Abstract

This is done by examining the results diferent values for the selection operator through simulations the 24 BBOB noise free benchmarks functions with 40 dimensions and comparing their behavior. The results are XXX (here is where i should state that the tournsize = 2).

1 Introduction

In this paper, the main goal is to observe the performance of the tournament selection in a real-valued Genetic Algorithm, named the GA-BBOB. For that we applied the GA-BBOB to 24 noise-free BBOB benchmark functions with 10, 20 and 40 dimensions, for more information see [12]. To achieve that, we first explore the number of individuals (k) to be selected by the Tournament operator, with values from 2 to 25. These values are arbitrarily chosen. Then we analyze the results of the GA-BBOB to verify any relationship between all the benchmark functions. Then we propose an alternative, an auto-adaptative k-value "generator". This alternative is done with the objective of chosing the ideal value of k.

Explore pressure

These things are described here and there. we do this and that here and there too. results and other stuff are here and there.

2 Tournament Selection

In this work we want to analyse the impact of varying the tournament size. For that first, we need to understand why selection schemes are important and how they improve the quality of the Genetic Algorithms (GA).

The selection scheme is intended to improve the avarage quality of the GA by giving individuals of higher quality a higher probabilty of composing the next generation [6]. It is also an important factor to the convergence quality of the GA and the goal of any selection scheme is to favor the proliferation of good individuals in the population of the GA [13].

Goldberg and Deb [10] stated that there exists several selection schemes commonly used in GA, such as the rank selection, the roulette-whell selection and the tournament selection. They also declare that the complexity of the tournament size is o(n) and say that tournaments are often held with tournament size s = 2, although larger tournament sizes may be analyzed.

2.1 Selection Schemes and BBOB functions

There are some examples of applications that applied these selection schemes with the BBOB functions. Next we show a few of this examples.

Holtschulte and Moses, in [15] used rank selection in their GA implementation and applied it to the BBOB functions. They declared that rank selection should be replaced by an elitism selection scheme to improve the performance

of their GA implementation by spreading of high quality solutions through the population.

One of these elitist selection scheme is the fitness-based roulette-wheel. However, it has been shown to lead to premature convergence [3]. Nicolau, in [19] uses the tournament selection in his GA implementation also aiming to solve the BBOB functions. He declares that "for harder problem domains, such as the ones on the BBO-Benchmark suite, the tournament selection scheme is more appropriate.".

Therefore, in this we work we chose to analyse the tournament selection behavior in the BBOB functions. The main reasons for that are because the tournament selection is a widely used selection operator [9, 11, 1, 13, 23, 14, 8, 4, 16, 5, 19, 22, 2, 20] and because, as showed previously, it is a more adequated selection scheme to by the BBOB benchmark functions. Likewise, it is simple to code, efficient in both parallel and non parallel architectures, and its pressure is able to be adjusted for different domains [17].

2.2 The Tournament Selection Procedure

To be able to understand the impact of the Tournament Selection and to be capable to analyse this selection scheme we not only need to describe how it is conducted, but also we need to study applications that used the Tournament Selection. Here we describe how it operates, and we show the studies next.

Tournament selection is conducted as follows: choose some number, k, of individuals randomly from the population and copy the best individual from this group into an intermediate population, and repeat N times.

Often tournaments are held only between two individuals (binary tournament) but a generalization is possible to an arbritrary group size k called tournament size [6, 10]. A tournament selection, as any selection mechanism in GA, simply favors the selection of better individuals of a population to influence the next generation.

2.2.1 Tournament Pressure

Muhlenbein et al., in [18], 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.

The selection pressure of tournament selection is expected to increse as the tournament size, k, becomes larger. In other words, the higher the selection pressure, the more the better individuals influences the next generation [17].

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 if it is too high, there is an increased chance of the GA prematurely converging [17].

Then, we understand that it is important to explore the Tournament size, hence it is possible that its value would change respectively to the domain of

the application that used Tournament Selection.

2.3 Tournament Size and its Application

In 1991, Goldberg [9], analysed the tournament selection with size 2 in a real-coded GA. They recognized, empirically, that after some time only individuals with relatively high function values will be represented in the population. Also it compares the importance of mutation operators and the tournament selection operator and states that "the addition of some small amount of creeping mutation or other genetic operators should not mat erially affect these results".

Goldberg et al. [11], in 1993, implemented a binary GA using the tournament selection as a conveniente way to control selection pressure. They wanted to study the theory aspects of the GA focusing in the mixing building block.

Agrawal [1], 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 with 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.

In 1999, Tsutsui et al. [23] proposed a simplex crossover (SPX), a multi-parent recombination operator for real-coded genetic algo- rithms. For their experiments, they used a real-coded GA with tournament selection with size 2 and showed that "SPX works well on functions having multimodality and/or epistasis with a medium number of parents: 3-parent on a low dimensional function or 4 par- ents on high dimensional functions.". In the same year, Harik et al. [14] 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. [13] implement a binary coded GA using tournament selection. They analyse 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.

Later, in 2000, Deb [8], implemented a binary GA and also a real-coded GA, both with tournament size 2. He aimed to devise a penalty function approach that does require any penalty parameter. Because all of their search spaces are defined in the real space, the real-coded GA were more suited in finding feasible solutions, but in all cases, that this approach " have a niche over classical methods to handle constraints".

In 2001, Beyer et al. [4], analysed 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.

In 2007, Kaelo et al. [16] compared the usage of the tournament size in a real-coded GA. They discuss the impact of using tournament with tournament

size of 2 and tournament size of 3. In their work, the results of comparing these two tournament sizes indicaded 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.

Bhunia et al. [5], in 2009, created a tournament GA application to solve an economic production lot-size (EPL) model.

Also in 2009, Nicolau [19] implemented a binary GA to solve the noise-free BBOB 2009 testbed. Trough a quick experimentation with a subset of benchmarck functions which yielded the following equation:

$$k = P/500, \text{ where P is the population size.} \tag{1}$$

Sawyerr et al. [20], in 2015, proposed the RCGAu, hybrid real-coded genetic algorithm with "uniform random directiom" 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 as f_7 and f_{21}.

2.3.1 Popular Tournament Size

From the works showed here, it is possible to understand that there is a preference to small values for the tournament size, as 2 or 3. We could trace back to 1991, and we found out that even from that time is has been a popular choice. However, we must highlight that these values are chosen arbirtrarily and has scientific background. For instance, in 2009, Nicolau used a more pragmatic approach and set an empirical equation to his binary GA.

3 GA-BBOB

In the work, we implemented the GA-BBOB, which is a real-valued Genetic Algorithm to explore the search space of the 24 noise free N dimension BBOB benchmark functions [12].

Genome Representation and Evolutionary Tools.

Each individual is represented as real valued array, where each element is one input to a N dimension noise free benchmark BBOB function. Therefore, each individual of the GA-BBOB has size of N real value elements. Each individual's values is set to the be into the interval [-5, 5], once those function are limited by these values. It uses the Uniform crossover, elitism and Gaussian mutation are used as evolutionary operators. The relevant parameters were set as Elite Size = 1, Crossover chance = 0.9, Mutation Chance = 0.1. For the Gaussian mutation, these parameters were chosen: mean = 0, Standard deviation = 1, and independent probability for each attribute to be mutated = 0.1.

Fitness Function.

The fitness function considered are the N dimensions noise free benchmark BBOB function. For more information, please refer to the work of Hansen et al. [12]. Consequently, the GA-BBOB has 24 different fitness functions.

Implementation Details.

Chuang et al. [7], proposed a Genetic Algorithm specific to run on the noise free BBOB benchmark functions. Some good practices presented in this work were incorporated to the GA-BBOB. These practice are the concept of "restart strategy" and the concept of "alleviate stagnation".

The restart strategy concept is defined by Chuang et al. as: "For each restart, the initial population is uniformly and randomly sampled within the search space. Whenever the restart condition is met, the algorithm will be reinitialized without using any information about the last test run. This process is iterated until the stopping criteria are met, i.e., maximum number of function evaluations has been reached, or the function value is less than the target precision."

The restart condition used in the GA-BBOB is value of the standard deviation of the current population. If it is smaller that 10^{-12}, then the restart condition is triggered. Therefore, when it is contemplated the algorithm population is reinitialized randomly, as it is in the first initialization.

In the GA-BBOB, we consider information about the last run, by utilizing the Elitism strategy on the best individual prior to the restart. The maximum number of function evaluations chosen is $(4*10^5)$ and the target precision chosen is (10^{-8}), values compatible with the ones in the previous work.

3.1 Genetic Algorithms and Benchmarks Functions.

IN 2009, Nicolau [19] proposed the first Genetic Algorithm on the BBOB-2009 noiseless testbed. It was a simple binary Genetic Algorithm and its results showed good results on separable functions, but poor performance is achieved on the other functions.

In 2012, Chuang et al. [7] presented the "DBRCGA", a real coded Genetic Algorithm that uses relative fitness information to direct the crossover toward a direction that significantly improves the objective fitness. The DBRCGA was tested on the BBOB-2012 noiseless testbed. Their results showed that the DBRCGA performs with difficulty in getting a solution with the desired accuracy for high conditioning and multimodal functions within the specified maximum number of evaluations, the DBRCGA presents good performance in separable function and functions with low or moderate conditioning.

In 2013, Holtschulte et al. [15] evaluated two Genetic Algorithm on the BBOB-2013 noiseless testbed, but none had results competitive with the best 2009 optimization algorithm. Their results highlight the importance of carefully chosen genetic operators.

Sawyerr et al found simliar results to the one achieved by Chuang et al. on their works in 2013 [21] and in 2015 [20]. In 2013, they proposed the "PRCGA",

or projection-based real-coded genetic algorithm. It incorporates exploratory search mecanism based on vector projection. In 2015, they have studied a hybrid application of the Genetic Algorithm (GA) with the *uniform random direction* search, named "RCGAu". BOth the PRCGA and the RCGAu were tested on the BBOB-2013 noiseless testbed. They stated that both of them had difficulties to achieve solutions with the desired accuracy for high conditioning and multimodal functions, though they were able to solve the benchmark functions. Sawyerr et al concluded that the RCGAu has excelled in solving the f_1, f_2, f_3, f_7 and f_{21}, tough for the other functions it achieved avarage results. They also state that real value GA do not efficiently solve highly conditioned problems and studies have currently been carried out to find out why [20].

4 Experiment Design

In this section we describe how the experiments were conducted.

4.1 (10, 20, 40) Dimensions Experiments

To analyze the impact of the tournament size into the GA-BBOB in therms of quality of result, the following experiments were performed. First we analyse different values for the tournament size applied to the BBOB benchmark functions with 40 dimensions. Then, we repeat the same experiments, but with 20 and 10 dimensions. The tournament sizes were select, arbitarily, from 2 to 25.

4.2 Experiment Design - Fixed tournament size

We also wanted to explore what would be the impact of varying the values of the tournament size *during* the execution. Here we want to explore what is the impact of shifting the tournament size during the execution of the GA-BBOB. We start with the tournament size equal to 2, as many of the work cited on section 2.3. Then when half of the evaluations are completed, the tournament size is changed to another value, chosen from 2 to 25.

5 Results

No significant variantion was found when varying the tournament size for any of the dimensions. With the Execution dependent tournament size, results were XXX - Yet to be found.

6 Discussion

7 Conclusion

Tournsize doesnt make any difference in quality

Maybe then you should use this
Maybe use two because of CPU...

Acknowledgments.

References

[1] Agrawal, R.B., Deb, K., Agrawal, R.: Simulated binary crossover for continuous search space. Complex systems 9(2), 115–148 (1995)

[2] Aranha, C., Lavinas, Y.C., Ladeira, M., Enescu, B.: Is it possible to generate good earthquake risk models using genetic algorithms? In: Proceedings of the International Conference on Evolutionary Computation Theory and Applications. pp. 49–58 (2014)

[3] Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms. pp. 14–21 (1987)

[4] Beyer, H.G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. IEEE Transactions on evolutionary computation 5(3), 250–270 (2001)

[5] Bhunia, A., Kundu, S., Sannigrahi, T., Goyal, S.: An application of tournament genetic algorithm in a marketing oriented economic production lot-size model for deteriorating items. International Journal of Production Economics 119(1), 112–121 (2009)

[6] Blickle, T., Thiele, L.: A mathematical analysis of tournament selection. In: ICGA. pp. 9–16 (1995)

[7] Chuang, Y.C., Chen, C.T.: Black-box optimization benchmarking for noiseless function testbed using a direction-based rcga. In: Proceedings of the 14th annual conference companion on Genetic and evolutionary computation. pp. 167–174. ACM (2012)

[8] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering 186(2), 311–338 (2000)

[9] Goldberg, D.E.: Real-coded genetic algorithms, virtual alphabets, and blocking. Complex systems 5(2), 139–167 (1991)

[10] Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms 1, 69–93 (1991)

[11] Goldberg, D.E., Deb, K., Thierens, D.: Toward a better understanding of mixing in genetic algorithms. Journal of the Society of Instrument and Control Engineers 32(1), 10–16 (1993)

[12] Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: Experimental setup. Ph.D. thesis, INRIA (2010)

[13] Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation 7(3), 231–253 (1999)

[14] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE transactions on evolutionary computation 3(4), 287–297 (1999)

[15] Holtschulte, N.J., Moses, M.: Benchmarking cellular genetic algorithms on the bbob noiseless testbed. In: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation. pp. 1201–1208. ACM (2013)

[16] Kaelo, P., Ali, M.: Integrated crossover rules in real coded genetic algorithms. European Journal of Operational Research 176(1), 60–76 (2007)

[17] Miller, B.L., Goldberg, D.E., et al.: Genetic algorithms, tournament selection, and the effects of noise. Complex systems 9(3), 193–212 (1995)

[18] Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm i. continuous parameter optimization. Evolutionary computation 1(1), 25–49 (1993)

[19] Nicolau, M.: 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. pp. 2473–2478. ACM (2009)

[20] Sawyerr, B.A., Adewumi, A.O., Ali, M.M.: Benchmarking rcgau on the noiseless bbob testbed. The Scientific World Journal 2015 (2015)

[21] Sawyerr, B.A., Adewumi, A.O., Ali, M.M.: Benchmarking projection-based real coded genetic algorithm on bbob-2013 noiseless function testbed. In: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation. pp. 1193–1200. ACM (2013)

[22] Sawyerr, B.A., Ali, M.M., Adewumi, A.O.: A comparative study of some real-coded genetic algorithms for unconstrained global optimization. Optimization Methods and Software 26(6), 945–970 (2011)

[23] Tsutsui, S., Yamamura, M., Higuchi, T.: 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. pp. 657–664. Morgan Kaufmann Publishers Inc. (1999)