Transposition, a bio-inspired operator

Eleonora Gatti[1] and Giulio Taiocchi[1]

University of Milan, Milan, ITA

Abstract. We study in the landscape of evolutionary algorithm a bio-inspired variation operator, the **transposition**, and we compare it with the uniform crossover operator. We use the Rastrigin and the Schwafel function and we run the algorithm for 30 times, in order to assure a valid statistical meaning. We graphically compare the best overall values and the generation average of the bests values. We perform statistical test with the best at the end values, to understand if there are significance difference between the methods.

Keywords: Evolutionary Computation · Transposition · Uniform crossover

1 Theoretical introduction

1.1 General evolutionary algorithm

Evolutionary computation can be used to face many different optimization problems. The process is mainly inspired by **Darwin's theory of evolution** and biological mechanisms from **genetic**. The theory of evolution hypothesize the "survival of the strongest": given a population in a specific environment, the fittest individuals will adapt better with respect to the others, and will have more probabilities to reproduce. In this way, in the new generations, a big part of the genetic material will be inherited from the old fittest individuals, and so on. In the evolution process, beside the selection, there are variation mechanisms that change the genetic materials of the individuals, some of them are mean to mate individuals, some others apply random variations. The mating usually happen just between the fittest individuals, creating a new generation that contain mesh of high fitted genetic material, so that the successive generations inherit only the better features. Moreover, some variation operators, like the mutation one, can insert new features in the population. What usually happen is that, if those new features are good, they spread among the population, becoming common between the individuals of the next generations. If the new features are bad in terms of adaptation or fitness, the individuals that have them will have low probability to mate or to survive, and they will probably not pass to the next generations.

In a general evolutionary algorithm there are different phases. We begin from an initial population, then we select the individuals that will be used for the mating, we mate these individuals generating a new population, we apply other possible variation operator, and eventually we select the survivor from the old and the new generation. A simple scheme is reported in Fig. 1.

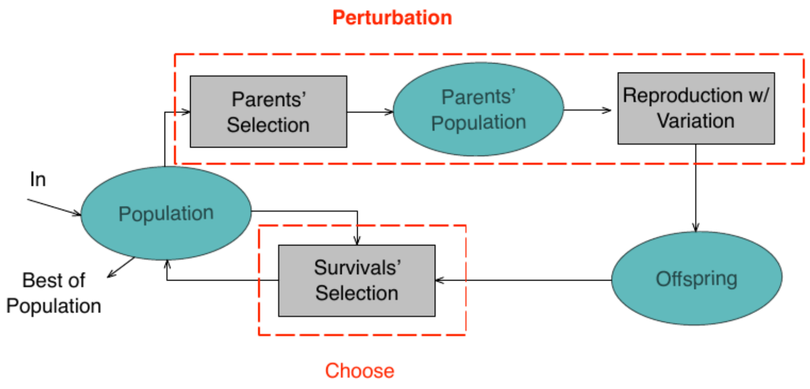

Fig. 1: General scheme of a evolutionary algorithm.

For each phase, there are many different ways to proceed. In this work we have chosen specific method for each phase of the algorithm that are explained right below. The aim of this work is to compare two different variation operator: **transposition** and **uniform crossover**.

1.2 The genetic algorithm phases

An **initial population** is randomly generated, then the algorithm pass through:

Parent selection: in our case we used the *tournament selection*, we randomly extract a T-size number of individuals from the population, and we take the fittest one. This procedure is done a number of time equal to the population size.

Variation: we used different methods. The first one is *uniform crossover*, that is performed with a certain probability. The chromosomes of two by two individuals are considered (let them be A and B), we take the first element of A and the first element of B and we swap them with a probability of 50%. Then we repeat for every couple formed by the i-th element of A and the i-th element of B. This corresponds to performing an exchange between the chromosomes using a random mask. The other operator is the *transposition*. It is based on the use of flags, called flanking sequence. Given the first individual, the presence of two flags (not overlapping) individuates a part of the chromosome. This part is candidate to be exchanged with a sequence of the same size from the second individual. This sequence is revealed, in the second individual, by the presence of the same flag. If we find two flanking sequence in the first chromosome and one in the second chromosome, we exchange genetic material between the two individuals.

Survival selection: at this point we have an old generation and a new one, we so want to decide the elements to keep. We used the *elitism* method.

All the old population is discarded, except for a percent that contain only the most fitted individuals. In this way we promote the transforming of the population without losing the ones that could still be the best individuals in our system pool.

2 The benchmark problem

Since every evolutionary algorithm strategy is problem depending, we need to analyze the problem that we want to face, that is the minimization of a function, with a continue and limited domain. The function considered are the *Rastrigin function* and the *Schwefel function*. We mainly have to decide three important things: representation, variation operators, and fitness. In our benchmark problem, the phenotype is obviously constituted by a vector of real number which length is defined by the dimension of the domain of the function. However, our goal is to test the transposition operator, that is not applicable with all the representation, and presents difficulties with some. For this reason we have chosen a **binary representation** for the genotype. This choice impose to define a level of decimal precision when converting a real number to a binary one, and this precision is set as a parameter. Also, the initial bit is used as a flag for the sign of the number. In dimensions higher than one, the genotype of an individual can be thought just as the merge of many one-dimensional genotypes. In this way, knowing the decimal precision parameter and evaluating at the beginning the maximum length for a one-dimensional chromosome, we can then split a multi dimensional genotype and individuate every binary component of the individual, each one corresponding to a specific dimension. The variation operators are mentioned in section 1.2, and cover an important task, since our aim is precisely to compare the uniform crossover with the transposition operator. For what concern the mutation, since we wanted it to apply just small changes, we limited its application to a fraction of the chromosome. Indeed the first left term of the genotype, if changed, lead to bigger modification in the phenotype, since the *power elevation* increase with the position, going from right to left. Note that variation operator can lead an individual out from the domain, in this case we assign to the individual the value of the bound. The fitness function is trivially the value of the benchmark function considered calculated with the phenotype.

3 Experimental setup

We now analyze the computational implementation for the problem. Given as known the implementation of most of the code, we focus on the transposition process, on the choice of parameters and on the strategy for storing and analyse the results.

3.1 Transposition

The general mechanism is described in the section 1.2, here we describe the computational implementation. Let's consider the binary representation of a couple of individuals. We chose the *flanking size* parameter (FS), that establishes the length of the flanking sequence. In the first individual a point P is randomly chosen in a subset of the chromosome[1]. Then we define the sequence of elements of length FS that is just before P as the flanking sequence. We search after P for an identical sequence using periodic boundary condition. If the one founded correspond to the initial one, transposition does not happen. Otherwise, we memorize the elements between the two flanking sequence, let's call them EM (exchange material). Then, starting from the first bit, we look in the second individual for a sequence identical to the flanking sequence. If not founded, transposition does not happen. Otherwise, we call Q the ending point of the sequence. Then we exchange EM plus the second flanking sequence (from the first individual) with a sequence of the same length that start from Q (from the second individual). In this procedure we use periodic boundary condition. We report the code for the transposition function.

```
# Transposition
def transposition(flank_size):
    def transpose(indiv_1, indiv_2):
        end = False
        start = False
        f1 = indiv_1[0]
        f2 = indiv_2[0]
        j = 0
        # define a random point in indiv_1
        size_indiv = len(f1)
        rnd_index = randrange(flank_size, size_indiv)
        flanking = f1[rnd_index-flank_size:rnd_index]
        # looking for the second flanking sequence in the first indiv
        while end == False:
            # if flanking==slice of indiv_1
            if (f1[ (rnd_index+j)%size_indiv : (rnd_index+j+flank_size)%size_indiv ] == flanking):
                end = True
                end_indiv_1 = (rnd_index+j+flank_size)%size_indiv # final index of the second flanking sequence
            else:
                j += 1
        j = 0
        while start == False and j <= size_indiv-flank_size:
            if (f2[j : j+flank_size] == flanking):
                start = True
                start_indiv_2 = j+flank_size # final index of the flanking sequence in the second indiv
```

[1] In particular we chose a random point between the FS-th element and the last one.

```
            else:
                j += 1
        if end_indiv_1 == rnd_index or start == False:
            return indiv_1, indiv_2
        # now the transposition happens
        i = 0
        # repeat len(transposone) times
        while (rnd_index+i)%size_indiv != end_indiv_1:
            f1[(rnd_index+i)%size_indiv] = indiv_2[0][(
                start_indiv_2+i)%size_indiv]
            f2[(start_indiv_2+i)%size_indiv] = indiv_1
                [0][(rnd_index+i)%size_indiv]
            i += 1
        return ((f1,0),(f2,0))
    return transpose
```

3.2 Data production

We have to decide which strategy apply in order to confront the results of uniform crossover and transposition. The choice of general parameters is important, and the variation operators could work better with different combination of them. Anyway, there are some specific parameters that are characteristic of some operators. In our case we deal with the *probability of crossover*, for the uniform crossover, and the *flanking sequence size*, for the transposition. The other parameters refers to other phase of the algorithm, and they will be listed further on. We can confront the goodness of an algorithm in different ways. In this work, we use the following strategy:

– Initialize a random population, that will be used as initial population for every algorithm.
– Run the algorithm for 30 times. This allow use to perform a statistical analysis. Note that statistic considerations are necessary since the method used are not deterministic.
– Memorize the best over all (BOA), the average over all the best of every run for each generation, and the best at the end (BAE) for each run. The best over all is the fittest individual among the best individual of every run and there is one for each generation. The best at the end is the best individual of the last population and we have one for each run.

The BAE allow us to perform statistical test, in order to investigate the compatibility or not of different distributions. The BOA and the average allow us to visualize and compare the fitness trend of the different methods in order to choose which algorithm is more appropriate for our problem. The general parameters are:

– Size of the population: 100 (and 200 for some of the crossover runs), the number of individuals that compose a population.

- Precision: set as 3 . The decimal numbers considered for the translation from real to binary.
- Tournament size: set as 3. Regarding the size of the tournament selection.
- Probability of mutation: set as 2%.
- Probability of crossover: set as 70%.
- Flanking sequence size: set as 3, 6, 9, 12, 15 and 18 (this last value has been used just for the first algorithm). The flanking size regulate the rate of changing in the population. Our hypothesis is: the lower is the flanking size, the more probable is to find one, and so the sequence in the between could be smaller. The higher is the flanking size, the bigger is the change and the algorithm could not be able to reach in the variation process some of the individuals. Probably, a mid value between too small and too big need to be found.
- Dimension: set as 12 and 20 for the Rastrigin function and as 15 for Schwefel function. Dimension of the domain of the function.
- Number of generations: set as 500 and 300 (depending on the study case), in order to be sure that the algorithm is able to find the minimum of the function.
- Number of runs: set as 30, as already mentioned.

4 Data analysis

We perform our analysis with different functions and in different dimensions.

4.1 Rastrigin function

The Rastrigin function is defined as:

$$f(x_1, ..., x_n) = A \cdot n + (\sum_{i=1}^{n} x_i^2 - A \cdot cos(2_i)), \quad x_i \in [-5.12, 5.12]$$

Where n is the number of dimension and $A = 10$. The value of the minimum of the function is 0 in every dimension.

Dimension 12
The first comparison done trough statistical test is between algorithms that use transposition with different values for the flanking size parameter. The predictor variable are categorical and we compare more than two element with the same conditions. We perform the Shapiro-Wilk and Levene test to verify if data are normally distributed and if the variance is uniform. The p-values obtained are extremely low, for this reason can we reject the null hypothesis. This preliminary analysis bring us to chose the Friedman ANOVA test. We chose a level of significance $\alpha = 0.05$. The result is:

$$p - value = 3.253294388381572e - 23$$

The p-vale is minor than α, this means that the methods are not statistically equivalent.
We visualize the distributions of the best at the ends.

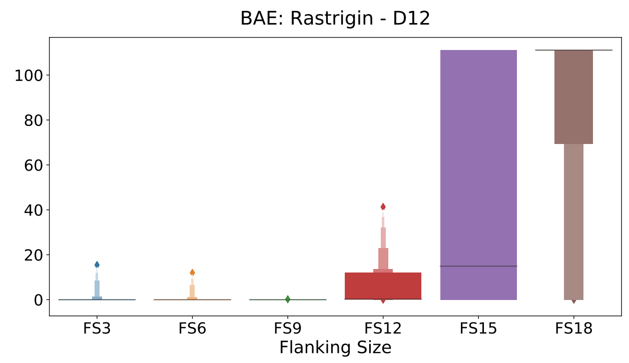

Fig. 2: Box plot of the best individual at the end of the run for the different flanking size in dimension 12.

The flanking size equal to 12, 15 and 18 present a elevated value of the best and so we discard them. In order to analyze the performance of the algorithms, we plot the BOA and the average over generations.

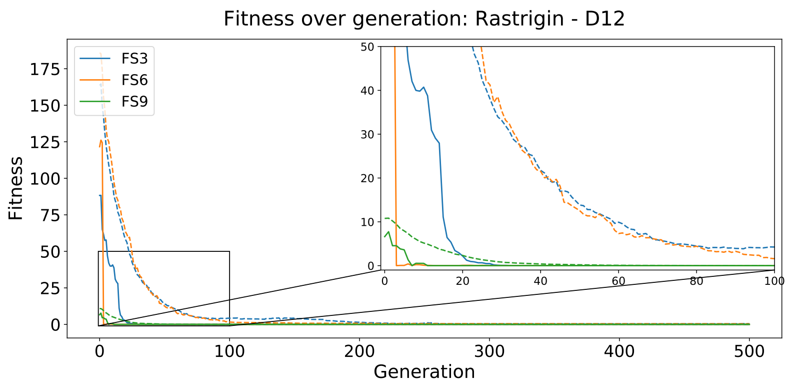

Fig. 3: Plot of the BOA and average of the solution for flanking size equal to 3, 6 and 9.

We see that the algorithm with the flanking size equal to nine reach the minimum faster than the other. Furthermore, the average behaviour is more compact and close to the minimum, this tells us that in the firsts runs the algorithm is acting more efficiently. We can also perform a statistical test between the possible pairs of this group of three flanking sizes. The test used is the Wilcoxon test. The results are here reported and the value of confidence level after the Bonferroni correction is $\alpha = 0.0166$.

- FS 3 and FS 6: $p - value = 0.003256361508313623$
- FS 3 and FS 9: $p - value = 1.0$
- FS 6 and FS 9: $p - value = 0.003281477916364575$

From these results we can conclude that the algorithm with FS equal to 3 and FS equal to 9 are statistically equivalent at the end of the last generation, while is the opposite for the algorithm with FS equal to 6. We chose to use FS9 as representative of the transposition when we will compare the different methods (Fig. 5).

Regarding the crossover two different runs has been performed, with population size equal to 100 and 200.

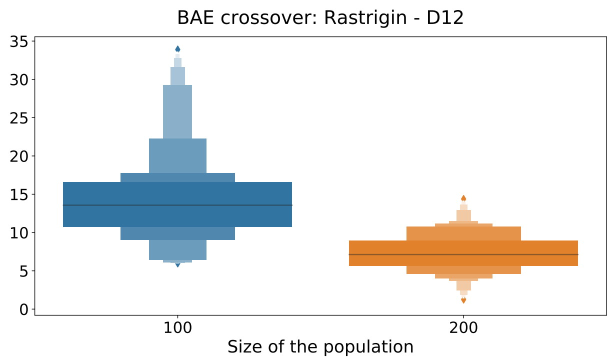

Fig. 4: Box plot of the best individual at the end of the run for crossover with different population size.

As expected, we can see that a bigger population lead to better results in terms of minimization. We now confront the generational behaviour between transposition and crossover methods.

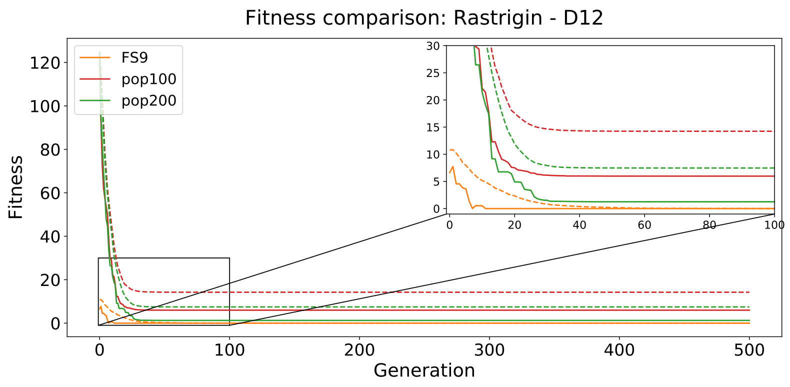

Fig. 5: Fitness comparison between uniform crossover with population size equal to 100 and 200 and transposition with flanking size equal to 9.

We can see from the slope and the low position of the orange curves, that the transposition method present a faster and more precise search of the minimum of the function than the two crossover method.

Dimension 20 Since we saw that for the transposition operator was not hard to find the best solution we decided to repeat the same confrontation but with the number of dimension increased to 20. The statistical test between algorithm that use flanking sequences of size 3, 6, 9, 12, 15 is performed. We used a non parametric test since the null hypothesis of being parametric had been rejected by Shapiro-Wilk test result. The result of Friedman ANOVA test is here reported[2]:

$$p - value = 4.8130356956262526e - 14$$

Again, different values for the flanking sequence lead to statistically different results. We visualize the box plot of the generational behaviour.

[2] We used a value of $\alpha = 0.05$ as usual

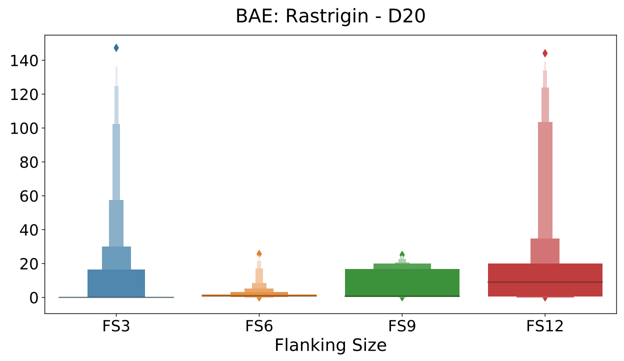

Fig. 6: Box plot of the best individual at the end of the run for different flanking size value in dimension 20.

The flanking sizes 12 and 15 (that here is not reported for scaling reason) present a mean bigger than the others, that have a similar mean. It is clear that the flanking size equal to six present a distribution more compact and close to zero than the others. We visualize the generational behaviour.

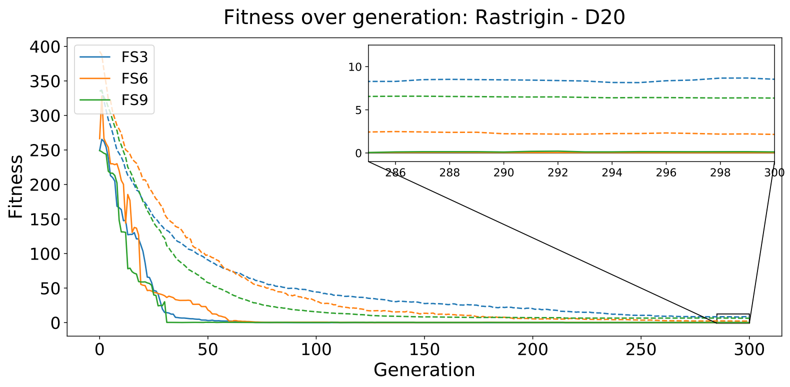

Fig. 7: Generational behaviour for different flanking size.

We can see that the flanking size equal to 9 is the fastest to find the minimum of the function. Anyway, as we saw from the box plot above, the average of the flanking size equal to 6 in the last generations is lower than the others. As done with dimension 12, we perform the Wilcoxon statistical text for the possible pairs.

- FS 3 and FS 6: $p - value = 0.051930665029697155$
- FS 3 and FS 9: $p - value = 0.02702915661831454$

– FS 6 and FS 9: $p - value = 0.13590778549641166$

We can see that every p-values is bigger than α after the Bonferroni correction ($\alpha = 0.0166$), and this means that the distributions can be statistically equivalent with a level of confidence equal to 0.05. We chose to use FS6 as representative of the transposition when we will compare the different methods (Fig. 9).

We report here the crossover results with population dimension 100 and 200.

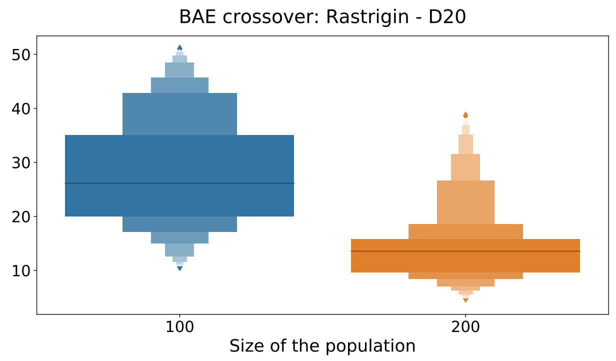

Fig. 8: Box plot of the best individual at the end of the run for crossover with different population size in dimension 20.

As expected with a bigger population the algorithm get closer to the minimum. Confronting with the results obtained in dimension 12, we can see that the distributions are more spread and further from the minimum value. This because, the higher the dimension, the higher the difficulty to minimize the function. A comparison between the transposition and the uniform crossover is performed.

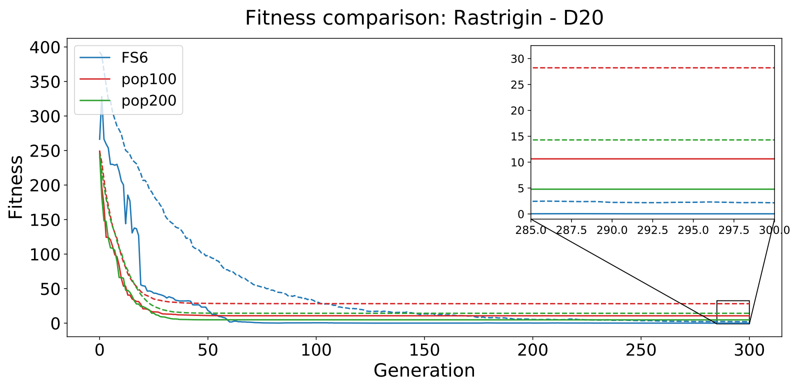

Fig. 9: Fitness comparison between uniform crossover with population size equal to 100 and 200 and transposition with flanking size equal to 6.

We can see that the crossover methods minimize faster the function, but they stop at a certain deep without being able to go lower. The transposition method seems to go slower (consider the dashed blue curve, the one of the average), but at the end of the generations it can identify a lower minimum.

4.2 Schwefel function

The Schwefel function is defined as:

$$f(x_1, ..., x_n) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right), \qquad x_i \in [-500, 500]$$

The value of the minimum of the function is $-n \cdot 418.9829$, where n is the number of dimensions[3].

Dimension 15 Due to the wide domain and limited computational power, we decided to analyze only the problem with dimension equal to 15. As the previous case we firstly realized Shapiro-Wilk and Levene tests for the results obtained from different values of the flanking size parameter. This preliminary analysis bring us to chose again the Friedman ANOVA test. We set: $\alpha = 0.05$ and we obtained from the test:

$$p - value = 5.105142138106372e - 08$$

The algorithm with different flanking sequence size are not statistically equivalent. We visualize the distributions of the best at the end.

[3] For 15 dimensions the value of the minimum is -6284.7435.

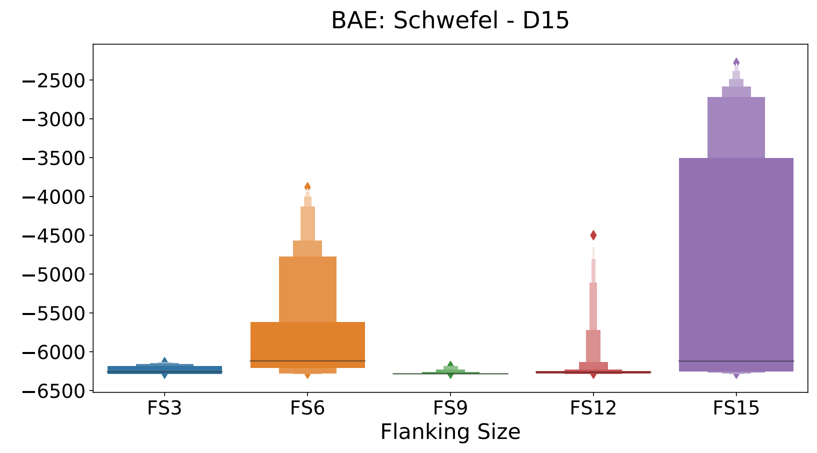

Fig. 10: Distributions of the BAE for the different flanking size.

The flanking size 6 and 15 seem to be inadequate compared to the others. The flanking size equal to 9 appear as the most compact to the minimum. Let's visualize the generational behaviour.

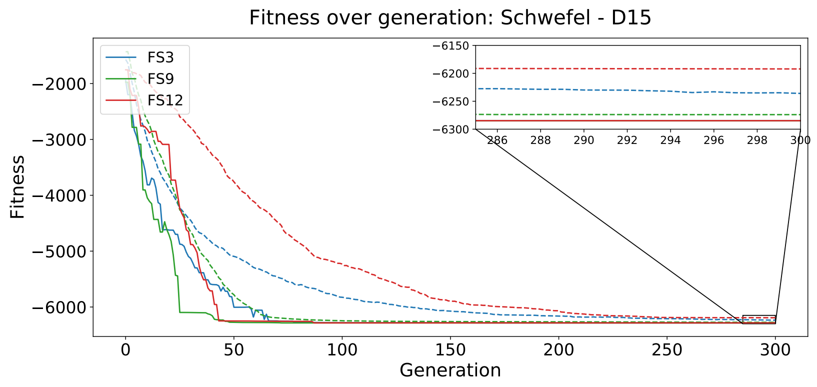

Fig. 11: Generational behaviour for different flanking size.

As predicted from the box plot, the flanking size equal to 9 has the more compact distribution and closer to the minimum. To understand if the distributions of the BAE are statistically equivalent, we perform the Wilcoxon test.

- FS 3 and FS 9: $p - value = 0.002957462130717483$
- FS 3 and FS 12: $p - value = 0.3933343721992797$

– FS 9 and FS 12: $p - value = 0.016565526979430395$

This results tells us that the algorithms with FS equal to 3 and 12 are statistically equivalent while when compared with the case with FS equal to 9 the differences are statistically significant. We can now confirm that the algorithm with FS9 is the best.

We used the uniform crossover method with population size equal to 100 and 200. We report the BAE distribution.

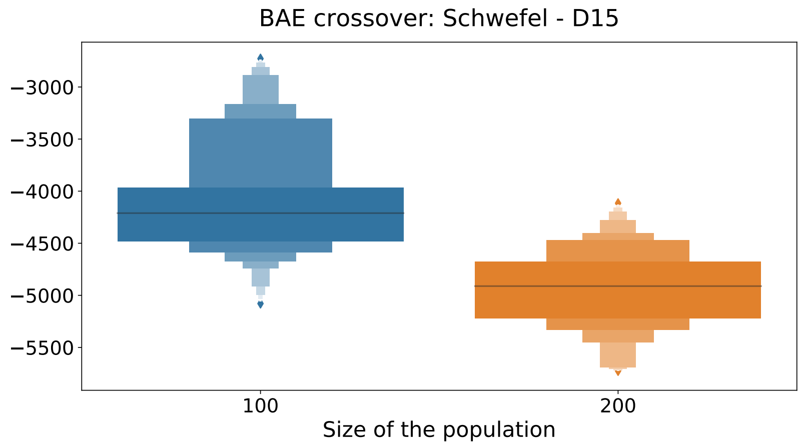

Fig. 12: Box plot of the best individual at the end of the run for crossover with different population size.

As expected and as before the algorithm with a bigger population reach better results. We can now show in the same plot the fitness comparison between the algorithm with transposition operator (flanking size equal to 9) and with crossover operator (with the two different size of population).

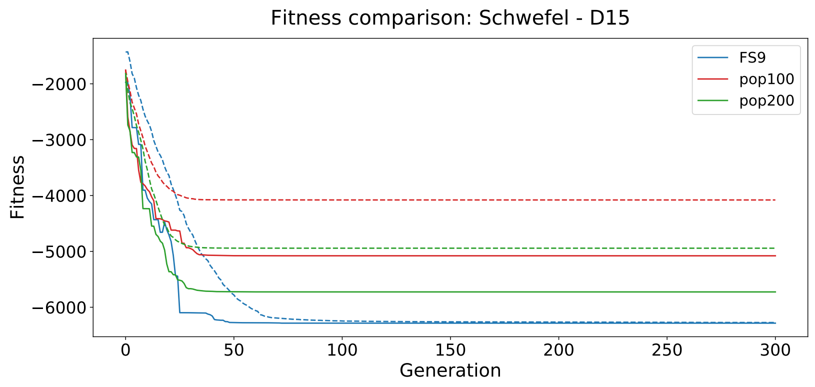

Fig. 13: Fitness comparison between uniform crossover with population size equal to 100 and 200 and transposition with flanking size equal to 9.

Again the transposition operator leads to more successful results.

5 Conclusions and possible improvements

In section 4.1, in dimension 12 (Fig. 5) we saw that using the transposition operator we are able to find a lower minimum for the function and in less generations compared to the algorithms that use uniform crossover method. Both the BOA and the generational average of the transposition method are below the corresponding curves of the uniform crossover algorithms. In dimension 20 (Fig. 9) the crossover seems to have a faster development toward the minimum, but after some generations the transposition is able to find a lower value and in last generations we find the same results as in dimension 12. In section 4.2 we arrived to a similar conclusion. Indeed the transposition operator seems to be more adequate to find the minimum of the function, as we can see in the last generation in figure 13. Based on these results, we can conclude that our benchmark problem, if faced with the reported parameters, is solved more appropriately by algorithms that use the transposition method. For sure, there are different improvements that we can apply to our algorithm. One of the thing that astonished us is the fact that, contrary to the transposition method, the crossover is not able to find the minimum in almost all the cases. Indeed, looking at the generational behaviours, it seems that after a certain number of generations the algorithms are not able to find a better individual. For this reason, we could think that such a number of generations is not necessary, and decrease it, in order to increase the population size and maintaining the same computational time.

References

1. Costa, E.: Nature-Inspired Artificial Intelligence. Springer Nature, (2021)
2. Costa, E.: Estatística Aplicada à Computação Evolucionária, (2018)
3. Costa, E., Simões, A.: Transposition: A Biological-Inspired Mechanism to Use with Genetic Algorithms. Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms (ICANNGA 99), pp. 178–186. Springer Vienna, Vienna (1999)
4. GitHub Repository, `https://github.com/eleonoragatti44/EvolutionaryComputationProject`. Last accessed May 2021.