X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará
© Sociedade Brasileira de Inteligência Computacional (SBIC)

CAR SETUP OPTIMIZATION USING MULTI-OBJECTIVE SWARM ALGORITHMS

M. C. Oliveira Junior, M. G. P. Lacerda, E. A. Barboza, P. R. G. Cordeiro, C. J. A. Bastos-Filho
University of Pernambuco
{macoj, mgpl, eab, prgc}@ecomp.poli.br, carmelofilho@ieee.org

Abstract – In this paper, we propose to use two multi-objective swarm optimization algorithms, called *MOPSO-CDR* and *MOABC*, to tackle the car setup optimization problem. We aim to find the best set of parameters for the car in order to improve its performance during the races. We used The Open Racing Car Simulator (*TORCS*) in our simulations and we compared our results to the ones presented in the *2010 Car Setup Optimization Competition*. We demonstrated that the *MOABC* can achieve similar results when compared to the state-of-art algorihms. The *MOPSO-CDR* outperformed all the previous approaches, including the *MOABC* algorithm for this problem.

Keywords – Multi-Objective Optimization, Swarm Intelligence, Particle Swarm Optimization, Artificial Bee Colony, Car Setup Optimization.

1. INTRODUCTION

The Open Racing Car Simulator (*TORCS*) is a car simulator which presents a very sophisticated physics engine. This engine considers many aspects of a real car and can predict the car performance for a given set of parameter by using a set of coupled equations. *TORCS* has been used in several international competitions, such as *the 2010 Simulated Car Racing Championship*, *the Demolition Derby Competition* and *the Car Setup Optimization Competition* [1]. The latter is an interesting scenario which can be viewed as a tough optimization problem, since it presents many continuous parameters to adjust. Thus, the goal is to develop an algorithm which can be able to optimize the parameters of the car, such as the angle of the wings, the pressure of the wheels, the gearbox ratios, etc.

TORCS presents an additional budget since the total time for optimization is limited. Moreover, during the optimization, each evaluation starts exactly at the point where the last one ended and, because of this, the environmental conditions for the individuals are in general different for all cases. One can increase the simulation time for each individual to overcome this temporal dependence. However, by doing this, the optimization algorithm will have a lower number of iterations to converge and the optimization procedure becomes hard. We present more details about this constrain in session 2.

In *TORCS*, the results are ranked based on the total length raced by the car. Cars with a parameter setup which run a longer total length are considered to have a better performance. On the other hand, to race for longer lenghts, the car has to increase its speed. If the parameters are set to increase the speed of the car and do not consider the control ability of the car, then the car can suffer more damages, and more damages imply in a shorter raced length. Therefore, one can observe a trade-off between the total length run by the car and the damages suffered by the car.

Multi-objective optimization algorithms can handle problems with conflicting objectives. Because of the observed trade-off, we believe these algorithms can be a suitable choice to tackle this specific problem. Besides, modern multi-objective optimization evolutionary algorithms (MOEA) using the dominance concept have been widely applied to solve real-world problems [2]. Actually, a well known MOEA, called *NSGA II*, was applied to the Car Setup Optimization problem and presented better results when compared to mono-objective approaches [3].

Swarm intelligence based algorithms have been proposed since the last decade of the last century and have been successfully applied to tackle optimization problems in hyper-dimensional continuous search spaces. Some swarm based approaches outperform evolutionary techniques for many different applications. Nevertheless, many multi-objective optimization algorithms based on swarms were recently proposed [4] [5] [6] [7].

Multi-objective Particle Swarm Optimization algorithms (MOPSOs) are based on the well known Particle Swarm Optimization algorithm (PSO) [8]. PSO was inspired by the behavior of flocks of birds when searching for food. MOPSOs are PSO variations which handle more than one conflicting objective. In a MOPSO, one has to define a scheme to select the leaders used to update the velocities of the particles. In this paper we consider the Multi-Objective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (*MOPSO-CDR*) algorithm. In *MOPSO-CDR*, the crowding distance (CD) parameter is used as the parameter of the roulette wheel to select the leaders.

The Multi-Objective Artificial Bee Colony (*MOABC*) is a multi-objective algorithm derived by the most famous bee based optimization algorithm. This algorithm was inspired by the behavior of bees searching for food and performing the waggle dance. Some adaptations were implemented to tackle multi-objective problems, such as a modification in the fitness evaluation and an addition of a method to store the non-dominated solutions found during the optimization process.

In the present paper, we propose to use two multi-objective optimization approaches based on swarm intelligence to tackle the Car Setup Optimization, the *MOPSO-CDR* and the *MOABC* algorithms.

The paper is organized as follow. In Section 2, we give some details about the car setup optimization problem and our application. The main concepts of multi-objective optimization are discussed in Section 3. The simulation setup is described in the Section 4. Some results are shown in Section 5. Finally, the conclusions are given in Section 6.

2. CAR SETUP OPTIMIZATION COMPETITION

The *TORCS* racing simulator is a very realistic open-source racing game which considers many real aspects such as gearbox ratios, rear and front wings angles, brake disc diameter, among others [3]. Because of this, *TORCS* has been used in several international academic competitions.

Our goal in this paper is the optimization of the Car Setup, in which we are interested to find out the best car configuration, *i.e.* the values for the parameters that yields to the best car performance. One must accomplish this task by using an optimization algorithm. Metrics regarding the car performance are often used as fitness functions.

In the competition, the optimizer can evaluate the average performance during 10 runs, where each run has $1,000,000$ game tics. Each call to the fitness function corresponds to a $10,000$ game tics simulation. At the end of each run, the optimizer must return the setup configuration which reached the longest length. The best algorithm will be the one which achieves the longest average length after 10 runs.

The car and the bot (computer controlled pilot) used by all competitors is always the same. In the 2010 edition of the competition, three different tracks were used on the competition: *CG-track2*, *Poli-track* and *dirt*. The shape of these three tracks can be seen in Fig. 1. The scores for each player depend on the final classification after the races. The winner is the one who reached the highest overall final score.

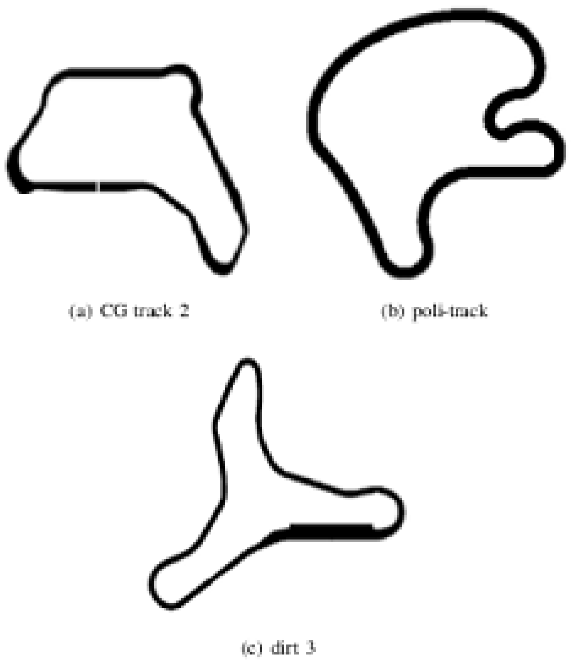


Figure 1: Tracks used in the *2010 Car Setup Optimization competition*.

The *TORCS* allows to setup the following 22 parameters of the car:

- Gearbox ratios – 5 parameters;
- Angle of the front and the rear wings – 2 parameters;
- Brake system – 4 parameters;
- Front and rear anti-roll bar – 2 parameters;
- Wheels camber, ride height and toe – 5 parameters;
- Suspension course and spring – 4 parameters.

The parameters must be passed to the *TORCS* as a vector composed by real numbers in the interval [0,1]. Then, it randomly maps this values to the parameters of the car. The mapping is different for each complete run (which corresponds to $1,000,000$ game tics in the competition). Therefore, it is impossible to use previous knowledge about the parameters. The competition provides an API that allows communication between the optimizer and the *TORCS*.

3. MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

In this section, we present some basic concepts related to Multi-Objective Optimization and we describe briefly multi-objective algorithms used in this paper. In Subsections 3.2 and 3.3, we present the *MOPSO-CDR* and the *MOABC* algorithms, respectively.

3.1. Multi-Objective Optimization Basic Concepts

A general multi-objective optimization minimization problem can be defined as:

$$\text{minimize } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})], \tag{1}$$

subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, ..., m, \tag{2}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., p, \tag{3}$$

where $\vec{x} = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^n$ is the vector on the decision search space; $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$ are the objective functions and g_i, $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$ are the constrain functions.

Given two vectors $\vec{x}$, $\vec{y} \in \mathbb{R}^k$, $\vec{x}$ dominates $\vec{y}$ if $x_i \leq y_i$ and $\vec{x} \neq \vec{y}$. The dominance of $\vec{x}$ over $\vec{y}$ is denoted by $\vec{x} \prec \vec{y}$. Thus, $\vec{x}$ is non-dominated if does not exist another current solution $\vec{y}_i$, such that $\vec{y}_i \prec \vec{x}$, $i = 1, ..., k$. The set of non-dominated solutions in the objective space is known as Pareto Front ($\mathcal{PF}^*$).

3.2. MOPSO-CDR

Multi-Objective Particle Swarm Optimization (*MOPSO*) is an algorithm to tackle multi-objective problems proposed by Coello Coello *et al.* [5]. It extended the Particle Swarm Optimization (*PSO*) algorithm to deal with two or more conflicting objectives [9].

In the standard *PSO*, each particle has four attributes: the position in the search space $\vec{x_i}(t)$, the velocity in the search space $\vec{v_i}(t)$, the best position found by the particle during the search process (cognitive memory – $\vec{p_i}(t)$) and the best position obtained by the entire swarm during the search process (social memory – $\vec{n_i}(t)$). The position of a particle represents a possible solution for the fitness function. At each iteration, the velocity and the position of all particles are updated. The velocity of the particles are updated by using a velocity equation, as shown in (4).

$$\vec{v_i}(t+1) = \omega\vec{v_i}(t) + c_1 r_1[\vec{p_i}(t) - \vec{x_i}(t)] + c_2 r_2[\vec{n_i}(t) - \vec{x_i}(t)], \tag{4}$$

where c_1 and c_2 are called acceleration constants, r_1 and r_2 are two random numbers generated by using an uniform probability distribution function $U(0,1)$ for each particle at each dimension. ω is the inertia factor and regulates the granularity of the search.

MOPSO algorithms present a different scheme to select the leaders used to update the velocities of the particles. In this case, an External Archive (EA) stores the non-dominated solutions obtained during the search process. The social leaders of the particles, $\vec{n_i}(t)$, are selected from the EA.

In the original *MOPSO*, the space of objectives is divided in hypercubes [5]. The fitness of each hypercube depends on the current number of EA solutions within the hypercube. A roulette-wheel mechanism is used to determine from which hypercube the leader will be selected. Once a hypercube is selected, one of the particles inside the hypercube is randomly chosen to be the social leader. *MOPSO* also uses a operator to perform a greedy search around the position of the particle. This operator is called turbulence.

Raquel *et al.* [10] proposed to use the crowding distance (CD) to select the leaders from the EA. Tsou *et al.* [11] proposed the *MOPSO-CDLS* which is a variation of the proposal introduced in [10]. It uses a roulette wheel based on CD to select the leaders from the EA. In this approach, there may occur two situations: the social leader of a particle is randomly chosen among the 10% less crowded solutions within the EA, if the particle is dominated by the EA solutions; otherwise, the social leader is randomly chosen from the entire EA. The cognitive leader, $\vec{p_i}(t)$, of each particle is updated if the new position dominates the current cognitive leader. If these solutions are incomparable, the cognitive leader is randomly chosen between the current position and the old cognitive leader.

The *MOPSO-CDR* [6] is an enhancement of the *MOPSO-CDLS*. It also selects the social leader based on a roulette wheel, where the particle within the EA with higher CD has higher probability to be chosen as a social leader. The CD is also used to limit the number of solutions within the EA. This returns a better distributed set of non-dominated solutions. The cognitive leaders are also selected comparing the CD of the closest solution within the EA. The Pseudocode of the MOPSO-CDR is depicted in Algorithm 1.

Algorithm 1: Pseudocode of the MOPSO-CDR algorithm

```
1  Initialize swarm;
2  Initialize leaders in the $EA$;
3  Evaluate the non-dominated solutions by using $CD$;
4  while stop criterion is not met do
5      for each particle do
6          Apply turbulence;
7          Select the leader using $CD$ as the criterion for the roulette wheel;
8          Update the velocity and the position;
9          Evaluate fitness in the new position;
10         Update $\vec{p}_{Best}$, if it is necessary;
11     end
12     Update leaders in the $EA$;
13     Evaluate the non-dominated solutions within $EA$ by using $CD$;
14     Truncate the $EA$ using the ranking based on $CD$, if it is necessary;
15 end
16 Return the solutions within the $EA$;
```

3.3. MOABC

Multi-Objective Artificial Bee Colony (*MOABC*) is another algorithm based on swarm intelligence conceived to tackle problems with multiple conflicting objectives. This algorithm was proposed by Hedayatzadeh *et al.* in [7]. It is based on the standard ABC algorithm introduced by Karaboga and Basturk in [12].

In the original ABC algorithm, there exists three categories of bees: employed, onlookers and scouts. There is only one employed bee for each food source, where a food source is a promising region of search space. The number of employed bees is defined in the beginning of the algorithm. The employed bees searches for good solutions around the food source and performs the waggle dance to attract the onlooker bees. The onlookers help to increase the exploitation ability around the selected food source. If the employed bee and the attracted onlookers can not find better results in a predetermined number of iterations, the employed bee abandons the food source and change its behavior to an exploration mode in order to find a new promising food source. When the employed bee is in the exploration mode, it is called a scout bee.

MOABC adds new features in the standard ABC to perform the multi-objective optimization. One of these features consists in a different procedure to evaluate the fitness of a food source ($\vec{x}_m$). This evaluation is performed using the following equation:

$$fit(\vec{x}_m) = \frac{dom(m)}{FoodNumber}, \tag{5}$$

where $dom(m)$ is the number of food sources dominated by the food source m. $FoodNumber$ is the total number of food sources.

MOABC also incorporated the use of an EA to store the non-dominated food sources found during the optimization process. The food sources stored in the EA are used by the employed bees to adjust their flying trajectories. It means that the *exploration* of the search space is guided by the EA. Then, it is necessary to store food sources with high diversity in order to maintain the population diversity along the entire process. The diversity of the EA is controlled using the dominance concept [4]. The Pseudocode of the *MOABC* is depicted in Algorithm 2.

4. SIMULATION SETUP

All simulations were run 10 times per each track and the average length covered in $10,000$ game tics gives the score for the algorithms. As in the competition, each run has $1,000,000$ game tics. A higher average length implies in a better fitness. Although there were three tracks in the competition, the *poli-track* is not currently available in the game. Because of this, we did not used this track in our simulations.

We selected two objectives for optimization: maximize the raced length and minimize the damage suffered by the car.

In the *MOPSO-CDR*, we used the mutation rate equal to 0.5 and the inertia factor linearly decreasing from 0.9 to 0.4. We tested the *MOPSO-CDR* with 4, 5, 10 and 20 particles. In the *MOABC*, the grid was divided in 30 sections. We tested the *MOABC* with 10, 16, 20 and 26 bees. The maximum size of the EA is equal to 200 particles for both algorithms. We compared our results to ones presented in the *2010 Car Setup Optimization Competition*.

5. RESULTS

Fig. 2 and 3 presents examples of accumulated Pareto Fronts after 10 trials in the *CG-track2* and *Dirty-track*, respectively, for *MOPSO-CDR* with 10 particles and *MOABC* algorithms with 16 bees. We always select the non-dominated solution which

Algorithm 2: Pseudocode of the MOABC

```
Initialize Food Sources;
while stop criteria is note met do
    Send Employed Bees;
    Send Onlooker Bees;
    Send Scout Bees;
    Define the Grid in the $EA$;
    Find the dominated hypercubes and remove the solutions of these hypercubes;
    for each remaining hypercube do
        if the hypercube contains more than one solution then
            Identify dominated solutions within the hypercube and remove them;
        end
        if the hypercube still contains more than one solution then
            Keep the solution which is closest to the left corner and remove the others;
        end
    end
end
Return the solutions within the $EA$;
```

provides the longest raced length. One must observe that the second objective (Damage) constrains automatically the first one (raced length).

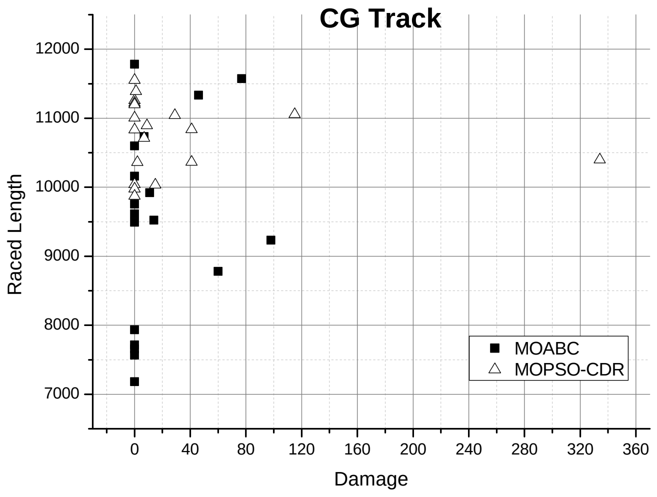


Figure 2: Accumulated Pareto Fronts after 10 trials in the CG-track for *MOPSO-CDR* and *MOABC* algorithms.

Fig. 4 and 5 depict the box plot of the total raced length for the *MOPSO-CDR* and *MOABC* algorithms. One can notice that the best results for the *MOPSO-CDR* were reached for 10 particles in the *CG-track2* and for 20 particles in the *Dirt-track*. Yet for the *MOABC* algorithm, the best results were reached for 16 bees in the *CG-track2* and 26 bees in the *Dirt-track*. The *MOPSO-CDR* achieved better results when compared to the *MOABC* algorithm in both tracks.

Table 1 shows the final results for our proposals and other approaches presented in the *2010 Car Setup Optimization Competition* [3]. As one can observe, the *MOPSO-CDR* achieved the best results in both tracks. The *MOABC* also obtained good results when compared to the approaches presented in the competition.

It is important to notice that the *Cardamone-SimpleGA* and *Kemmerling-CMA-ES* did not participate of the competition, since they were proposed by the competition organizers. However, even considering these two techniques, *MOPSO-CDR* wins on both tracks. Furthermore, the *MOABC* out performed the other previous approaches and just failed in the *CG-track2* by the *Kemmerling-CMA-ES*.

6. CONCLUSIONS AND FUTURE WORK

We showed that multi-objective swarm based algorithms can be successfully applied to the Car Setup Optimization problem. We outperformed all previous approaches presented in the competition by using the *MOPSO-CDR*. We believe we outperformed

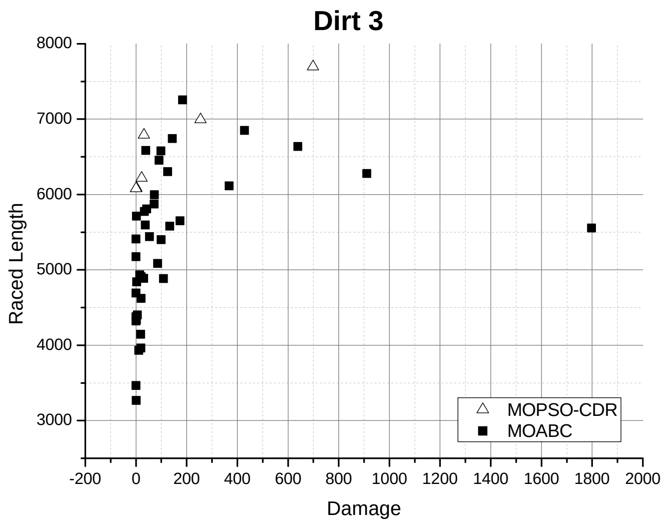


Figure 3: Accumulated Pareto Fronts after 10 trials in the Dirty-track for *MOPSO-CDR* and *MOABC* algorithms.

Table 1: The best results of the algorithms MOPSO-CDR, MOABC and the bests algorithms of EVOStar 2010, for each track

	CG track 2	Dirt 3	Overall
MOPSO-CDR	**11032.52**	**6773.94**	**17806.46**
Kemmerling-CMA-ES	10410.13	6415.87	16826
MOABC	10136.498	6423.71	16560.208
Munoz-MOEA	9831.83	6128.29	15960.12
Cardamone-SimpleGA	9563.08	5932.09	15495.17
Fernandez/Cotta/Fuentes	7553.21	6263.40	13816.61
Walz-PSO	8408.35	5336.88	13745.23
Garcia/Saez-PSO	8386.77	5021.41	13408.18
Munoz/Martin/Saez-EA	8167.60	4629.33	12796.93

the previous approaches because the optimizer has to present fast convergence and the ability to avoid local minima, since the competition just allows 100 fitness function evaluations. For future work, we expect to employ other swarm algorithms.

REFERENCES

[1] D. Loiacono, J. Togelius, P. Lanzi, L. Kinnaird-Heether, S. Lucas, M. Simmerson, D. Perez, R. Reynolds and Y. Saez. "The WCCI 2008 simulated car racing competition". In *IEEE Symposium On Computational Intelligence and Games, 2008. CIG '08*, pp. 119 –126, december 2008.

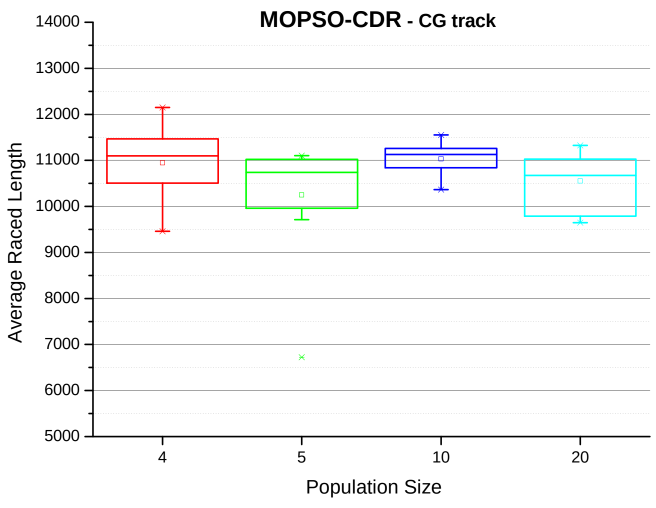
[2] C. A. C. Coello, G. B. Lamont and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer, New York, second edition, 2007.

[3] J. Muandoz, G. Gutierrez and A. Sanchis. "Multi-objective evolution for Car Setup Optimization". In *2010 UK Workshop on Computational Intelligence (UKCI)*, pp. 1 –5, september 2010.

[4] C. A. C. Coello and M. Reyes-Sierra. "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art". *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[5] C. A. C. Coello, G. T. Pulido and M. S. Lechuga. "Handling multiple objectives with particle swarm optimization". *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256 – 279, june 2004.

[6] R. A. Santana, M. R. Pontes and C. J. A. Bastos-Filho. "A Multiple Objective Particle Swarm Optimization Approach Using Crowding Distance and Roulette Wheel". In *Ninth International Conference on Intelligent Systems Design and Applications, 2009. ISDA '09*, pp. 237–242, december 2009.



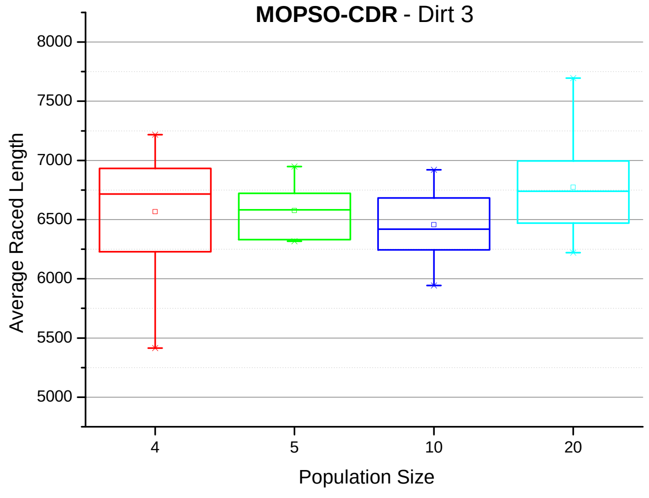


Figure 4: Box plot of the total raced length for the *MOPSO-CDR* algorithm for 4, 5, 10 and 20 particles.

[7] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari and K. Ziarati. "A multi-objective Artificial Bee Colony for optimizing multi-objective problems". In *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 5, p. 277. IEEE, 2010.

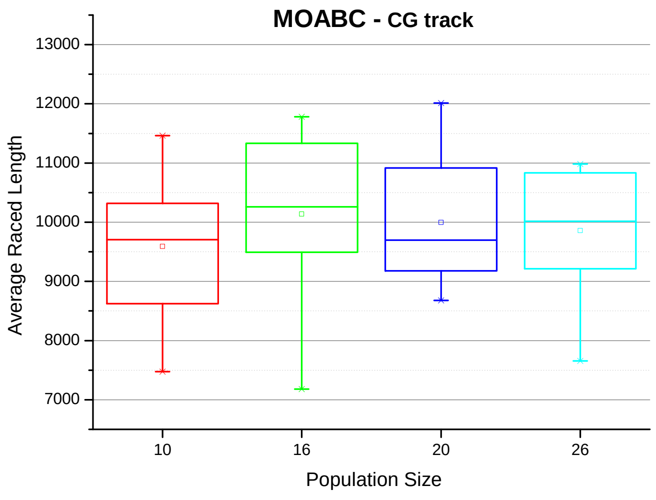
[8] J. Kennedy and R. C. Eberhart. *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[9] J. Kennedy and R. C. Eberhart. "Particle swarm optimization". In *IEEE International Conference on Neural Networks*, volume 4, pp. 1942–1948, 1995.

[10] C. Raquel and P. Naval Jr. "An effective use of crowding distance in multiobjective particle swarm optimization". In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 257–264. ACM New York, NY, USA, 2005.

[11] C. Tsou, S. Chang and P. Lai. *Using Crowding Distance to Improve Multi-Objective PSO with Local Search - Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*. IN-Tech Education and Publishing, 2007.

[12] D. Karaboga and B. Basturk. "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems". *Foundations of Fuzzy Logic and Soft Computing*, pp. 789–798, 2007.



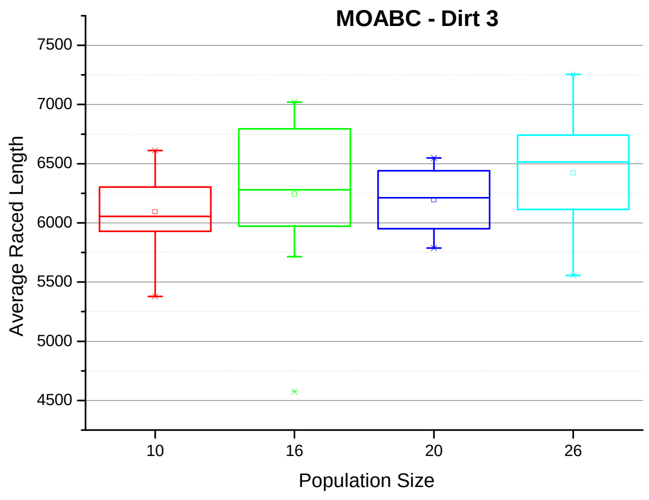


Figure 5: Box plot of the total raced length for the *MOABC* algorithm for 10, 16, 20 and 26 bees.