Exploratory Study of Island-Based Parallel Differential Evolution Algorithms using Apache Spark

Quantitative Analysis

Davide Anghileri <davidean@kth.se>
Nathan Consuegra <nacon@kth.se>

2 November 2017

1 Introduction

First of all we want to premise that this is only a preliminary analysis of some of the results we got since now to improve our capabilities on elaborating and presenting our data and results, a more detailed and significative analysis will be included in the final report.

Since we are not going to collect any data and our experiment is an exploratory study based on the evaluation of the parameter for a differential evolution algorithm we will not do any preprocessing analysis. Also an exhaustive and well detailed description of the benchmark we are going to use (BBOB 2015) can be found here [1].

2 Results

Our evaluations are done on different instances of 25 predefined functions in the BBOB benchmark, the functions are different in term of shape, complexity and number of local optimum. Different instances of the same functions are just a random translation of the function in the D-dimensional space where it is defined, and allow us to have multiple running of the same function with different global optimum.

2.1 Analysis of the mutation factor F

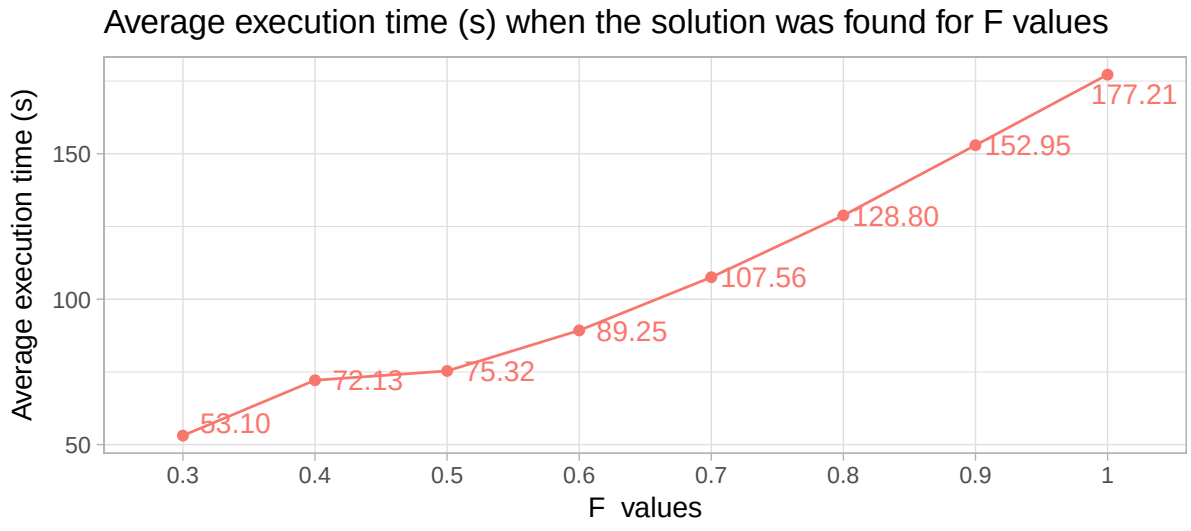

Figure 1: Average execution time with different values of F

Figure 1 shows for different values of the parameter F the average execution time to find a solution for a specific optimization problem. Is possible to see that large values of F, meaning that at each mutation the chromosomes change more, affect the execution time more or less linearly. This is reasonable and explainable by the fact that when you are near the optimum, if you continue to do large modifications, it can oscillate around the optimum and the time to reach it increases.

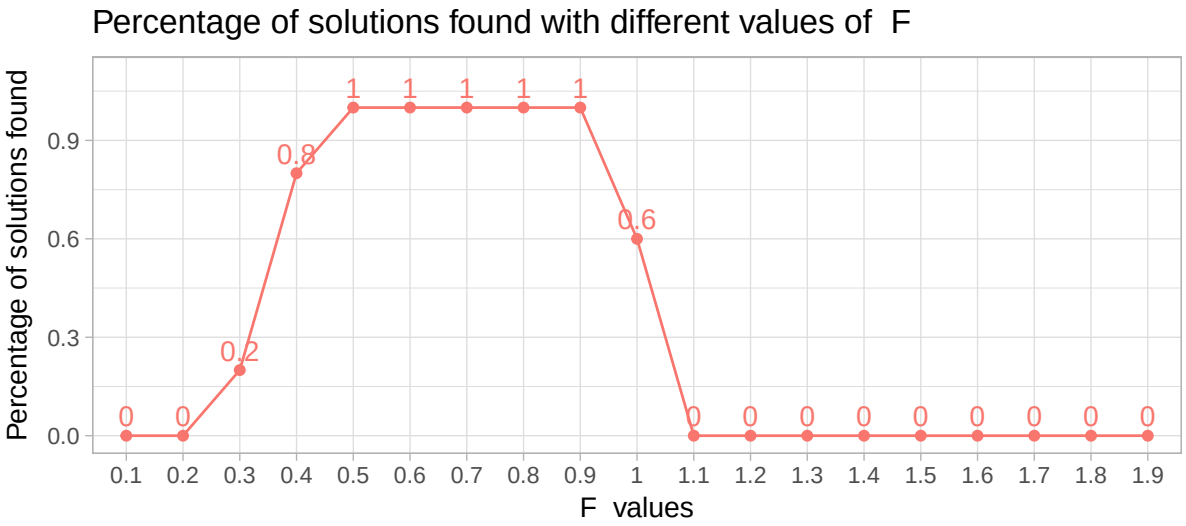

Figure 2: Percentage of solutions found with different values of F

Figure 2 shows for different values of the parameter F the percentage (# functions we found a solution / # total functions) of functions our algorithm found a solution. For this analysis we stopped the algorithm when it founds a solution (with a margin of 10^-8) or it reached 5000 evaluations. From this graph we can see that only values between 0.3 and 1 allows the algorithm to find a solution and for values between 0.5 and 0.9 there is no difference on the capacity of the algorithm to find a solution or not.

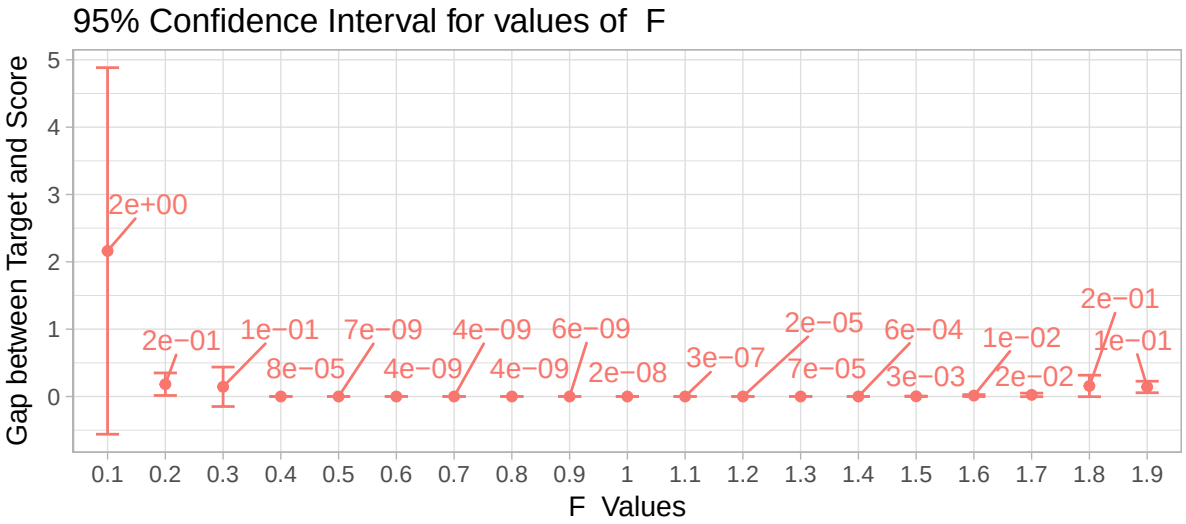

Figure 3: 95% CI for finding the solution with different values of F

Figure 3 shows how varying the parameter F impacts on the solution found within a maximum of 5000 evaluations. Is possible to see that for values of F between 0.4 and 1.5 the algorithm is able to find the global optimum, while for the value 0.1 the solution is really variable

and can be really far from the optimum, and finally for the remaining values the algorithm found a solution that is close to the optimum but not exactly it.

2.2 Analysis of the crossover factor CR

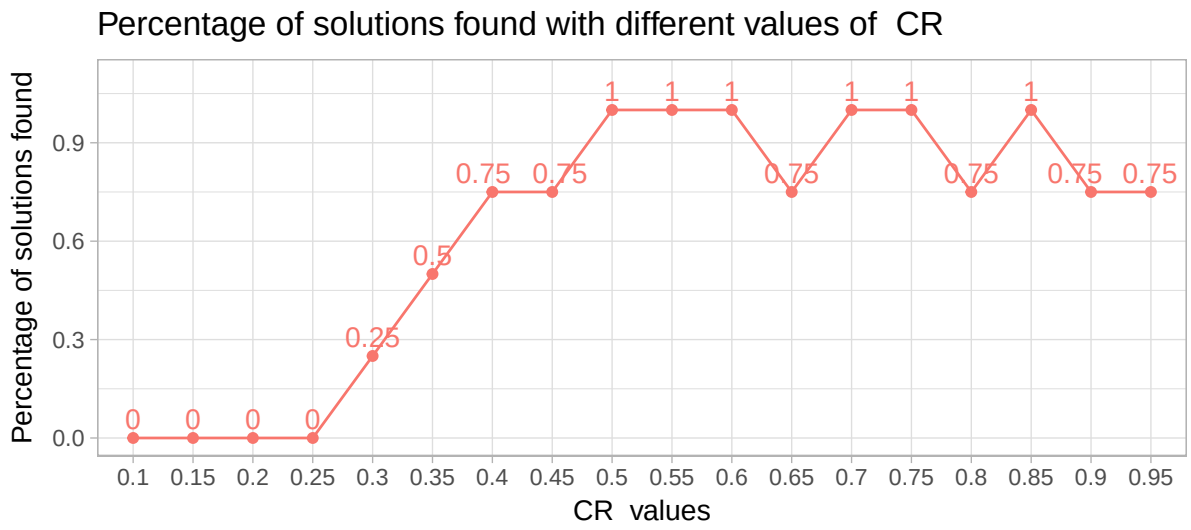

Figure 4: Percentage of solutions found with different values of F

Figure 4 shows the same analysis as Figure 2 but related to the CR parameter. For time constraint this analysis is done only over 6 of the 25 available functions and is possible to see that values of CR greater than 0.4 the algorithm is able to find most of the solutions in a reasonable time.

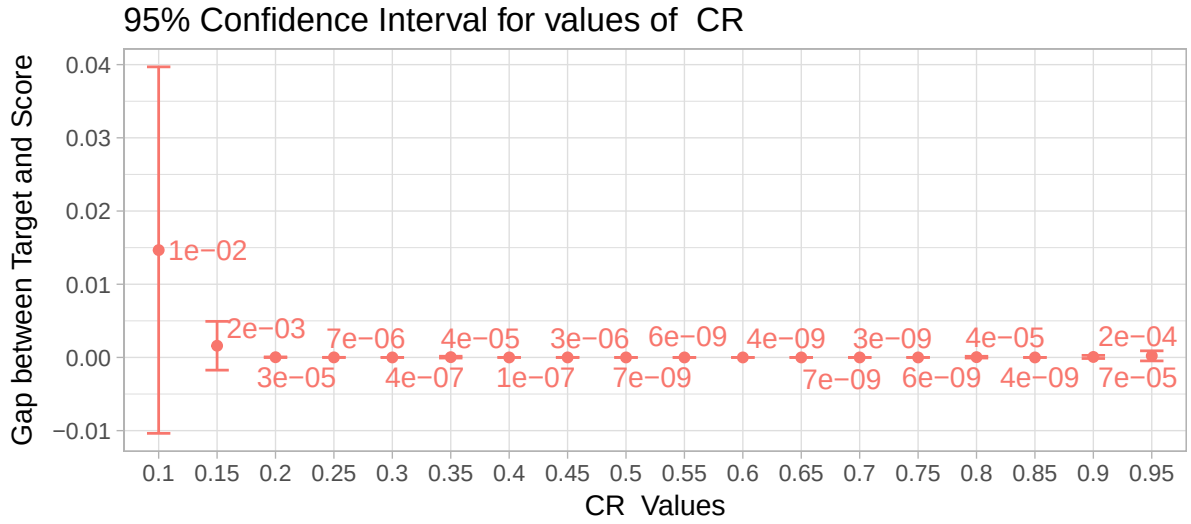

Figure 5: 95% CI for finding the solution with different values of CR

Figure 5 shows the same analysis of Figure 3 but related to CR parameter and similar conclusions can be easily derived from the graph.

Figure 6 shows how the parameter CR impact on the execution time and we can deduce from the plot that there is no a clear difference between different values of CR and the difference can be due to others factors. More analysis of this relation need to be done.

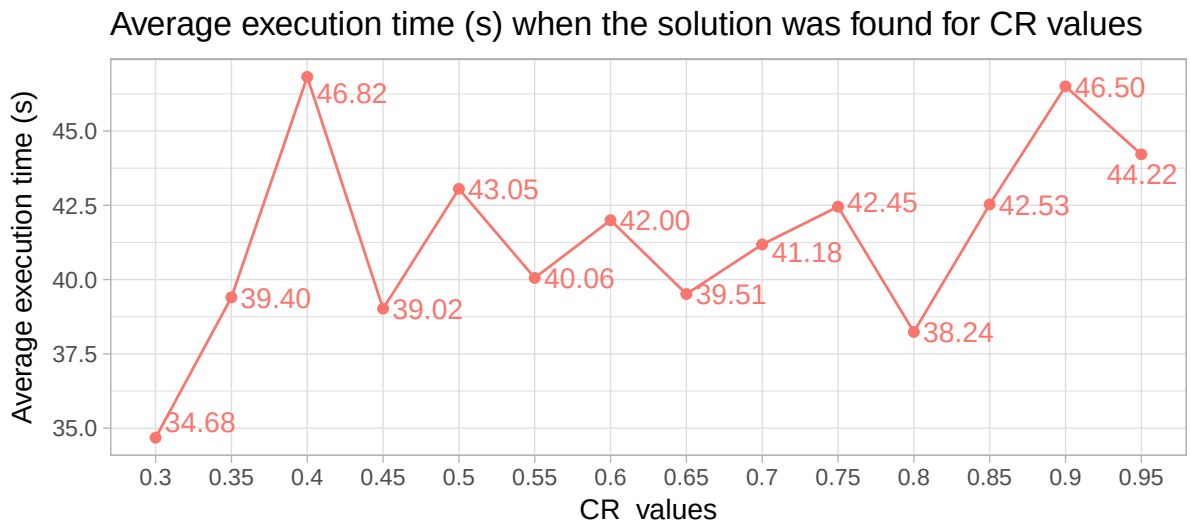

Figure 6: Average execution time with different values of CR

References

[1] Comparing continuous optimisers: Coco. http://coco.gforge.inria.fr/doku.php?id=bbob-2015-downloads.