). In other words, F is the total of squared differences between the polarity values of amino acids coded by their original codons and the ones coded by mutated codons different in one codon position from the original one. Thereby, the fitness function considers costs of all possible single-point mutations between codons. Mutations involving stop codons were ignored. The aim of the optimization procedure was to minimize this function in order to find such assignment of codons to amino acids that minimizes the costs of amino acid replacements.

2.3. Simulation procedure

For each model of the genetic code, we tested nine probabilities of mutations (0.1, 0.2, . . ., 0.9) together with 10 probabilities of crossover (0, 0.1, 0.2, . . ., 0.9). Each type of simulation was run up to 1000 generation steps and repeated 100 times with different seeds and 280 initial codes randomly generated. Thus, it gave in total 9000 simulation runs for one model. An example of variation in the fitness function calculated from 100 independent simulations under the same parameters and different seeds is presented in Fig. 3. The

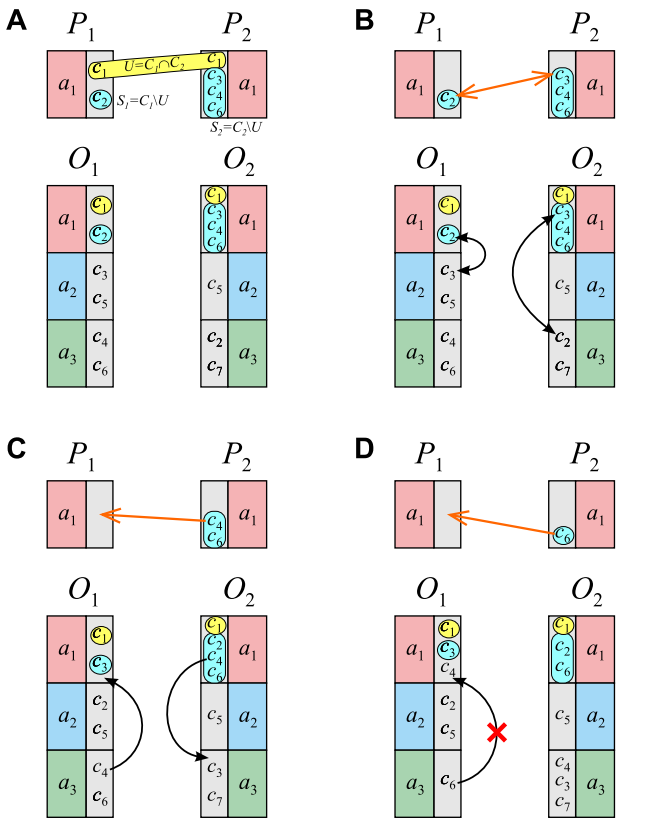

Fig. 2. The schema of crossover operator for the US model. A. Codons (marked in yellow) shared by two parents P_1 and P_2 are not exchanged, whereas those present in one parent but absent in another (marked in blue) will be subjected to swap. B. A mutual exchange of codons (orange arrow) between offspring O_1 and O_2 is realized by a swap of corresponding codons in the individuals (black arrows). C. A shift of codon from one individual to another (orange arrow) in the case when all codons were already exchanged is accomplished by a movement of corresponding codons in the individuals (black arrows). D. The example in which the shift of a codon cannot be carried out because the codon to be shifted in the offspring 1 is the only codon for its amino acid. (For interpretation of the references to color in this legend, the reader is referred to the web version of the article.)

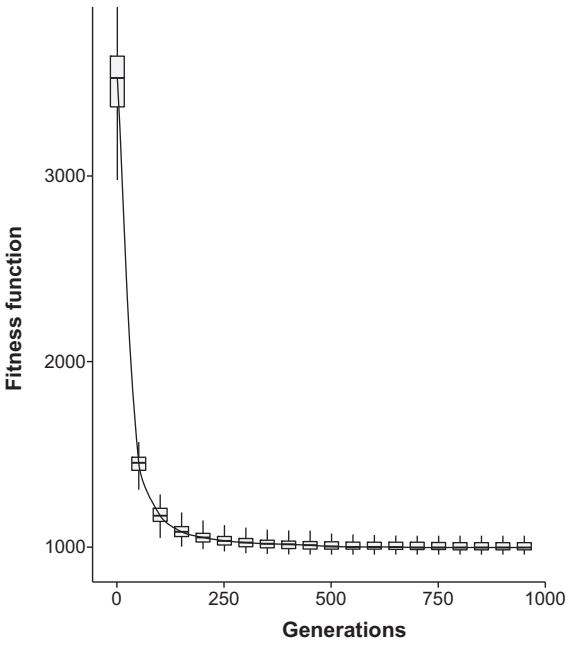

Fig. 3. An example of the variation in the fitness function F under the US model with the number of generations, calculated from 100 independent simulation runs with different initial seeds. The mutation and crossover probabilities are equal to 0.5. The thick line indicates median, the box shows quartile range and the whiskers denote the range without outliers.

F shows the decreasing variation during simulation and converges to very similar values in the independent simulations.

The huge amount of data allowed us to perform statistical analysis of the stochastic processes and evaluate effectiveness of tested operators and parameters in finding optimal solutions. To detected a general tendency of the observed processes with the 0.95 confidence interval, we used general additive model (GAM) approximation method (Wood, 2006) (Fig. 4).

2.4. Performance measures

Besides the fitness function, we considered also the measure S to characterize the efficiency of carried out simulations and algorithms. This measure describes the mean time t (measured by the number of generations) to reach the "nearly" steady state of the simulation runs, i.e., when the minimum value of the fitness function $F_{min}(t)$ varies no more than 1% of the $F_{min}(T)$, where T is the final simulation step = 1000 generations.

We also calculated improvements of the fitness function resulting from the application of the crossover operator in simulations with fixed values of mutation probability. This parameter was expressed as a percentage difference between the values of the fitness function F_{Cr} obtained in a simulation with crossover and the values of the fitness function F_{nocr} calculated for a simulation when only the mutation operator was applied:

$$imp_F = 100\% \cdot \frac{F_{nocr} - F_{Cr}}{F_{nocr}}.$$

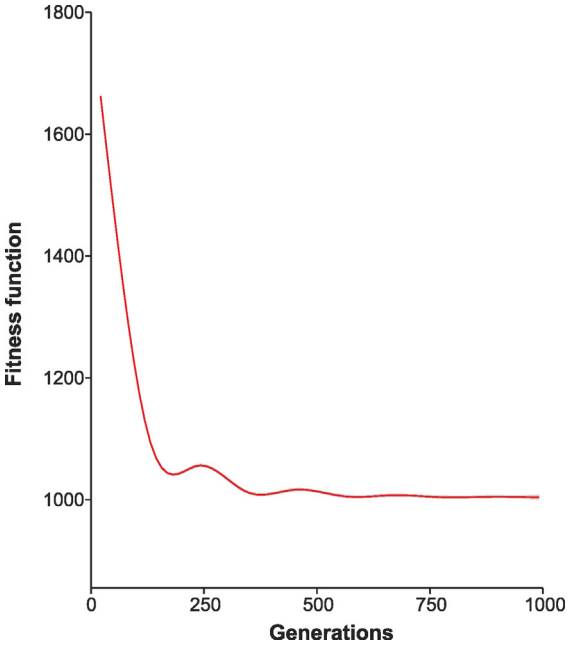

Fig. 4. Changes in the best approximation of the fitness function F with the number of generations, based on the GAM model and computed from 100 simulations with different initial seeds under the US model. The mutation and crossover probabilities are equal to 0.5. The approximation is characterized by a very narrow 0.95 confidence interval depicted by the gray border, which may not be clearly visible in this scale.

Similarly, we introduced an improvement of the time to reach the "nearly" steady state given by the following equation:

$$imp_S = 100\% \cdot \frac{S_{nocr} - S_{Cr}}{S_{nocr}},$$

where S_{Cr} is the mean time to reach the "nearly" steady state calculated from all simulation runs with a given crossover value, whereas S_{nocr} is the mean time for simulations without crossover.

To check the potential statistical significance of differences in the values of the proposed measures between simulations with different combination of mutation and crossover probabilities, we performed the non-parametric Kruskal–Wallis test (KW) using R package (R Core Team, 2015). Depending on the outcome of the test, we did also the Dunn *post hoc* test, which is a pairwise multiple comparison procedure appropriate to follow the rejection of the KW test results. The resulted *p*-values were adjusted by the Holm's method. Other statistical analyses were carried out in Statistica software (StatSoft, 2014).

3. Results and discussion

3.1. Influence of crossover operator on fitness function

At first, we tested if simulations with different probabilities of crossover and the fixed probability of mutation converge to similar values of the fitness function *F* after 1000 generation steps. An example for the CS1 model is shown in Fig. 5. It is clearly visible that the final value of *F* depends on the applied crossover probability and it is best minimized in simulations with the largest applied crossover probability (0.9), whereas the exclusion of this operator from the algorithm results in the worst optimization.

To visualize the influence of crossover on the function *F*, we calculated the mean values of this function for all combinations of mutation and crossover probabilities, and plotted them as a heatmap (Fig. 6). It is clear that the larger probabilities of both mutation and crossover operators cause better optimization of the fitness function.

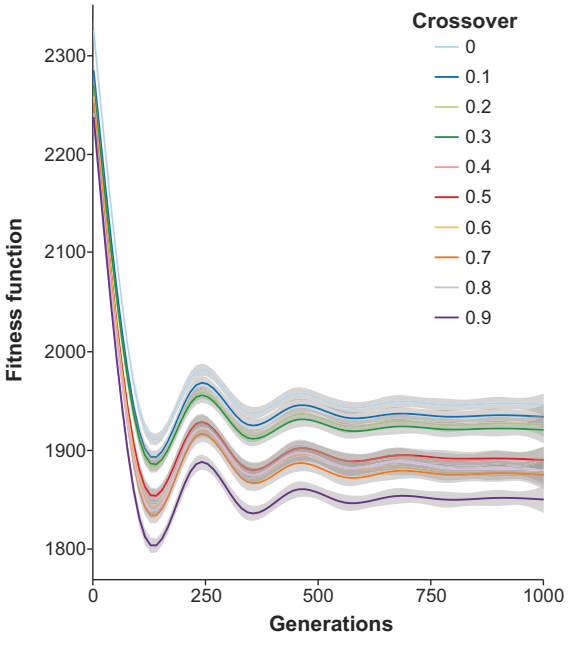

Fig. 5. Changes in the fitness function *F* of genetic codes under the CS1 model with the number of generations for different crossover probabilities. The mutation probability equals 0.2. It is clearly visible that the fitness function is best minimized in the simulation with the largest crossover probability = 0.9. Grey borders show the 0.95 confidence interval.

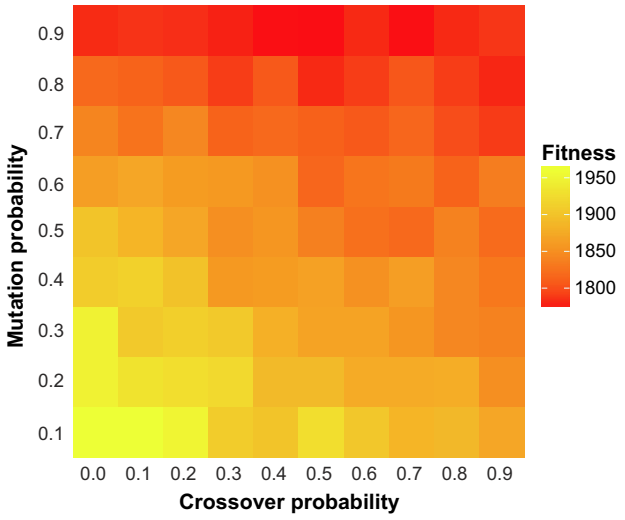

Fig. 6. Values of the fitness function *F* for all combinations of considered mutation and crossover probabilities. The values of *F* were averaged over 100 runs at the end of simulations for the CS1 model.

Sections through the heat-map are presented in Fig. 7. They confirm the significant decrease of the fitness function with the growth of the mutation probability. The positive influence of the crossover operator on the minimization is also visible. Calculated correlation coefficients for these relationships were significant and negative, similarly to the slopes of fitted linear functions (Table 1). The only exception is the simulation with the largest mutation probability = 0.9, for which the relationship is not significant. It results most probably from the domination of the mutation operator over the advantageous contribution of crossover in finding the optimal solution. In this case, the generation of new potential solutions by the mutation operator is so intensive that it decreases the impact of crossover.

We tested the significance of the differences between 10 groups of simulation results with various values of crossover probability and the fixed mutation probabilities using the Kruskal–Wallis test. The test rejected the null hypothesis (with *p*-value <0.001) about the similarity of the distribution of the fitness function values in these groups. Furthermore, the Dunn *post hoc* test applied to this data revealed statistically significant differences (*p*-value <0.05) in 354 pairwise comparisons (out of 405) of these groups. In particular, for the fixed mutation probabilities, the values of the function *F* were significantly smaller in 72 out of 81 simulations with applied crossover operator than without it. In only four comparisons, the simulations without crossover produced significantly smaller values of the fitness function. These results indicate a considerable influence of the crossover operator on the optimization of the function *F*.

Table 1
Slope and correlation coefficient (*r*) of approximated linear functions between the fitness function and the crossover probability for the respective mutation probabilities (mut) under the CS1 model. Significance levels of *p*-value for *r* are indicated by: * for $p < 0.05$, ** for $p < 0.01$ and *** for $p < 0.001$.

Mut	Slope	*r*
0.1	−100.6	−0.92***
0.2	−99.0	−0.97***
0.3	−110.2	−0.97***
0.4	−83.0	−0.91***
0.5	−86.9	−0.92***
0.6	−55.9	−0.82**
0.7	−50.4	−0.89***
0.8	−32.0	−0.72*
0.9	−2.0	−0.14

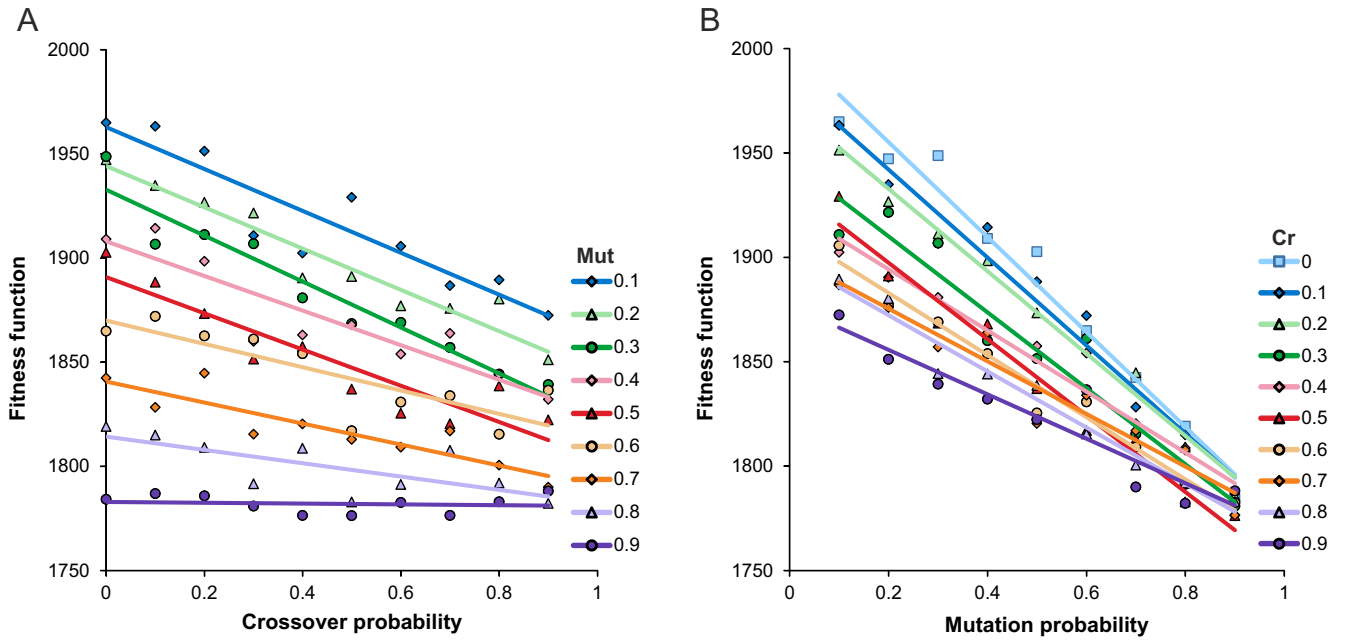

Fig. 7. Relationship of the average values of the fitness function with the probability of crossover (A) and mutation (B) for different values of mutation (Mut) and crossover (Cr), respectively, under the CS1 model. The decrease in the fitness function with the growth of crossover and mutation probabilities is visible.

Table 2
The largest improvement of the fitness function (imp_F) averaged over 100 simulation runs obtained for the fixed mutation probability (mut) and given crossover (Cr) under the CS1 model.

Mut	Cr	imp_F [%]
0.1	0.9	4.7
0.2	0.9	4.9
0.3	0.9	5.6
0.4	0.9	4.0
0.5	0.7	4.3
0.6	0.8	2.7
0.7	0.9	2.8
0.8	0.9	2.0
0.9	0.5	0.4

This influence is also supported by the maximal percentage improvement of the fitness function (imp_F) resulting from the applied crossover operator, calculated for each fixed mutation probability (Table 2). The greater probability of crossover (Cr >0.7) is required to minimize the function better. However, its contribution is masked when the greater mutation probability is applied. There is a significant negative correlation between the mutation probability and the largest improvement of fitness function (Pearson correlation coefficient, $r = -0.93$, p-value = 0.0003).

In contrast to the CS1 model, there were no statistically significant differences (p-value >0.05) in the CS2 and US models between distributions of the F function for simulations with different values of crossover probabilities. It suggests that the algorithm with and without crossover stabilizes around similar values at the end of simulations. However, it should be emphasized that the crossover was not responsible for any kind of deterioration of the final results in these cases and the fitness function was always substantially minimized (Fig. 8).

3.2. Influence of crossover operator on convergence to stable solutions

Although we did not find any significant relationship between the crossover probability and the fitness function values for the CS2 and US models, the application of crossover influenced another important performance measure, which describes convergence to stable solutions, i.e., the mean time to reach the "nearly" steady state S, measured by the number of simulation steps. The shortest mean time S for the CS2 model equalled 155 steps for simulations with the probability of crossover = 0.7 and mutation = 0.7. For simulations without crossover and the same mutation probability, this time was longer: 256 steps. For the US model, the shortest mean time, i.e., 314 steps, was for simulations with the probabilities of crossover = 0.8 and mutation = 0.3. For the corresponding simulations without crossover it was 438 steps. The longest mean times were found for simulations without crossover: 438 steps for the CS2 model (mutation probability = 0.1) and 751 for the US model (mutation probability = 0.9).

Moreover, we found other interesting relationships between S and the probabilities of mutation and crossover. The dependencies are different for the CS2 and US model (Fig. 9). In CS2, the values of S decline substantially with the simultaneous growth of the mutation and crossover probabilities (Fig 9A). In the US model, the values of S also fall monotonically with the increase in the crossover probability but considering the mutation operator, S reaches smaller values only for the mutation probability in the range 0.2–0.5 (Fig 9B). It is clearly visible in Fig. 10A, where the time to reach the steady state decreases linearly with the growth of the crossover probability for all values of the mutation probabilities. The larger probability of crossover, the shorter time to find the optimal solution. On the other hand, Fig. 10B demonstrates a non-linear relationship between the time S and the mutation probability for different values of the crossover probability. The quickest convergence to the steady state is reached for the mutation probabilities about 0.3, whereas the longest times are observed for the maximal mutation probability = 0.9. These results suggest that a customized set of parameters should be applied to speed up the convergence of the algorithm, especially in the case of the US model. Moreover, the increase in the mutation probability is not always beneficial. When the mutation probability is too large, the time to find the optimum gets longer because of too often destruction of promising solutions. Theoretically, the same could also concern the crossover but we observed that the increase in the crossover probability to 0.9 has an advantageous effect in this particular case studied by us.

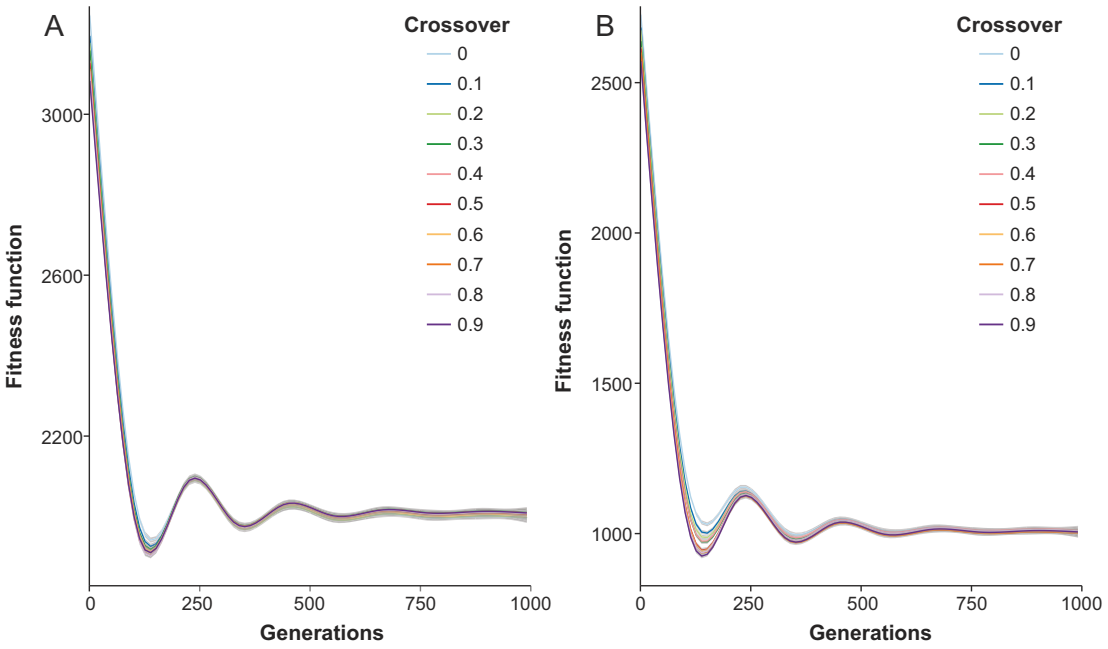

Fig. 8. Changes in the fitness function F of genetic codes under the CS2 (A) and US (B) models with the number of generations for different crossover probabilities. The mutation probability equals 0.5. Despite different values of the crossover probability, the fitness function converges to similar small values.

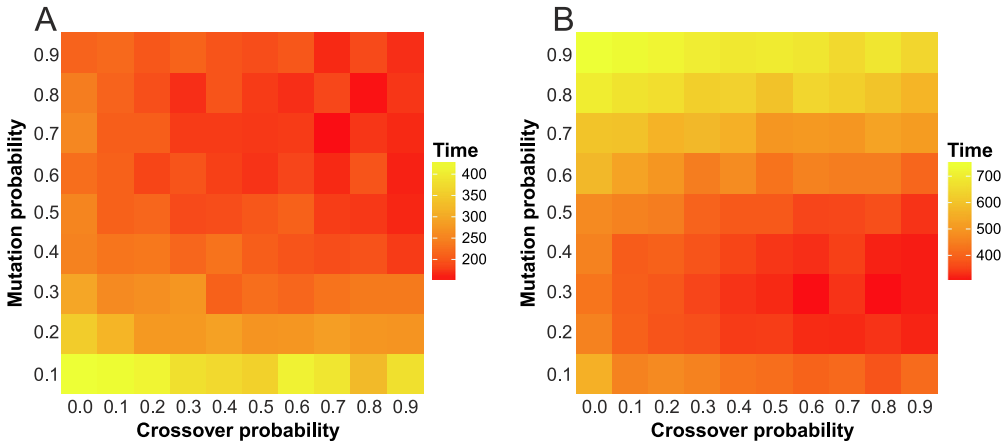

Fig. 9. Values of the mean time to reach the "nearly" steady state S for all combinations of considered values of crossover and mutation probabilities under the CS2 (A) and US (B) models. The values of S were averaged over 100 runs at the end of simulations.

The significant linear fall of the time S with the increase in the crossover probability was also found under the model CS2 in simulations with all mutation probability values (Table 3). The slopes of approximated linear functions are also negative and the correlation coefficients are high and significant. However, these relationships are less pronounced than those in the US model, in which the slopes are almost two times larger. In the case of the CS1 model, we did not observe such significant relationships although negative slopes were found in seven cases of different mutation probabilities.

The results show that the impact of crossover on shortening the time of finding the optimal solution increases with the complexity of considered task, from the CS1 and CS2 to US model. Correspondingly, for these models, there are 5.225×10^8, 5.559×10^{64} and 8.788×10^{78} candidate solutions. Finding the optimal solution is the most difficult and time-consuming in the US model, under which the advantageous influence of crossover is the most pronounced.

Table 3
Slope and correlation coefficient (r) of approximated linear functions between the time to reach the steady state and the crossover probability for the respective mutation probabilities (mut) under the CS2 and US models. Significance levels of p-value for r are indicated by: * for $p < 0.05$, ** for $p < 0.01$ and *** for $p < 0.001$.

Mut	CS1 model		US model	
	Slope	r	Slope	r
0.1	−72.9	−0.67*	−142.5	−0.85**
0.2	−62.3	−0.73*	−124.3	−0.89***
0.3	−62.2	−0.67*	−120.0	−0.9***
0.4	−78.3	−0.97***	−128.4	−0.9***
0.5	−71.0	−0.83**	−148.6	−0.95***
0.6	−47.9	−0.72*	−139.0	−0.86**
0.7	−80.0	−0.83**	−113.3	−0.86**
0.8	−65.9	−0.78**	−112.5	−0.87**
0.9	−51.6	−0.86**	−101.1	−0.95***

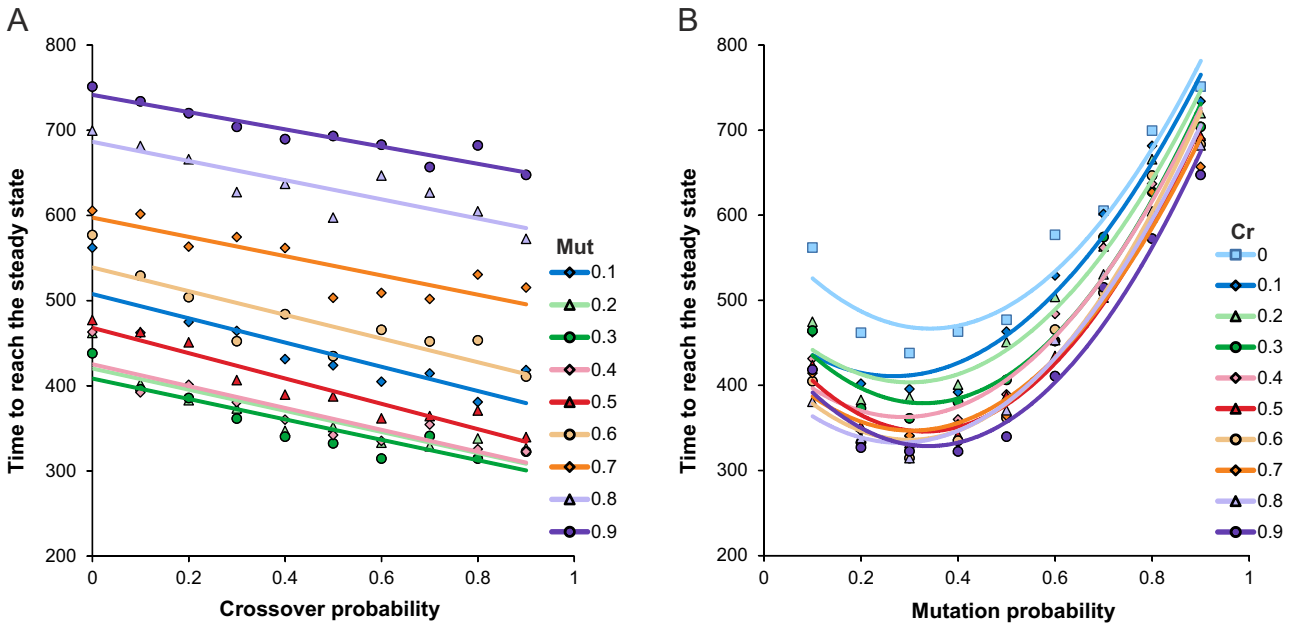

Fig. 10. Relationship of the averaged time to reach the steady state (measured in the number of simulation steps) with the probabilities of crossover (A) and mutation (B) for different values of mutation (Mut) and crossover (Cr), respectively, under the US model. The linear dependence of the time on the crossover and non-linear on the mutation operator are clearly visible.

Table 4
The largest improvement of the time to reach the "nearly" steady state (imp_S) averaged over 100 simulation runs obtained for the fixed mutation probability (mut) and given crossover (Cr) under the CS2 model.

Mut	Cr	imp_S [%]
0.1	0.8	25.0
0.2	0.5	22.6
0.3	0.4	29.2
0.4	0.9	30.2
0.5	0.9	35.3
0.6	0.9	27.9
0.7	0.7	39.4
0.8	0.8	35.4
0.9	0.7	23.1

To confirm statistically the properties of the S function, we compared the values of S between simulations with various crossover and fixed mutation probabilities using the Kruskal–Wallis test. In the CS2 and US models, the null hypothesis about similar distributions of the S function values between groups was rejected (p-value <0.001), which corresponds well with our previous results. Furthermore, the Dunn *post hoc* test applied to this data revealed substantial differences in pairwise comparisons of simulations with various crossover parameters (p-value <0.05). In particular, these results support our findings that there are statistically significant differences between simulations with no crossover and with crossover greater than 0.5 for all mutation probabilities. The simulations with greater probabilities of crossover showed shorter times to reach the steady state than the simulations with only mutation operator. This tendency is clearly visible in the box-plots of the S measure (Fig. 11).

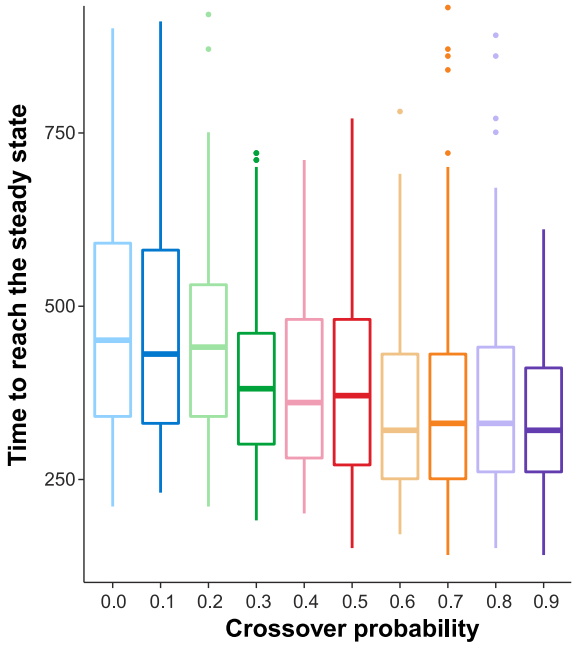

Fig. 11. Box-plots of the time to reach the "nearly" steady state S from simulations under the US model with the fixed value of mutation probability (0.5) and different probabilities of crossover. The thick horizontal line indicates median, the box shows the range between the first and third quartiles (IQR, the inter-quartile range) and the whiskers determine the range without outliers for the assumption 1.5 × IQR.

These results are in agreement with the calculated improvement of the time to reach the "nearly" steady state imp_S (Table 4, 5). The values for CS2 model are in the range 22–39%, which indicates that the application of the crossover operator substantially shortens the time of convergence to the "nearly" stable solution (Table 4). The largest improvements concern simulations with crossover probabilities greater than 0.5 and do not seem to depend on the mutational probability ($r = -0.379$, $p = 0.315$). However, in the case of the US model, there is a clear dependence between mutation probability and imp_S ($r = -0.876$, $p = 0.002$) (Table 5). For mutation probabilities less than 0.7, the improvement is about 30%, whereas for the greater probabilities, the imp_S declines to the range 13.8% to 18.1%. A considerable improvement is also observed for simulations with large crossover probabilities (≥ 0.7).

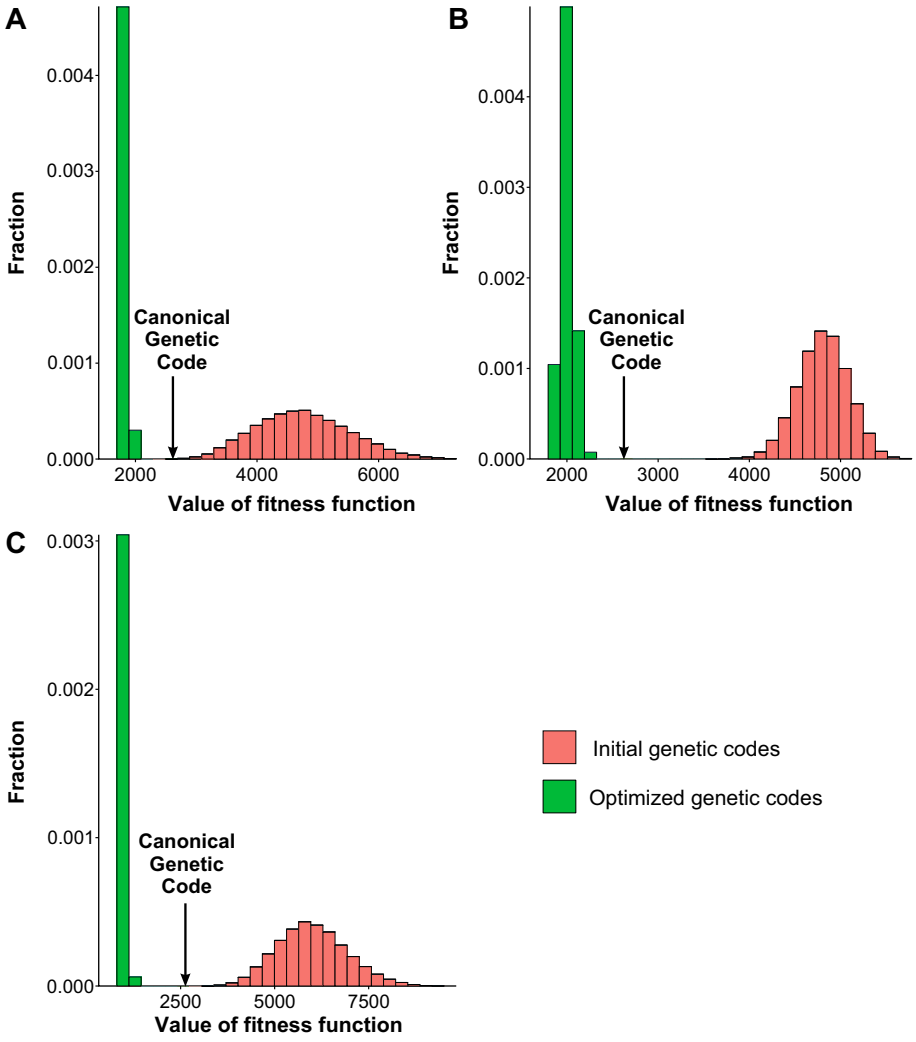

Fig. 12. Distribution of the fitness function values for initial (random) and the best optimized sets of genetic codes for three restrictions (models) of the genetic code: CS1 (A), CS2 (B) and US (C). The value of the canonical genetic code was also marked.

Table 5
The largest improvement of the time to reach the "nearly" steady state (imp_S) averaged over 100 simulation runs obtained for the fixed mutation probability (mut) and given crossover (Cr) under the US model.

Mut	Cr	imp_S [%]
0.1	0.8	32.2
0.2	0.9	29.2
0.3	0.8	28.2
0.4	0.9	30.3
0.5	0.9	28.8
0.6	0.9	28.7
0.7	0.7	17.1
0.8	0.9	18.1
0.9	0.9	13.8

Some influence of the crossover parameter on imp_S was also observed for the CS1 model. Considering the case with the smallest improvement of the fitness function, i.e., 0.4%, for the probability of mutation 0.9 and crossover 0.5 (Table 2) and comparing it to the simulation with the same mutation probability but without crossover, we found 33% improvement of the time to reach the "nearly" steady state.

Our results indicate that the application of crossover operators in evolutionary algorithms approach improves significantly the quality of solutions in the problem of finding the optimal genetic code. Their usefulness may, however, depend on the optimization problem (Kokosiński, 2005). For example, several authors (Fogel and Atmar, 1990; Spears, 1992, 1994) observed that the crossover is responsible for a premature convergence and a loss of population diversity in studied optimization problems, whereas Park and Carter (Park and Carter, 1995) pointed out that the influence of the crossover operator is negligible in many combinatorial problems. Searching the huge space of genetic codes is a sufficiently complex problem that the benefit of the crossover operator is apparent.

3.3. *The found optimal genetic codes*

The application of mutation and crossover operators with divers probabilities enabled us to effectively search the space of potential genetic codes for one that minimizes the costs of amino acid replacements. The distribution of the fitness function for the best optimized codes in comparison to the initial (random) codes and the canonical one are presented in Fig. 12 for three restrictions (models) imposed on the genetic code. The optimized codes are

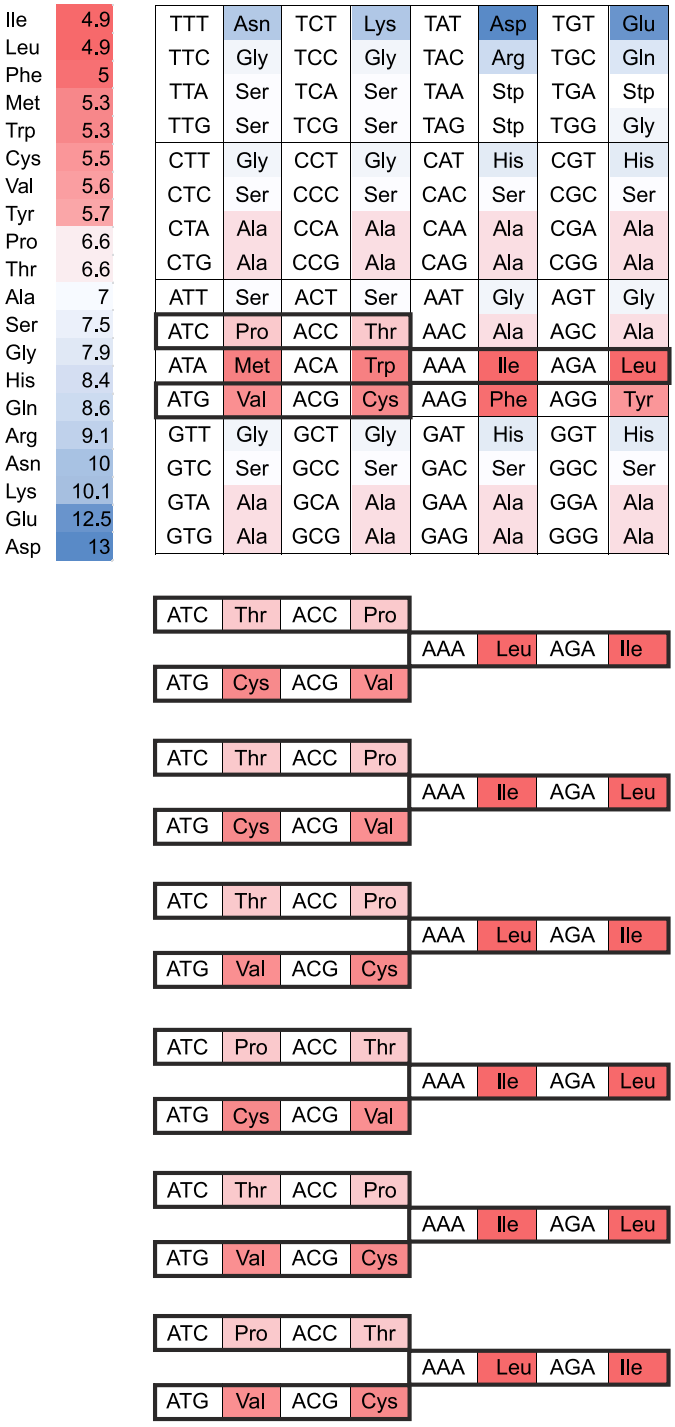

Fig. 13. The optimal genetic codes found under the least restrictive US model. The selected amino acids and their codons that differ in seven equivalent variants with the same smallest value of the fitness function were marked by bold rectangles. The areas of amino acids were colored according to their values in the polarity scale by Woese (1973) (shown on the left).

clearly shifted to the lower values of the fitness function. The distance between the distributions of the optimized and the initial codes is smallest for the most restricted model CS1 (409), larger (1331) for the model CS2 with bigger number of alternative codes and greatest (1813) for the most general model US. The value of the fitness function for the canonical genetic code is located between these distributions but more closer to the optimized codes.

The codes that decreased the cost of amino acid replacements to the greatest extent were found under the most general US model, which assumes the least restrictions on the genetic code structure. Under this model, we found seven codes with the same smallest value of the fitness function = 967. In the case of the CS1 and CS2 models the values were larger, i.e., 1758 and 1904, respectively.

The average value for randomly generated codes under the US model at the beginning of the simulations was 6069 ± 917 standard deviation, which indicates more than six-fold reduction of the fitness function value in the optimal codes. For comparison, the value of F for the canonical genetic code is 2623, which means that the real code minimizes the costs of amino acid replacements 2.7 times worse than the optimized solutions.

Another way to determine the optimization level of the genetic code is the percentage of distance minimization (pdm). This measure was defined by Di Giulio (Di Giulio, 1989) as: $pdm = 100 \times (f_{mean} - f_{code})/(f_{mean} - f_{optimal})$, where f_{mean} is the estimated average value of the fitness function for random codes, f_{code} is the value for the canonical genetic code and $f_{optimal}$ is the value for the found optimal solution. The pdm is interpreted as the optimization of the canonical genetic code in relation to the randomized mean code and the best optimized code. The larger values indicate a similar distance of the canonical code and the optimal one from the random codes. Our simulations showed that the pdm for the optimal codes in the US model is 68%, whereas for CS1 and CS2, 71% and 75%, respectively. If the genetic code was optimal, its pdm would be 100%, whereas if the value for the optimal code approaches zero (indicating no costs of replacements), the pdm would approach 57% in the US model and 45% in the CS1 and CS2 models.

Our results indicate that it is quite easy to improve significantly the optimization properties of the canonical genetic code when restrictions imposed on the code structure are substantially relaxed (here we only assumed the fixed position of stop translation codons). This conclusion is in agreement with studies that also included less restrictive assumptions on the code structure and states that the genetic code is far from being optimal and represents only a locally optimized solution (Novozhilov et al., 2007; Santos and Monteagudo, 2010, 2011).

The found optimal codes are presented in Fig. 13. There are only three pairs of positions in these codes that differ between them. The following amino acid pairs occur interchangeably in these positions: proline and threonine, valine and cysteine as well as isoleucine and leucine. The differences have no influence on the fitness function of these codes because the amino acids in pairs have the same or very similar polarity values. Therefore, the codes are equivalent in this respect. Interestingly, these codes are dominated by alanine, which occupies 18 codons and serine, which is coded by 14 codons. The next frequent amino acids in the codes are glycine with 9 codons and histidine with 4 codons. The others are encoded by only one codon. In the canonical genetic code, these amino acids are coded by 4 (Ala and Gly), 6 (Ser) or 2 (His) codons. The amino acids expanded in the optimal codes are characterized by polarity values close to the average value of all amino acids (Fig. 14). The Spearman correlation coefficient between the number of codons for a given amino acid in the optimal codes and the absolute difference between the polarity value of the amino acid and the average polarity equals -0.657 and is statistically significant ($p = 0.0017$). The domination of the amino acids with the average polarity value in the optimal codes minimizes the costs of replacements between their codons and others that code for amino acids with extreme polarity values.

Although we found that the optimal codes show a substantially different structure from the canonical one, it does not exclude that the canonical code evolved under some constraints on the minimization of mutational and translational errors. However, its structure was imposed rather by the initial assignment of amino

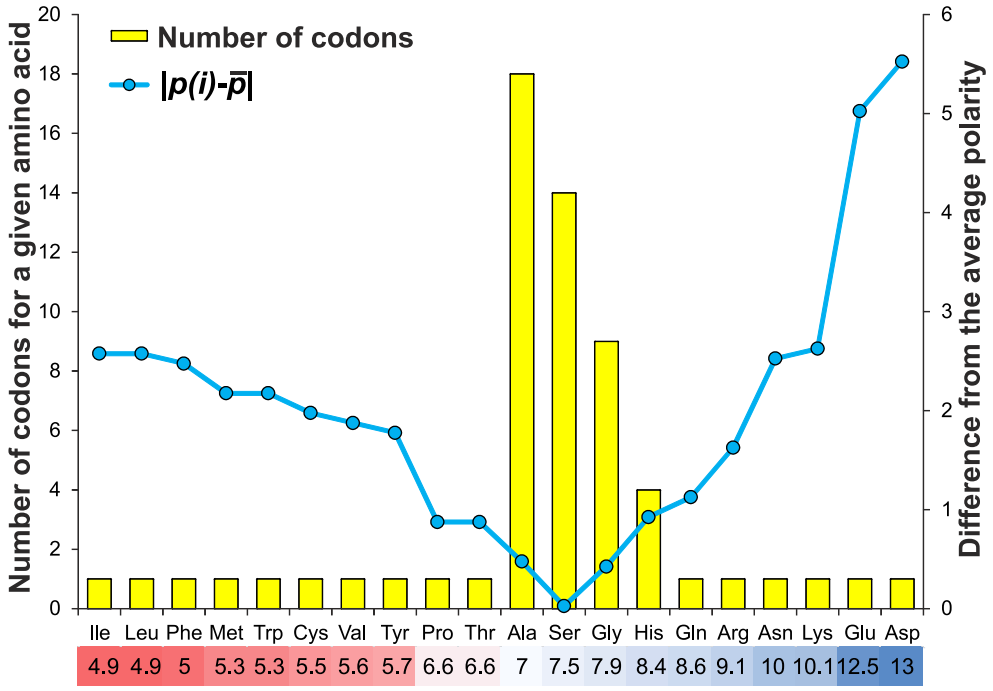

Fig. 14. The number of codons for respective amino acids in the found optimal genetic codes compared with the absolute difference in the polarity value for the given amino acid and the average value of all amino acids: $|p(i) - \bar{p}|$. The amino acids were arranged according to the polarity scale. It is visible that the amino acids coded by the largest number of codons have polarity close to the average value.

acids to codons and subsequent extension of the code by other amino acids, according to the coevolution theory (DiGiulio, 2005), which makes it only locally optimized (Novozhilov et al., 2007; Santos and Monteagudo, 2010, 2011).

4. Conclusions

The idea of searching for the optimal genetic code using the evolutionary-based algorithm initiated by Santos and Monteagudo (2010, 2011) is very promising and opens new possibilities in studying the properties of the genetic code. Due to the flexibility of the algorithm, it can be modified to achieve better performance by introducing various genetic operators whose advantageous role is not always clear. Therefore, we tested the influence of the crossover operator on effectiveness of the algorithm in the problem of optimization of the genetic code. Our results indicate that application of this operator improved significantly the optimization process, especially in tasks with larger space of potential solutions. The simulations with the crossover were characterized by the smaller values of the fitness function and reached the stable solution in shorter time than simulations with only mutation operator. As a result, it is possible to search the space of possible genetic codes more effectively to find the optimal one. The shortening time of searching for the potential solution can be useful in further extensive and time-consuming studies when different fitness functions and structures of genetic code will be considered. The found optimal codes minimize the costs of amino acid replacements in terms of polarity much better than the canonical genetic code and are characterized by the domination of amino acids with the average polarity value.

Acknowledgements

We are very grateful to anonymous reviewers for their comments and insightful remarks that significantly improved the paper.

References

Crick, F., 1968. The origin of the genetic code. J. Mol. Biol. 38, 367–379.
Di Giulio, M., 1989. The extension reached by the minimization of the polarity distances during the evolution of the genetic code. J. Mol. Evol. 29, 288–293.
Di Giulio, M., 1991. On the relationships between the genetic code coevolution hypothesis and the physicochemical hypothesis. Z. Naturforsch. C 46 (3–4), 305–312.
Di Giulio, M., Feb 2000. The origin of the genetic code. Trends Biochem. Sci. 25 (2), 44.
Di Giulio, M., Jun 2016. The lack of foundation in the mechanism on which are based the physico-chemical theories for the origin of the genetic code is counterposed to the credible and natural mechanism suggested by the coevolution theory. J. Theor. Biol. 399, 134–140, http://dx.doi.org/10.1016/j.jtbi.2016.04.005.
Di Giulio, M., 2005. The origin of the genetic code: theories and their relationship, a review. Biosystems 80 (80), 175–184.
Dunnill, P., 1966. Triplet nucleotide-amino-acid pairing: a stereochemical basis for the division between protein and non-protein amino-acids. Nature 210 (June (5042)), 1265–1267.
Fogel, D., Atmar, J., 1990. Comparing genetic operators with Gaussian mutations in simulation evolutionary processes using linear systems. Biol. Cybern. 63, 111–114.
Freeland, S., Hurst, L., 1998. The genetic code is one in a million. J. Mol. Evol. 47 (3), 238–248.
Freeland, S.J., Knight, R.D., Landweber, L.F., Hurst, L.D., Apr 2000. Early fixation of an optimal genetic code. Mol. Biol. Evol. 17 (4), 511–518.
Freeland, S.J., Wu, T., Keulmann, N., 2003. The case for an error minimizing standard genetic code. Orig. Life Evol. Biosph. 33 (October (4–5)), 457–477.
Gilis, D., Massar, S., Cerf, N., Rooman, M., 2001. Optimality of the genetic code with respect to protein stability and amino-acid frequencies. Genome Biol. 2 (11).
Haig, D., Hurst, L., 1991. A quantitative measure of error minimization in the genetic code. J. Mol. Evol. 33, 412–417.
Kokosiński, Z., 2005. Effects of versatile crossover and mutation operators on evolutionary search in partition and permutation problems. In: Intelligent Information Processing and Web Mining. Volume 31 of the series Advances in Soft Computing. Springer-Verlag, Berlin Heidelberg, pp. 299–308.
Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif. Intell. Rev. 13 (April (2)), 129–170, http://dx.doi.org/10.1023/A:1006529012972.
Mackiewicz, P., Biecek, P., Mackiewicz, D., Kiraga, J., Baczkowski, K., Sobczynski, M., Cebrat, S., 2008. Optimisation of asymmetric mutational pressure and selection pressure around the universal genetic code. In: Bubak, M., Dongarra, J., VanAlbada, G.D., Sloot, P.M.A. (Eds.), Computational Science – ICCS 2008, PT 3. Vol. 5103 of Lecture Notes in Computer Science. Elsevier, Springer, pp. 100–109.

Novozhilov, A.S., Wolf, Y.I., Koonin, E.V., 2007. Evolution of the genetic code: partial optimization of a random code for robustness to translation error in a rugged fitness landscape. Biol. Direct 2, 24, http://dx.doi.org/10.1186/1745-6150-2-24.
Park, K., Carter, B., 1995. On the effectiveness of genetic search in combinatorial optimization. In: SAC, pp. 329–336, http://dx.doi.org/10.1145/315891.316011.
Pelc, S.R., Welton, M.G., 1966. Stereochemical relationship between coding triplets and amino-acids. Nature 209 (February (5026)), 868–870.
R Core Team, 2015. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, http://www.R-project.org/.
Santos, J., Monteagudo, A., 2010. Study of the genetic code adaptability by means of a genetic algorithm. J. Theor. Biol. 264 (June (3)), 854–865, http://dx.doi.org/10.1016/j.jtbi.2010.02.041.
Santos, J., Monteagudo, A., 2011. Simulated evolution applied to study the genetic code optimality using a model of codon reassignments. BMC Bioinformatics 12, 56, http://dx.doi.org/10.1186/1471-2105-12-56.
Spears, W.M., 1992. Crossover or mutation? In: Whitley, D. (Ed.), Proceedings of the 2nd Foundations of Genetic Algorithms Workshop. Morgan Kaufman, pp. 221–237.
Spears, W.M.,1994. Simple population schemes. In: Proceedings of the 1994 Evolutionary Programming Conference. World Scientific, pp. 296–317.
StatSoft, 2014. STATISTICA (Data Analysis Software System), Version 12. www.statsoft.com.
Syswerda, G., 1991. Schedule optimization using genetic algorithms. In: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, pp. 332–349.
Taylor, F.J., Coates, D., 1989. The code within the codons. Biosystems 22 (3), 177–187.
Woese, C.R., 1973. Evolution of the genetic code. Naturwissenschaften 60 (10), 447–459.
Wong, J.T., 1975. A co-evolution theory of the genetic code. Proc. Natl. Acad. Sci. U. S. A. 72 (May (5)), 1909–1912.
Wong, J.T.-F., 2005. Coevolution theory of the genetic code at age thirty. Bioessays 27 (April (4)), 416–425, http://dx.doi.org/10.1002/bies.20208.
Wood, S., 2006. General Additive Models: An Introduction With R. Chapman & Hall/CRC.
Yarus, M., Caporaso, J.G., Knight, R., 2005. Origins of the genetic code: the escaped triplet theory. Annu. Rev. Biochem. 74, 179–198, http://dx.doi.org/10.1146/annurev.biochem.74.082803.133119.