Experimental Analysis of the Tournament Size on Genetic Algorithms

Yuri Lavinas*, Claus Aranha†, Tetsuya Sakurai† and Marcelo Ladeira‡
*University of Tsukuba, Faculty of Engineering, Information and Systems Email: yclavinas@gmail.com
†University of Tsukuba, Faculty of Engineering, Information and Systems
Email: caranha@cs.tsukuba.ac.jp and sakurai@cs.tsukuba.ac.jp
‡University of Brasilia, Department of Computer Sciences Email: mladeira@unb.br

***Abstract*—We perform an experimental study about the effect of the tournament size parameter from the Tournament Selection operator. Tournament Selection is a classic operator for Genetic Algorithms and Genetic Programming. It is simple to implement and has only one control parameter, the *tournament size*. Even though it is commonly used, most practitioners still rely on rules of thumb when choosing the tournament size. For example, almost all works in the past 15 years use a value of 2 for the tournament size, with little reasoning behind that choice. To understand the role of the tournament size, we run a real-valued GA on 24 BBOB problems with 10, 20 and 40 dimensions. We also vary the crossover operator and the generational policy of the GA. For each combination of the above factors we observe how the quality of the final solution changes with the tournament size. Our findings do not support the indiscriminate use of tournament size 2, and recommend a more careful set up of this parameter.**

I. Introduction

The Tournament Selection is a very popular selection operator for Genetic Algorithms and Evolutionary Computation in general. Some of the reasons for this popularity are that it is very simple to implement, and that it has a single control parameter, the *Tournament Size*, also known as "Tournament's k".

How should the value of the Tournament Size be chosen? Early works claimed that the value of k could be used to control the balance between exploration and exploitation in the search [1]. As k gets larger, the selection pressure is expected to increase, and better solutions are expected to exert a larger influence in the following generations [2].

On the other hand, works in the past 15 years have commonly used very small values for k, such as 2 or 3 [3]–[9]. The reason for this change is hard to identify, as most papers either do not justify their choice of parameters, or lightly mention a concern over computational cost for this rule of thumb.

Precious few works look too closely at their choice of tournament size. One example is Nicolau [6] who tried to empiricaly define a relationship between population size and tournament size for fine tuning a binary GA.

In order to question and further understand these assumptions and rules of thumb, we present an experimental study on the role of the tournament selection size. Our study focuses on a real-valued GA applied to the noise-free Black Box Optimization Benchmark (BBOB) functions.

We vary the crossover operator (uniform or SBX), the generational scheme, the modality of the objective function (unimodal or multimodal), and the dimensionality of the objective function (10, 20, 40). We observe how the quality of the final answer changes as the value of k changes.

Our results show that for functions with only one local optimum (easier problems), the choice of k does not show an effect on the final result. For harder problems, when the functions have many local optimum we saw an effect. The exact best value of k may depend on the function, dimensionality and algorithm composition and this relationship is not clear. Instead of using k equals to 2 or 3, we recommend that the choice of this parameter be considered more carefully.

II. The Tournament Selection

A selection scheme is a process that aims to improve the average quality of the GA by giving solutions of higher quality a higher probability of composing the next generation [1]. The tournament selection, as any selection mechanism in GA, simply favors the selection of better individuals of a population to influence the next generation.

Tournament Selection is a selection scheme that has a unique set of characteristics that makes it a very popular choice [1]. It is simple to code, and its complexity is O(n) on the size of the population [10] which makes it efficient for both nonparallel and parallel architectures.

The tournament selection is also translation and scaling invariant [11], which means that the behavior of the selection scheme is not affected by the scale of the fitness values [12].

It is often claimed that the tournament selection is able to adjust the selection pressure by changing its only parameter, which controls the number of individuals chosen to take part in the tournament. That is an important feature, because any selection scheme should have the capacity to adjust its selection pressure given different domains. When only two individuals are selected, it is often called a binary tournament [1], [10], which is a popular choice. However, in principle there is no limit for the size of the tournament.

A. The Tournament Selection Procedure

Tournament selection is conducted as: choose a number K of solutions randomly from the population and select the best solution from this group as a parent for the crossover operator. Repeat this operation for each parent necessary. This K is the tournament size parameter that we investigate in this paper.

B. Tournament Pressure

Muhlenbein et al., in [13], introduced the concept of selection intensity, which is to measure the pressure of selection schemes. The selection intensity is defined as the expected average fitness of the population after selection. This definition can only be applied because the tournament selection is scale and translation invariant [1].

Blickle et al. [1], declare that the selection intensity can be analytically calculated for tournament sizes of 2 and 3 and for larger values with numerical integration. They calculate the selection intensity for values between 2 and 30 and showed that with higher values for the tournament size the selection intensity increases. This calculation allows the prediction of the convergence time and can be used to compare different selection schemes.

Blickle et al. [14] compared some selection schemes, given their selection intensity. They concluded that for the same selection intensity the tournament selection has the smallest loss of diversity and the highest selection variance when compared with the truncation selection, the linear ranking, the exponential rank, and the proportional rank.

Miller et al. [2], argued that with higher tournament sizes, the selection pressure rises resulting in higher convergence rates, because the convergence rate of a GA is largely determined by the selection pressure. Accordingly, if the selection pressure is too low, the convergence rate will be slower, while if it is too high, there is an increased chance of the GA prematurely converging.

Based on the studies from above, we understand that the convergence directly influences the the final results of a GA, once any variation on the convergence rates changes the lost of diversity, and the selection variance. Consequently, it is important to explore the values for tournament size, since this value seems to determine the balance of exploration and exploitation by changing the selection intensity. It is possible that this balance could vary given a domain, once it is expected that every domain holds a specific characteristic that would need a different proportion of this balance.

C. Tournament Size in the Literature

In 1991, Goldberg [15], analyzed the tournament selection with size 2 in a real-coded GA. He recognized empirically that after some time only individuals with relatively high function values would be represented in the population.

Agrawal [16], in 1995, implemented real-coded GA with tournament selection with size 2. Their goal was to compare the results of the simulated binary crossover (SBX) with the binary single-point crossover and the blend crossover. They concluded that the SBX has search power similar to the single-point crossover and the SBX performs better in difficult test functions than the blend crossover. They understand that for some functions the selection pressure is not enough.

Tsutsui et al. [17] proposed a simplex crossover (SPX), a multi-parent recombination operator for real-coded genetic algorithms. For their experiments, they used a real-coded GA with tournament selection with size 2 and showed that "SPX works well on functions having multi modality and/or epistasis with a medium number of parents: 3-parent on a low dimensional function or 4 parents on high dimensional functions.".

In 2001, Beyer et al. [4], analyzed the results of using the simulated binary crossover (SBX), the fuzzy recombination operator and the blend crossover (BLX), all with no mutation and tournament size 2 in a real-coded GA. The results indicated that those crossovers exhibit similar performance and linear convergence order.

Sawyerr et al. [8], in 2015, proposed the RCGAu, hybrid real-coded genetic algorithm with "uniform random direction" search mechanism. They used tournament selection with size 3 to explore the suite of noiseless black-box optimization testbed and the results showed great performance in some functions such as f_7 and f_{21}.

Goldberg et al. [18], in 1993, implemented a binary GA using the tournament selection as a convenient way to control selection pressure. They wanted to study the theory aspects of the GA focusing in the mixing building block. The actual effect of the tournament value was not investigated empirically in the work.

In 1999, Harik et al. [19] studied a simple GA and experimented with the tournament size. Their work shows that a given solution quality can be obtained faster by the tournament size equal to 4 or 8, instead of tournament size equal to 2 or 40.

Also, Harik et al. [20] implement a binary coded GA using tournament selection. They analyze the impact of the tournament size on their GA with a 100-bit one-max function. They inspected this implementation with tournament sizes of 2, 4, and 8. They observed that the proportion of the building blocks changed with the tournament size, being higher for smaller tournament sizes.

In 2007, Kaelo et al. [5] compared the usage of the tournament size in a real-coded GA. They discuss the impact of using tournament size of 2 and tournament size of 3. The results of comparing these two tournament sizes indicated that the tournament with size 3 is more greedy, reduces the number of functions evaluations greatly and also improves CPU usage, therefore, it impacts negatively in the success rate of their GA.

Also in 2009, Nicolau [6] implemented a binary GA to solve the noise-free BBOB 2009 testbed. Through a quick experimentation with a subset of benchmark functions which yielded the following equation $k = P/500$ where P is the population size.

From the works reviewed here, it is possible to understand that there is a preference to small values for the tournament size. We traced back to 1991, and we found out that it has been a popular choice ever since. We must highlight that these values are usually chosen without much justification.

III. Noise free benchmark BBOB functions

The noise free benchmark BBOB functions [21] are a selected group of functions that are often used to evaluate the performance of evolutionary computation algorithms.

These functions can be separated in two groups by their complexity. The first group is composed on unimodal functions while the second group is composed on multimodal functions. Here we consider that unimodal functions are functions with only one local optimum (equal to the global optimum) while multimodal functions are functions that have many local optimum (with one of them being the global optimum).

The unimodal functions used here are: Sphere Function, Ellipsoidal Function (F2), Linear Slope, Atractive Sector Function, Step Ellipsoidal Function, Rosenbrock Function - original, Rosenbrock Function - rotated, Ellipsoidal Function (F10), Discus Function, Bent Cigar Function, Sharp Ridge Function, and Different Powers Function.

While the multimodal functions used here are: Rastrigin Function (F3), Buche-Rastring Function, Rastring Function (F15), Weierstrass Function, Schaffers F7 Function, Schaffers F7 Function moderately ill-conditioned, Composite Griewank-Rosenbrock Function F8F2, Schwefel Function, Gallagher's Gaussian 101-me Peaks Function, Gallagher's Gaussian 21-hi Peaks Function, Katsuura Function, and Lunacek bi-Rastrigin Function.

IV. Experiment Design

A. The Genetic Algorithm

To test the influence of the tournament size we implemented a simple real-valued Genetic Algorithm to explore the search space of the 24 noise free N dimension BBOB benchmark functions [21], while changing the value of the tournament size.

Genome Representation and Evolutionary Tools.: Each solution is represented as a N-dimensional real valued array, which is used as the input parameters to a N dimensional noise free BBOB function. The search space for each function is bounded to the [-5, 5] interval for every dimension.

We considerer two crossover operators when experimenting with the GA: the Uniform crossover and the Simulated Binary crossover (SBX). The other evolutionary operators are elitism and Gaussian mutation. The relevant parameters were set as in the Table I and Table II.

Number of Evaluations	40000
Population Size	800
Crossover chance	0.9
Mutation Chance	0.1
Elitism size	1

TABLE I: GA Relevant Parameters

Mean	0
Standard Deviation	1
Ind. Prob. Attribute to be Mutated	0.1

TABLE II: Gaussian Mutation parameters

Fitness Function.: The fitness function considered are the N dimensional noise free benchmark BBOB functions [21].

Generational Scheme.: Two generation schemes were used, the first is named the (λ, λ), also called the comma version, and the second is named the $(\lambda + \lambda)$, also called the plus version. The (λ, λ) scheme has a λ population of parents and for every generation it creates the λ offspring to compose the next population. The $(\lambda+\lambda)$ scheme creates the λ offspring and the best λ individuals from combined population of offspring and parents are select to the next generation, for more information refer to Pilát publication [22].

Code for reproduction of the experiment is available at github.com/yclavinas/gaParameterAnalysis.

B. Experiment

Our experiment aims to find out whether the value of K makes a consistent impact on the performance of GA, and if it does, whether this impact is dependent on the problem being solved, and on the composition of the GA. To answer this question, we follow this procedure:

First, we perform GA searches while varying the following factors: Function (the 24 unimodal and multimodal noise-free BBOB), Dimension Size (10, 20, 40), Crossover Operator (SBX or Uniform), Generational Scheme (λ, λ) or $(\lambda + \lambda)$ and Tournament Size (2 to 25). For each combination above, we repeat the GA search 40 times, and the mean final function evaluation value is chosen as the representative value for that particular combination.

We group the experiments by the modality of the function (unimodal or multimodal), the dimension and the GA composition, and for each group perform an inference test on the value of the tournament size. On each group we perform a Friedman Test to determine whether any of the values for K show a significant difference in the average performance. In each test, we set $\alpha = 0.05$, the dependent variable being the mean final function evaluation value, the blocking variable is the BBOB function and the treatment variable is the tournament size.

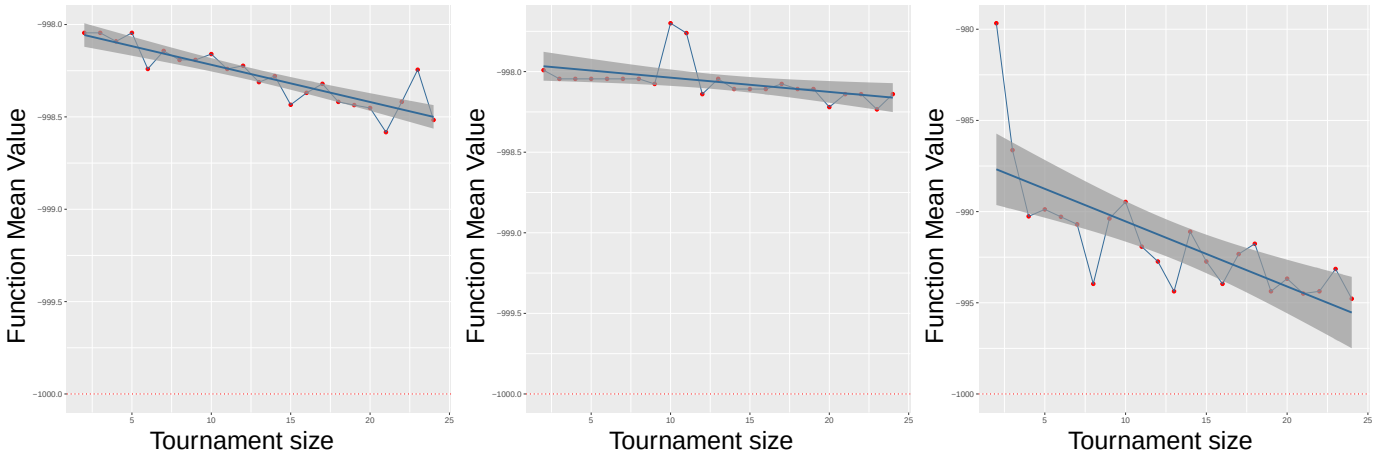
V. Results

Function Group	Dimension size	Chi-squared	P-value
Unimodal	10	24.955	0.2992
Multimodal	10	66.904	2.012e-06
Unimodal	20	20.24	0.5681
Multimodal	20	57.525	5.152e-05
Unimodal	40	25.34	0.2811
Multimodal	40	37.091	0.02312

TABLE III: Friedman Test results for Uniform Crossover - (λ, λ) scheme.

Function Group	Dimension size	Chi-squared	P-value
Unimodal	10	53.379	0.0002011
Multimodal	10	81.653	8.645e-09
Unimodal	20	113.02	3.181e-14
Multimodal	20	88.885	5.302e-10
Unimodal	40	163.13	$>$ 2.2e-16
Multimodal	40	118.46	3.316e-15

TABLE IV: Friedman Test results for SBX Crossover - (λ, λ) scheme.

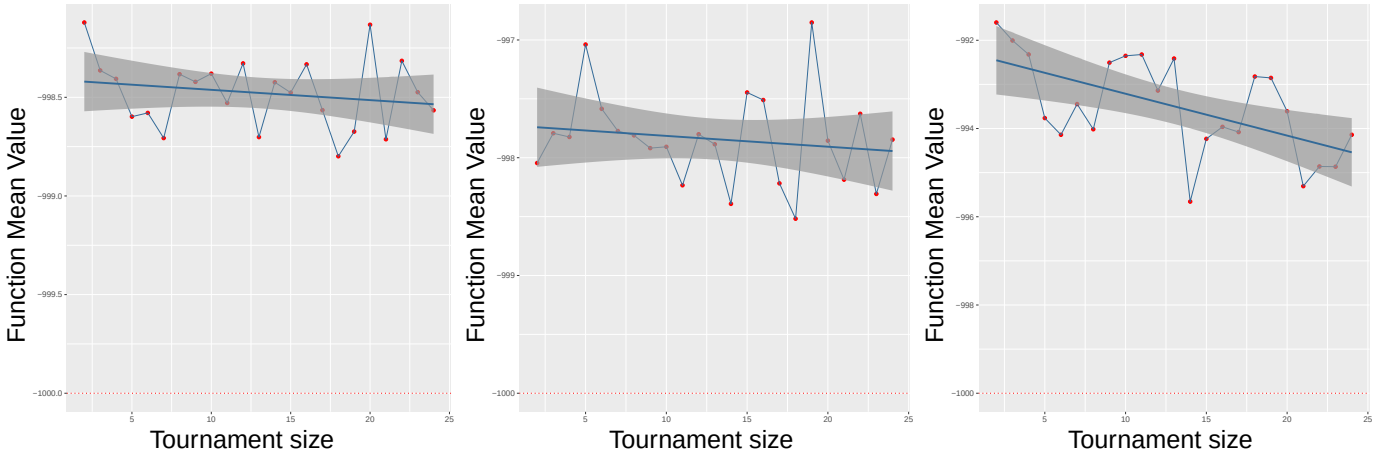


(a) Gallagher's Gaussian 21-hi Peaks Function - 10 dimensions.

(b) Gallagher's Gaussian 21-hi Peaks Function - 20 dimensions.

(c) Gallagher's Gaussian 21-hi Peaks Function - 40 dimensions.

Fig. 1: SBX crossover - (λ, λ) scheme.

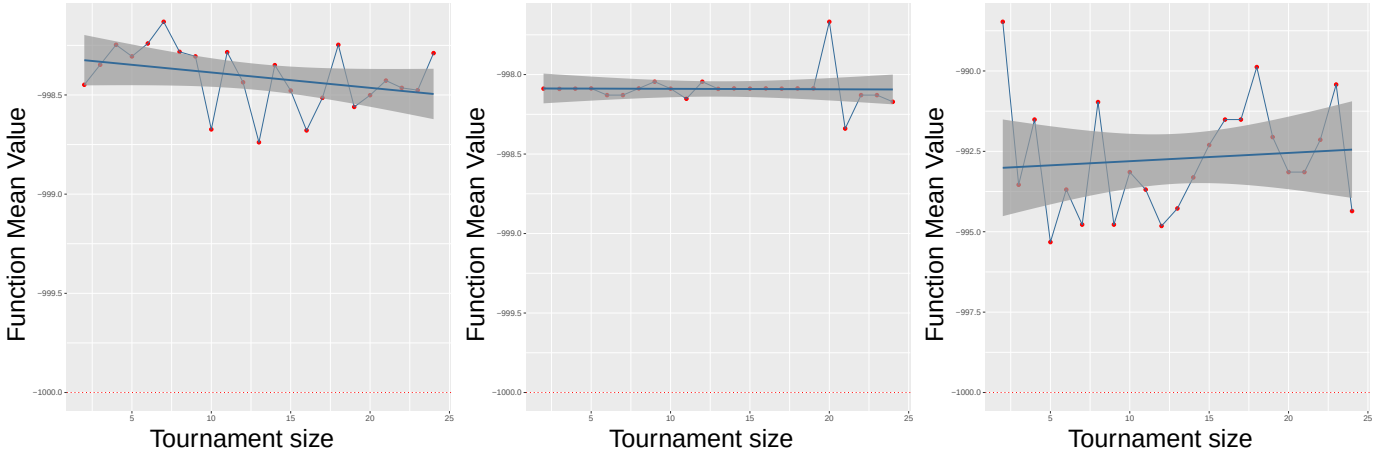


(a) Gallagher's Gaussian 21-hi Peaks Function - 10 dimensions.

(b) Gallagher's Gaussian 21-hi Peaks Function - 20 dimensions.

(c) Gallagher's Gaussian 21-hi Peaks Function - 40 dimensions.

Fig. 2: Uniform crossover - (λ, λ) scheme.

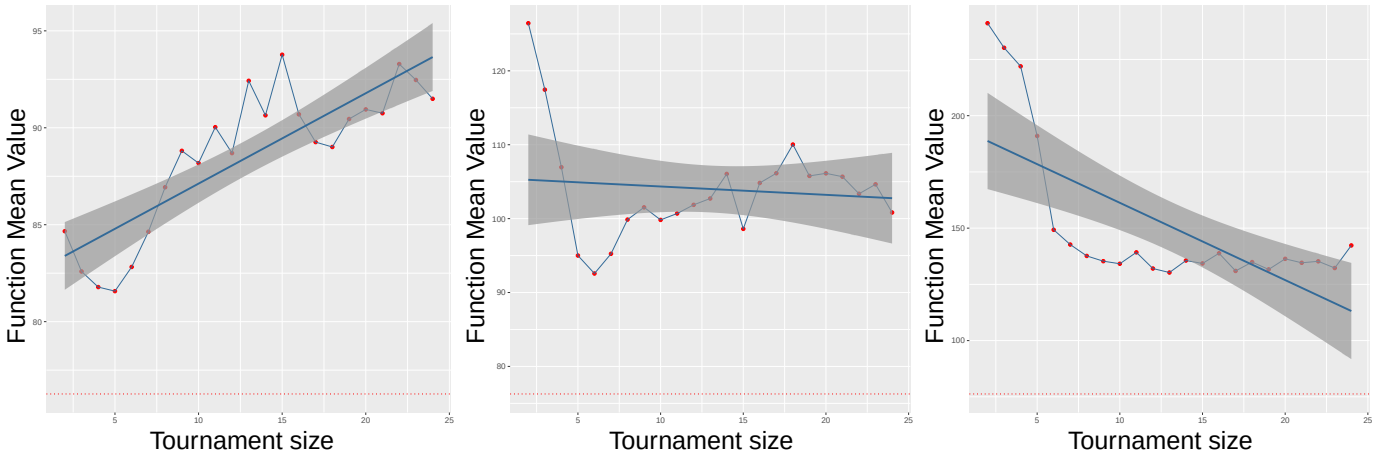


(a) Gallagher's Gaussian 21-hi Peaks Function - 10 dimensions.

(b) Gallagher's Gaussian 21-hi Peaks Function - 20 dimensions.

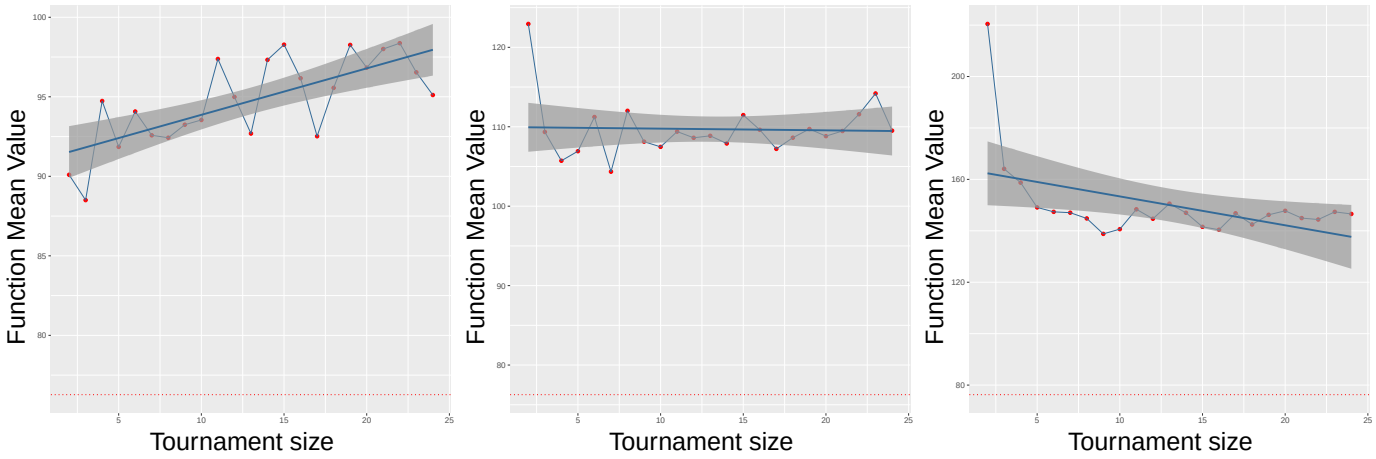
(c) Gallagher's Gaussian 21-hi Peaks Function - 40 dimensions.

Fig. 3: SBX crossover - $(\lambda + \lambda)$ scheme.



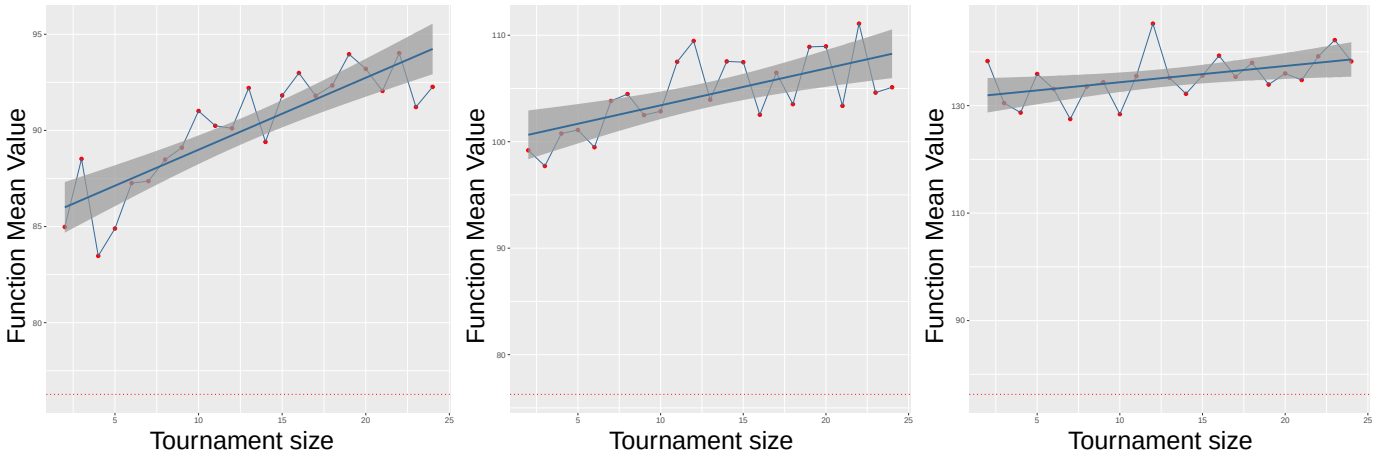
(a) Discus Function - 10 dimensions. (b) Discus Function - 20 dimensions. (c) Discus Function - 40 dimensions.

Fig. 4: SBX crossover - (λ, λ) scheme.



(a) Discus Function - 10 dimensions. (b) Discus Function - 20 dimensions. (c) Discus Function - 40 dimensions.

Fig. 5: Uniform crossover - (λ, λ) scheme.



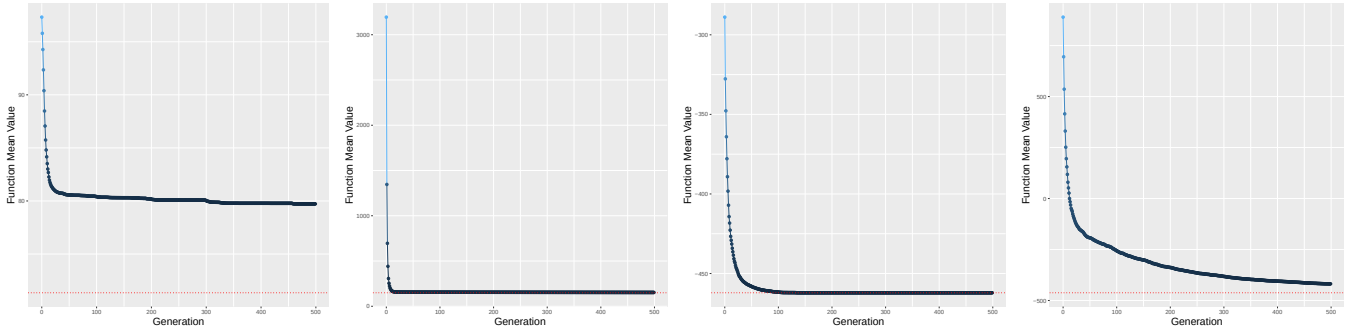
(a) Discus Function - 10 dimensions. (b) Discus Function - 20 dimensions. (c) Discus Function - 40 dimensions.

Fig. 6: SBX crossover - $(\lambda + \lambda)$ scheme.

A. *Overall Effect of K*

The Friedman Test showed no effect of changing the value of K for unimodal functions with the uniform crossover and any number of dimensions (Table III).

On the other hand, for unimodal functions with the SBX crossover, and for multimodal functions with any other setup,



(a) Weierstrass Function with the SBX crossover - $(\lambda + \lambda)$ - 20 dimensions

(b) Rosenbrock Function - original with the SBX crossover - (λ, λ) - 10 dimensions

(c) Buche-Rastrigin Function with the SBX crossover - $(\lambda + \lambda)$ - 10 dimensions

(d) Rastrigin Function with the SBX crossover - $(\lambda + \lambda)$ - 40 dimensions

Fig. 7: Examples of convergence for different combinations of functions and generational schemes.

Function Group	Dimension size	Chi-squared	P-value
Unimodal	10	54.011	0.0001639
Multimodal	10	75.724	8.073e-08
Unimodal	20	73.078	2.15e-07
Multimodal	20	95.524	3.867e-11
Unimodal	40	115.38	1.198e-14
Multimodal	40	87.318	9.762e-10

TABLE V: Friedman Test results for SBX Crossover - $(\lambda + \lambda)$ scheme.

the test indicated a significant effect of changing the value of K on the final result of the search (Tables IV, V).

An example of this effect can be seen in Figure 2, where it is possible to see that for the Gallagher's Gaussian 21-hi Peaks Function higher values for the tournament size indicate better results.

B. Analysis by Function

To get a finer intuition of these results, we show some visual examples separated in two groups of figures. The first group shows the mean value achieved by the GA for the Gallagher's Gaussian 21-hi Peaks Function, and the second group does the same for the Discus Function. Both groups show the mean function values of 40 repetitions for each set of parameters, and a linear regression line on the value of K against the mean function value. While the results discussed here are representative for the other functions studied, similar figures for all other functions are available on the source repository.

Figures 1 and 2 show that for the Gallagher's Gaussian 21-hi Peaks Function (with 10, 20 and 40 dimensions) changing the tournament size to higher values tends to increase the quality of the results for the generational scheme (λ, λ). On the other hand, for the generational scheme $(\lambda + \lambda)$, Figure 3 shows that changing the dimension has a dominating impact on the final values found by the GA. This means that in this case the best tournament value depends on the number of dimensions in the problem.

The dominance of the dimensionality observed in Figure 3 also occurs in the Discus Function. Figures 4, 5, (with 10, 20 and 40 dimensions) show that changing the dimensionality of the problem may have a high impact in the choice of the tournament value with the generational scheme (λ, λ). Figure 6 shows that with the generational scheme $(\lambda + \lambda)$ changing the tournament size leads to better final results. From these figures we understand that smaller values positively affect the results, specially with low dimensionality.

For all figures, the gray shaded area represents the 95% confidence level interval of a linear regression model. From these regressions, we can see that sometimes choosing small values for K can be a poor choice, a good choice, or make no difference at all. For all Figures, the mean of 40 repetitions is shown as bullets and the red line at the bottom shows the target value for the function.

C. Convergence Assumption

Finally, we want to make sure that all GA compositions used are actually performing the search (even though the final effectiveness varies with parameter choice). To verify this assumption, we perform a visual examination of the convergence of the functions studied.

Figure **??** is a selection of randomly samples of this examination, namely the Weierstrass Function, the Rosenbrock Function - original, the Buche-Rastrigin Function, and the Rastrigin Function (F3), respectively.

From these figures we can see that the studied GAs are indeed converging towards the optimum target value, represented by the bottom red line. Similar figures for other GA/function combinations are available on the source repository.

VI. Conclusion

We proposed an experimental analysis of the impact of the tournament size on the BBOB benchmark functions. We verified that there are few mathematical or experimental studies that supports a choice of value for the tournament size. Usually, the tournament size is chosen to be 2 or 3, based on "previous results".

We analyzed a group of tournament size values on different selection operators. From the results of this experiment, we found that using the tournament size with values like 2 or 3 may not always be a good choice, depending on the characteristic of the problem being explored.

Our experiments demonstrated that for each problem domain the effect of tournament size on the final result can have completely different characteristics. In some cases a higher tournament size achieves the best result, in others, the opposite happens. We also observed that even for the same problem, the choice of tournament size may depend on the dimensionality or the structure of the Genetic Algorithm being used.

In conclusion, we suggest that future works consider the use of self-adaptive schemes for the control of the tournament size parameter, instead of relying on fixed values. We believe that using this technique may diminish the time spend on the design process, and minimize the impact of bad choice of values for the tournament size by adjusting this parameter on-line during the search process.

Some of the limitations of the current experiment are the limited range of tournament size values tested, and the analysis of iteraction effects between the tournament size and other parameters of the GA, including the choice of operators. Overcoming these limitations would allow the community to better understand the role of the tournament size in the quality of the search.

REFERENCES

[1] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection." in *ICGA*, 1995, pp. 9–16.

[2] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[3] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[4] H.-G. Beyer and K. Deb, "On self-adaptive features in real-parameter evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 5, no. 3, pp. 250–270, 2001.

[5] P. Kaelo and M. Ali, "Integrated crossover rules in real coded genetic algorithms," *European Journal of Operational Research*, vol. 176, no. 1, pp. 60–76, 2007.

[6] M. Nicolau, "Application of a simple binary genetic algorithm to a noiseless testbed benchmark," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2473–2478.

[7] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained global optimization," *Optimization Methods and Software*, vol. 26, no. 6, pp. 945–970, 2011.

[8] B. A. Sawyerr, A. O. Adewumi, and M. M. Ali, "Benchmarking rcgau on the noiseless bbob testbed," *The Scientific World Journal*, vol. 2015, 2015.

[9] A. Oztekin, L. Al-Ebbini, Z. Sevkli, and D. Delen, "A decision analytic approach to predicting quality of life for lung transplant recipients: A hybrid genetic algorithms-based methodology," *European Journal of Operational Research*, vol. 266, no. 2, pp. 639–651, 2018.

[10] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[11] M. d. l. Maza and B. Tidor, "An analysis of selection procedures with particular attention paid to proportional and boltzmann selection," in *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1993, pp. 124–131.

[12] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2000, vol. 1.

[13] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm i. continuous parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 25–49, 1993.

[14] T. Blickle and L. Thiele, "A comparison of selection schemes used in genetic algorithms," 1995.

[15] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex systems*, vol. 5, no. 2, pp. 139–167, 1991.

[16] R. B. Agrawal, K. Deb, and R. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[17] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Morgan Kaufmann Publishers Inc., 1999, pp. 657–664.

[18] D. E. Goldberg, K. Deb, and D. Thierens, "Toward a better understanding of mixing in genetic algorithms," *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 10–16, 1993.

[19] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE transactions on evolutionary computation*, vol. 3, no. 4, pp. 287–297, 1999.

[20] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, 1999.

[21] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2010: Experimental setup," Ph.D. dissertation, INRIA, 2010.

[22] M. Pilát and R. Neruda, "Parallel evolutionary algorithm with interleaving generations," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 865–872.