GRAND TOUR ALGORITHM: NEW SWARM-BASED OPTIMIZATION

Gustavo Meirelles[1], Bruno Brentan[2], Joaquín Izquierdo[3], and Edevar Luvizotto Junior[4]

[1]Universidade Federal de Minas Gerais
[2]UFMG
[3]Universitat Politecnica de Valencia
[4]Universidade Estadual de Campinas - Campus Cidade Universitaria Zeferino Vaz

May 6, 2020

Abstract

Metaheuristic algorithms based on the collective behavior of nature social groups, such as ants and bees, have been widely explored to solve many optimization problems in engineering and other sciences. The processing time and the chance to end up in a local optimal solution are drawbacks of these algorithms, and none has proved to outperform the others. In this paper, an improved swarm optimization technique, named Grand Tour Algorithm (GTA), based on the behavior of a peloton of cyclists, is introduced and applied to sixteen benchmarking optimization problems in the literature to evaluate its performance in comparison to the original particle swarm optimization (PSO) algorithm. Most of these benchmarking problems are tackled with a number of 20,000 variables, a really huge number inspired in the human genome. Under these conditions, GTA clearly outperforms the classical PSO. In addition, various sensitivity analyses are performed to verify the influence of the initial parameters in the GTA efficiency. It can be demonstrated that the GTA fulfils such coveted main aspects of an optimization algorithm as ease of implementation, speed of convergence, and reliability, thus confirming GTA's improved performance.

GRAND TOUR ALGORITHM: NEW SWARM-BASED OPTIMIZATION

Gustavo Meirelles [1*], Bruno Brentan[2], Joaquín Izquierdo [3], Edevar Luvizotto Jr. [4]

[1] Department of Hydraulic Engineering and Water Resources - ERH, Universidade Federal de Minas Gerais, 31270-901, Belo Horizonte, Brazil; gustavo.meirelles@ehr.ufmg.br

[2] Department of Hydraulic Engineering and Water Resources - ERH, Universidade Federal de Minas Gerais, 31270-901; brentan@ehr.ufmg.br

[3] Fluing-Institute for Multidisciplinary Mathematics, Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain; jizquier@upv.es

[4] Department of Water Resources - DRH, Universidade Estadual de Campinas, 13083-889, Campinas, Brazil; edevar@fec.unicamp.br

ABSTRACT: Metaheuristic algorithms based on the collective behavior of nature social groups, such as ants and bees, have been widely explored to solve many optimization problems in engineering and other sciences. The processing time and the chance to end up in a local optimal solution are drawbacks of these algorithms, and none has proved to outperform the others. In this paper, an improved swarm optimization technique, named Grand Tour Algorithm (GTA), based on the behavior of a peloton of cyclists, is introduced and applied to sixteen benchmarking optimization problems in the literature to evaluate its performance in comparison to the original particle swarm optimization (PSO) algorithm. Most of these benchmarking

problems are tackled with a number of 20,000 variables, a really huge number inspired in the human genome. Under these conditions, GTA clearly outperforms the classical PSO. In addition, various sensitivity analyses are performed to verify the influence of the initial parameters in the GTA efficiency. It can be demonstrated that the GTA fulfils such coveted main aspects of an optimization algorithm as ease of implementation, speed of convergence, and reliability, thus confirming GTA's improved performance.

KEYWORDS: optimization, swarm optimization, benchmarking problems.

INTRODUCTION

Many optimization problems in engineering are of a very complex nature and must be solved subjected to various sometimes complicated constraints. As a consequence, the achievement of an optimal solution is often hard. In addition, frequently, a large number of mixed variables, and differential and non-linear equations are used to describe the problem. As a result, in many cases, classical optimization procedures based on differential methods cannot be used. Metaheuristic techniques arise to bridge this gap, as they can explore the search space for optimal and feasible solutions in a less restrictive (derivative-free) framework. Since in continuous problems the set of feasible solutions is infinite, metaheuristic algorithms use an empirical iterative search method, mostly based on natural intelligence, to guide the search in a way that the solution is expected to always improve.

Among the various approaches utilized by metaheuristic algorithms, swarm intelligence uses social features of a group to define the behavior of its individuals, leading them to local optimal solutions [1]. In general, the position of an individual represents a potential solution, and has a defined score based on the objective function of interest. First, initial positions for individuals are defined randomly, and this defines their initial score. Then an individual evolves according to: i) its own preferences, based on its past experience, i.e., on its best performance ever achieved, and ii) according to the group experience, i.e., the best solution ever found by any individual in the group. Each algorithm has its own rules to express this social behavior. Additional random coefficients are used to guarantee better exploration of the search space, especially in the initial period, and those coefficients are also modified along iteration [2] to achieve better exploitation, mainly at the end of the search.

Examples are algorithms based on ants [3] and bees [4] looking for food, the breeding behavior of cuckoos [5], the behavior of birds flocking [6], and multi-agent theory [7], among many others.

One of the major issues of metaheuristic algorithms is the fact that there is no guarantee that the global optimal solution is achieved. Despite the randomness deployed to build the initial solutions and during iteration, the search can be easily biased and the final result may be just a point close to a local optimum [8]. In addition, to account for possible constraints of the problem, in a single-objective setting, penalty functions have to be used, adding a value to the objective function to hinder unfeasible solutions [9]. If the added value is too high, optimal solutions on the boundaries can be disregarded because of the steep path this represents for the individuals and the strong deformation of the objective function close to the boundary. On the other hand, if the penalty is too soft, unfeasible solutions can be considered as optimal [10]. The randomness of the process and the penalties added can lead to inconsistent results, so that multiple runs of the algorithm are necessary to verify the quality of the solution.

Another crucial issue of these methods is the computational effort required. Multiple solutions (individuals) must be evaluated in each iteration. This number varies according to the number of variables of the problem, and can be very large. In general, the number of individuals has to be at least the same as the number of variables [11]. Thus, in complex problems, the number of objective functions evaluations can be very large. In addition, many engineering problems use external models to calculate parameters of the objective function and the constraints, as is the case of hydraulic [12], thermal [13], and chemical models [14], thus increasing even more the computational effort.

A final crucial idea is that there is not such a thing as the best metaheuristic algorithm. Each of them has advantages and disadvantages regarding three major aspects [15]: i) ease of implementation due to the number of parameters to be adjusted (sometimes costly fine-tuned); ii) computational complexity, which reflects on the convergence speed; and iii) reliability of the results, with consistent optimal values obtained through a series of tests.

As an attempt to help in the solution of these problems, a new swarm-based algorithm that uses the metaphor of the cyclists' behavior in a peloton, which we have named Grand Tour Algorithm (GTA), is proposed in this paper. The main features of the algorithm are: i) the drag, defined according to the distance to the leading cyclist (embodying the best solution), and ii) the speed, defined using the difference between two consecutive objective function evaluations. These two elements are used to calculate the coefficients that determine the route of each cyclist. As a result, the peloton will try to follow the cyclist closest to the finish line, i.e. the optimal solution, and also the cyclist who is going faster at this point. We describe the methodological aspects in the next section. Then, sixteen benchmarking functions, most of them with 20,000 decision variables, are used to evaluate the performance of the algorithm when compared with the classical Particle Swarm Optimization (PSO) algorithm [6]. Finally, various sensitivity analyses verify the relevance the algorithm parameters have in the optimization convergence and stability.

METHODOLOGY

In this section we present a few general ideas about iterative methods, then we concisely describe the classical PSO and, finally, fully present GTA.

Iterative optimization processes: a general view

Typically, an iterative optimization algorithm can be mathematically described as in Eq. (1):

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha \mathbf{d}_t, \quad (1)$$

where $\mathbf{x}_{t+1}$ is (the position of) a solution at time step $t+1$, α is the step size, and $\mathbf{d}_t$ is the direction of the movement.

Since there is no general *a priori* clue about which direction to take to progress towards the optimal solution, various mathematical properties are used to define the right track and the step size, better leading to optimal points.

First-order derivative methods use the opposite direction of the gradient vector with a certain step size to find minimal points. Despite first-order methods have fast convergence, sometimes, depending of the step size, an oscillatory process can start, and the optimal point is never found. The classical well-known gradient method, a first-order method, is shown in Eq. (2):

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \frac{\nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\|} \ . \quad (2)$$

Here $\nabla f(\mathbf{x}_t)$ is the gradient vector calculated at $\mathbf{x}_t$.

To improve the convergence of first-order methods, second-order methods use the opposite direction of the gradient vector improved by the information of the Hessian matrix, which provides second-order information. The Modified Newton method, for example, adopts a unitary step size, and the method is described by Eq. (3).

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \left[\nabla^2 f\left(\mathbf{x}_t\right)\right]^{-1} . \nabla f\left(\mathbf{x}_t\right) . \quad (3)$$

Even though derivative methods are mathematically efficient, the calculations of the Jacobian vector and the Hessian matrix are computationally hard or, simply, not possible in many cases.

To cope with this drawback, metaheuristic algorithms, many of them based on natural processes, are widely applied in real-world problems. Among them, swarm-based algorithms use the same search process described by equation (1). However, in contrast with derivative-based algorithms, in metaheuristic algorithms each solution is brought to a new position (solution) based on a set of rules that replace the derivative information with other kind of information, while providing a conceptually similar approach.

We first concisely present the classical PSO, since we are going to use it for comparisons purposes.

Particle Swarm Optimization (PSO)

The Particle Swarm Optimization algorithm [6] is one of the most popular swarm-optimization techniques. Each particle represents a candidate solution and has its own position X and velocity V. Initially, these values are randomly defined according to the boundaries of the problem. As the time (iterations) goes by, the particles move according to its velocity and reach a new position. This velocity of a particle, see equation (4), is a linear combination of: the particle's inertia, through coefficient ω· its memory, through coefficientc_1; and some social interaction with the group, through coefficient c_2. The best position found by the group, G , and the best position ever reached by the particle, P , are used to update its velocity. In addition, random values, $\text{rand}_1$and rand_2, are used to boost good exploitation abilities of the search space. Equations (4) and (5), used to update the particles' positions, describe the complete iteration process.

$$V_i^{k+1} = \omega \bullet V_i^k + c_1 \bullet \text{rand}_1 \bullet \frac{\left(P_i^k - X_i^k\right)}{t} + c_2 \bullet \text{rand}_2 \bullet \frac{\left(G^k - X_i^k\right)}{t} \qquad (4)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \bullet t \qquad (5)$$

Grand Tour Algorithm

The principle of the Grand Tour Algorithm (GTA) is similar to most swarm-based algorithms': each cyclist represents a possible solution with position and velocity updated along the iteration.

GTA fundamentals

The difference hinges on how the velocity is updated. Instead of the social and cognitive aspects, the power spent by cyclists is the main feature for this update. Eq. (6) shows how the power, P, of a cyclist can be calculated according to its speed, S.

$$P = \frac{(F_g + F_f + F_d)S}{1-l} \qquad (6)$$

Three main forces are considered for this calculation: gravitational,F_g; friction, F_f; and drag, F_d. In GTA, friction is disregarded, as it represents only a small fraction of the power; also, the losses, l, mainly represented by the friction of the chain and other mechanical components of the bike, are negligible.

Obviously, the speed used in this equation has no relationship with the velocity used to update a particle's position. Here, the speed is calculated through Eq. (7), representing the cyclist ascending or descending (vertical) speed, using two consecutive values of the cyclist's objective function (OF).

$$S = \frac{\mathrm{OF}_{k+1} - \mathrm{OF}_k}{t} \qquad (7)$$

So, if the objective function is reduced, the cyclist is closer to the finish line, i.e., to the optimal solution. Larger difference of the objective function between two successive iterations means that the cyclist is fast approaching the finish line.

Calculation of the drag force

The drag force can be calculated by Eq. (8).

$$F_d = 0.5 \bullet C_d \bullet A \bullet \rho \bullet (S+W)^2 . \qquad (8)$$

Here C_d is the drag coefficient, A is the frontal area of the cyclist, ρ is the air density, S the cyclist vertical speed, and W the wind speed. Frontal area and air density may be considered the same for every cyclist, both herein taken as 1. The wind speed is considered to be null.

The drag coefficient, C_d, is the major component of this equation. As shown in an aerodynamic model for a cycling peloton using CFD simulation [16], in a cycling peloton the front cyclists have to pedal with much more power than those in the back of the peloton due to the drag produced. Thus, the cyclist who achieves the best solution is the leader and, according to how far the others are behind, they are more or less benefited from the leader's drag. In this sense, cyclists far from the leader find it easy (and are bound) to follow him or her because their paths require less power, while the cyclists closest to the leader are freer because the power difference in following the leader or not is less relevant. Using the physical results obtained by [16], the drag coefficient may be estimated by creating a ranking of the cyclists according to their OF values. The leader has no benefit from the peloton, and has a drag coefficient equal to one. The rear cyclist is the most benefited and has a drag coefficient of only 5% of the leader. The drag coefficient of the remaining cyclists is obtained by means of a linear interpolation between the first and the last cyclists (Equation 9).

$$C_d = 1 - 0.95 \frac{OF - \mathrm{OF}_{\mathrm{best}}}{\mathrm{OF}_{\mathrm{worst}} - \mathrm{OF}_{\mathrm{best}}} . \qquad (9)$$

In this expression, OF is the cyclist objective function value, OF$_{\mathrm{best}}$is the leader objective function, andOF$_{\mathrm{worst}}$ is the value of the objective function of the last cyclist in the peloton.

Calculation of the gravitational force

The gravitational force has a great impact on the speed of convergence of the algorithm. It is calculated using Eq. (10).

$$F_g = g \bullet sin\left(\tan^{-1}(I)\right) \bullet m \qquad (10)$$

Here g is the acceleration of gravity, I is the slope, defined as the difference between the values for the objective function in two successive iterations, and m is the cyclist mass.

As previously described, the objective function represents the (vertical) distance to the finish line. And, as the race is downhill, towards the minimum, the objective function is also the elevation of the cyclist's place on the terrain with respect to the minimum. Thus, when cyclists are going downhill (negative slope), they need to put less power, because the gravity is aiding. In contrast, a cyclist's weight causes an opposite force to the movement. This procedure avoids cyclists to go uphill (positive slope), towards a worst solution, because they would spend too much power in this case.

The value of the gravitational acceleration is constant, and the cyclists' mass is randomly defined at the beginning, in a range from 50 to 80 kg. This procedure allows lightweight cyclists, known as climbers, not to be hardly affected by going uphill, i.e., with a direction towards a worse solution. This guarantees better search capabilities.

Velocity and position updating

The power component corresponding to each force is calculated separately and will be used to update the cyclist velocity according to Eq. (11), which is used, in turn, to define its new position, as in Eq. (5).

$$V_i^{k+1} = k_g \bullet V_i^k + k_d \bullet \text{rand}_1 \bullet \frac{(X_d - X_i^k)}{t} + k_g \bullet \text{rand}_2 \bullet \frac{(X_g - X_i^k)}{t} \tag{11}$$

The elements of this formula are calculated as follows. First, the values of the power spent by the cyclists, according to Eq. (6), are ranked. Then, the two weight coefficients for drag and gravitational power, k_d and k_g, shown in Eq. (11), are calculated by normalizing F_d and F_g between 0.5 and 1.0, where 1.0 represents the cyclist with the lowest power.

Regarding the drag component, k_d, this procedure reduces the step taken by the leading cyclists, which are closer to an optimal solution, thus resulting in a more specific local search around the leader position, X_d, while the latter cyclists are rapidly pushed towards this point to benefit from the leaders drag.

With the gravitational component, k_g, the purpose is to encourage the cyclists that are rapidly converging to a better solution,X_g, to keep going in this direction, while cyclists going uphill or slowly improving are kept to explore with a more local search focus around their place.

Finally, instead of an inertia coefficient, as used in PSO, the gravity coefficient k_g is also used to give a weight to the cyclist current direction. Random values are also used to avoid a biased search.

The flowchart in Figure 1 summarizes the GTA optimization process.

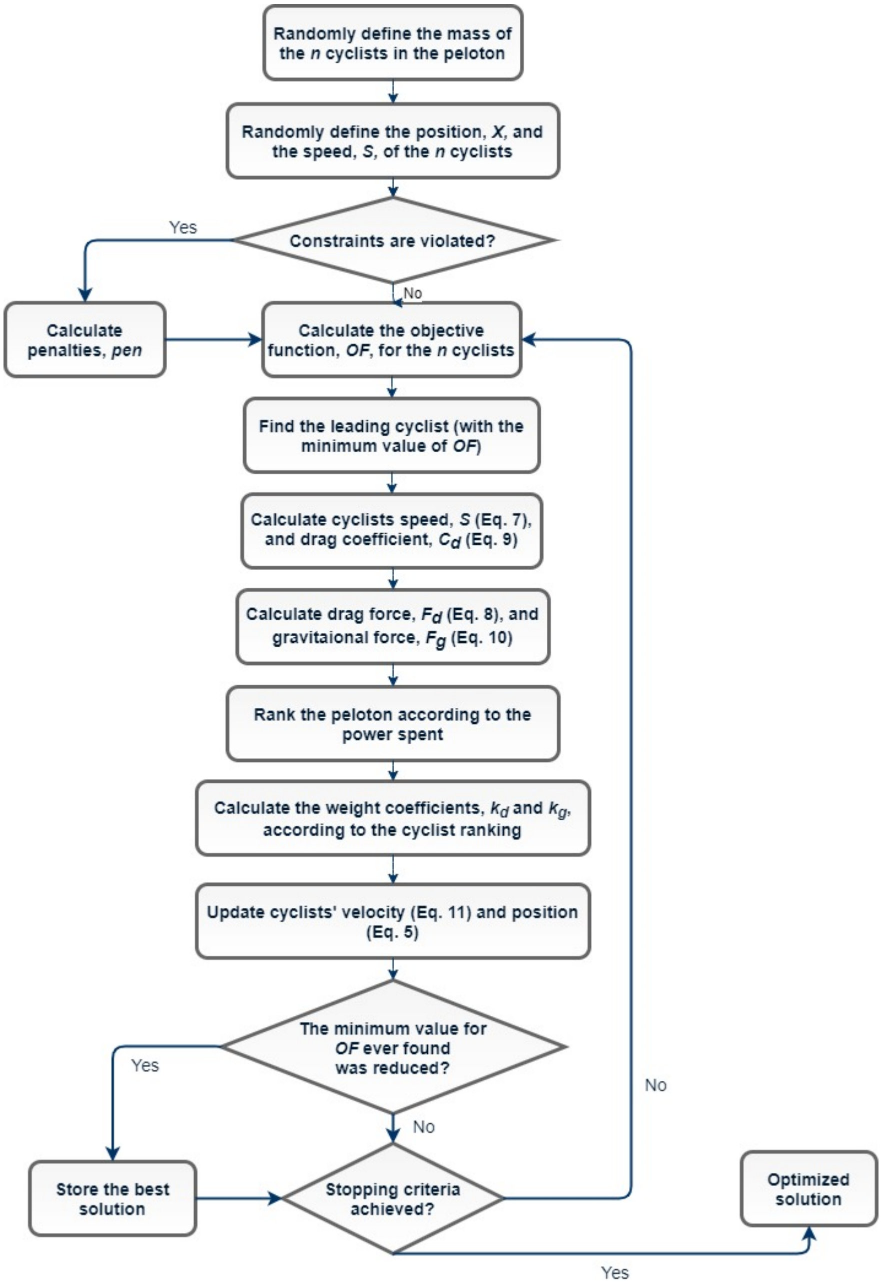

Randomly define the mass of the *n* cyclists in the peloton
Randomly define the position, *X*, and the speed, *S*, of the *n* cyclists
Constraints are violated?
Yes
No
Calculate penalties, *pen*
Calculate the objective function, *OF*, for the *n* cyclists
Find the leading cyclist (with the minimum value of *OF*)
Calculate cyclists speed, *S* (Eq. 7), and drag coefficient, *Cd* (Eq. 9)
Calculate drag force, *Fd* (Eq. 8), and gravitaional force, *Fg* (Eq. 10)
Rank the peloton according to the power spent
Calculate the weight coefficients, *kd* and *kg*, according to the cyclist ranking
Update cyclists' velocity (Eq. 11) and position (Eq. 5)
The minimum value for *OF* ever found was reduced?
Yes
No
No
Store the best solution
Stopping criteria achieved?
Optimized solution
Yes

Figure 1 – Flowchart of the Grand Tour Algorithm

Test conditions

The GTA algorithm is implemented in Matlab, and the comparison with PSO is performed using the built-in PSO algorithm from the *Global Optimization Toolbox* [17], with default parameters ($c_1 = c_2 =$ 1.49 and ω varying from 0.1 to 1.1, linearly).

To evaluate the performance of GTA against PSO, both algorithms were applied to sixteen well-known benchmark functions from the literature [18-19] and collections online of test functions, such as the library GAMS World [20], CUTE [21], and GO Test Problems [22]. In addition, [23] provides an exhaustive list of up 175 functions.

Table 1 shows the name and the equation of each function, the dimension we have used for the problem, and the boundaries considered for the variables. The minimum value for all of these functions is given in the last column. Let us note here that the dimension for most of these benchmark functions has been taken inspired in the human DNA, which may be represented by 20,000 genes [24-25].

Table 1 – Benchmark functions

Function	Equation	Dimension (n)	Search Space (x_{min}, x
Paraboloid	$f\ (\ x\ ) = x_1^2 + x_2^2$	2	[-100,100]
Sphere	$f\ (\ x\ ) = \sum_{i=1}^{n} x_i^2$	20,000	[-100, 100]
Rosenbrock	$f\left(x\right) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	20,000	[-30, 30]
Rastrigin	$f\left(x\right) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right]$	20,000	[-5.12, 5.12]
Griewank	$f\left(x\right) = \frac{1}{4000} \sum_{i=1}^{n} \left[x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} + 1\right)\right]$	20,000	[-600, 600]
Shaffer f6	$f\left(x\right) = 0.5 - \frac{\left(\sin\sqrt{x_1^2+x_2^2}\right)^2 - 0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}$	2	[-100, 100]
Alpine	$f\left(x\right) = \sum_{i=1}^{n} \left[x_i \sin\left(x_i\right) + 0.1x_i\right]$	20,000	[-10, 10]
Brown	$f\left(x\right) = \sum_{i=1}^{n-1} \left[\left(x_i^2\right)^{x_{i+1}^2+1} + \left(x_{i+1}^2\right)^{x_i^2+1}\right]$	20,000	[-1, 1]
Chung Reynolds	$f\left(x\right) = \left(\sum_{i=1}^{n} \left[x_i^2\right]\right)^2$	20,000	[-100, 100]
Dixon Price	$f\left(x\right) = \left(x_1 - 1\right)^2 + \sum_{i=2}^{n} \left[i\left(2x_i^2 - x_{i-1}\right)^2\right]$	20,000	[-10, 10]
Exponential	$f\left(x\right) = -exp\left(-0.5\sum_{i=1}^{n} \left[x_i^2\right]\right)$	20,000	[-1, 1]
Salomon	$f\left(x\right) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} \left[x_i^2\right]}\right) + 0.1\sqrt{\sum_{i=1}^{n} \left[x_i^2\right]}$	20,000	[-100, 100]
Schumer Steiglitz	$f\left(x\right) = \sum_{i=1}^{n} \left[x_i^4\right]$	20,000	[-100, 100]
Sum of Powers	$f\left(x\right) = \sum_{i=1}^{n} \left[\left\|x_i\right\|^{i+1}\right]$	20,000	[-1, 1]
Sum of Squares	$f\left(x\right) = \sum_{i=1}^{n} \left[ix_i^2\right]$	20,000	[-1, 1]
Zakharov	$f\left(x\right) = \sum_{i=1}^{n} \left[x_i^2\right] + \left(\sum_{i=1}^{n} \left[0.5ix_i\right]\right)^2 + \left(\sum_{i=1}^{n} \left[0.5ix_i\right]\right)^4$	20,000	[-10, 10]

RESULTS

Performance

The sixteen optimization problems presented were solved with GTA using 500 cyclists and considering 1000 as the maximum number of iterations allowed. Figure 2 shows the boxplot analysis for the 100 optimizations performed. In most of the cases the deviation is so low that it is not possible to observe the height of the boxplot in the figure. For the Rosenbrock and Dixon Price functions a local optimal solution was invariably found, but the minimum value was never reached. The number of evaluations of the objective function for all the cases was, in average, high, close to the maximum of 500,000. However, since no improvement tolerance was used to stop the optimization, after just a few iterations the gains soon become negligible, as can be seen in Figure 3. Thus, the three fundamental features of an optimization procedure, namely, few parameters to adjust, fast convergence, and reliability of the results, are satisfied for the studied cases.

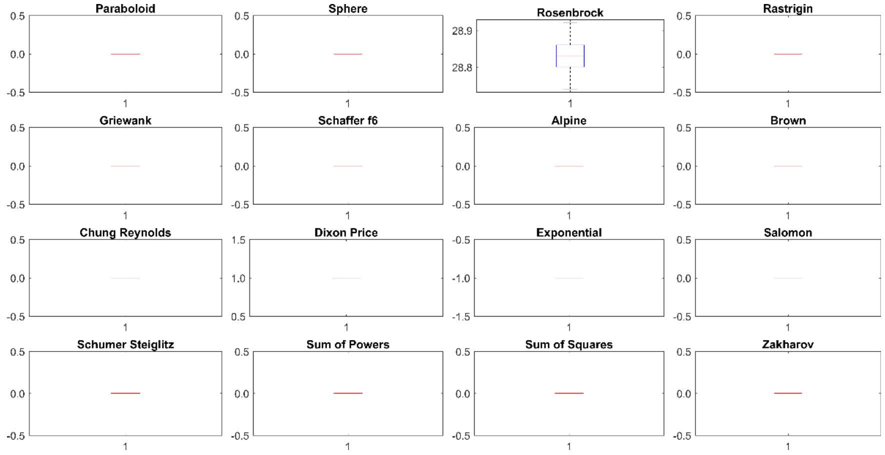


Figure 2 – Boxplot analysis of the GTA performance for the sixteen benchmarking functions; values obtained for eachOF are represented on the vertical axis.

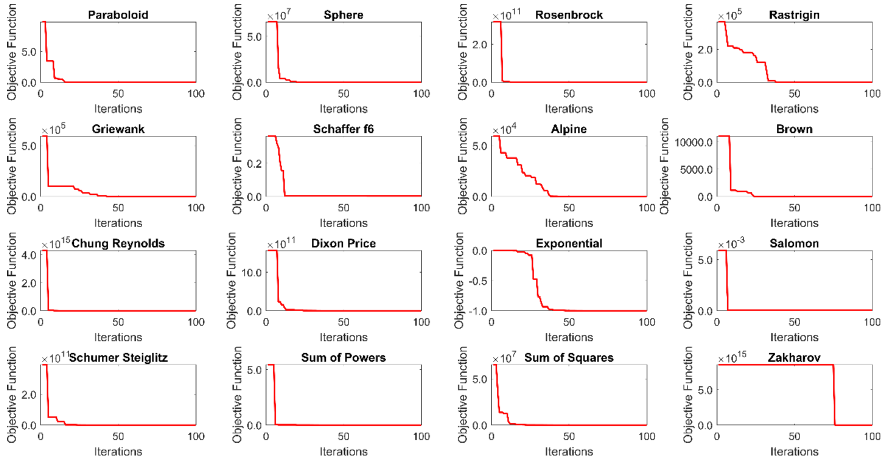


Figure 3 – GTA Objective function evolution for the sixteen benchmarking functions

For comparison purposes, the same functions were tested once using PSO, with a higher value, namely 3,000, for the maximum number of allowed iterations, but with the same number of cyclists (particles), 500. Table 2 shows the results obtained. The global minimum was achieved only in three cases: Paraboloid, Shaffer f6 and Salomon. Moreover, a higher number of evaluations were necessary to achieve these results. In addition, it can be observed that in most cases there was no convergence, even with a higher number of iterations, reaching the maximum number of evaluations possible. The extremely high dimension of the problems, with most of them depending on 20,000 decision variables, is the major contributor for this behavior. Many more iterations and many more particles could have been used to reach (perhaps) an optimal solution; however, the increase in the computational effort would had been huge (or unaffordable).

Table 2 – PSO results for the sixteen benchmarking functions

Function	Number of OF evaluations	Best Solution
Paraboloid	23,500	0
Sphere	1,500,500	2.97E+07
Rosenbrock	1,500,500	9.05E+10

Function	**Number of** OF **evaluations**	**Best Solution**
Rastrigin	1,500,500	256,392
Griewank	1,500,500	266,962
Shaffer f6	25,500	0.0025
Alpine	1,500,500	35,760
Brown	1,500,500	4,824
Chung Reynolds	1,500,500	8.35E+14
Dixon Price	1,500,500	4.17E+11
Exponential	11,000	0
Salomon	12,500	0
Schumer Steiglitz	1,500,500	1.17E+11
Sum of Powers	24,500	4
Sum of Squares	1,500,500	2.73E+07
Zakharov	11,000	695,675

Sensitivity Analysis

We report here three tests. The first test is performed to evaluate the relevance of the number of cyclists and the maximum number of iterations in the optimization process. Figure 4-a shows that the increase in the numbers of cyclists and maximum number of iterations has no significant influence on the reduction of the objective function. Only a combination of low numbers of cyclists and iterations resulted in poor results. As expected, Figure 4-b shows the increase of the number of objective function evaluations when both parameters are increased. Therefore, the algorithm reaches the best results after an extensive searching, but, as observed in Figure 3, with no significant changes. These results agree with the ones obtained previously, showing that GTA has a fast convergence to a feasible and high-quality solution.

(a)

(b)

Figure 4 – Sensitivity analysis for the number of cyclists and the maximum number of iterations: a) Objective function variation; b) Number of objective function evaluations

The second sensitivity analysis focused on the variation of the cyclists' mass. To this purpose, the minimum and maximum values considered were changed in a range from 25 to 125 kg. The results showed no significant difference. Only when all the cyclists had the same weight, regardless the value, the optimization became slower and with worse results. This lack of randomness in the cyclists' weight hampers the exploration of points near the boundaries, since there are no lightweight cyclists, who are less affected by the gravitational force, to venture closer to the boundaries because of the penalties.

Finally, the score used to calculate the k_g and $k_d$coefficients of the ranked cyclists according to the power spent in each iteration was evaluated. The best cyclist, or the one who spends less energy, receives the higher score and the worst receives the minimum. Figure 5 shows that only with a combination of high values, the results of the objective function got worse. In terms of the number of evaluations of the objective function, no significant difference was observed. In all the analyses made, the GTA presented consistent results despite changes in its parameters. This robustness is very interesting for the user, since there is no need of heavy fine-tuning to adjust the default parameters so as to perform a better optimization.

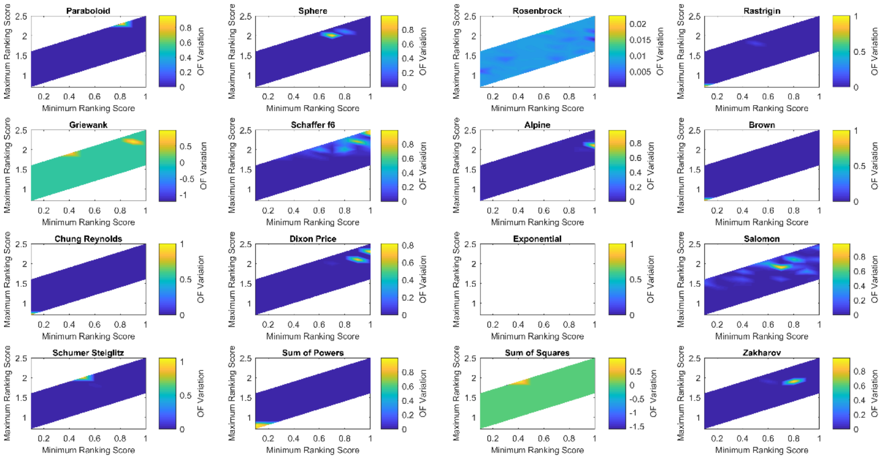


Figure 5 – Sensitivity analysis for the score of parameters$\mathbf{k_g}$ and $\mathbf{k_d}$

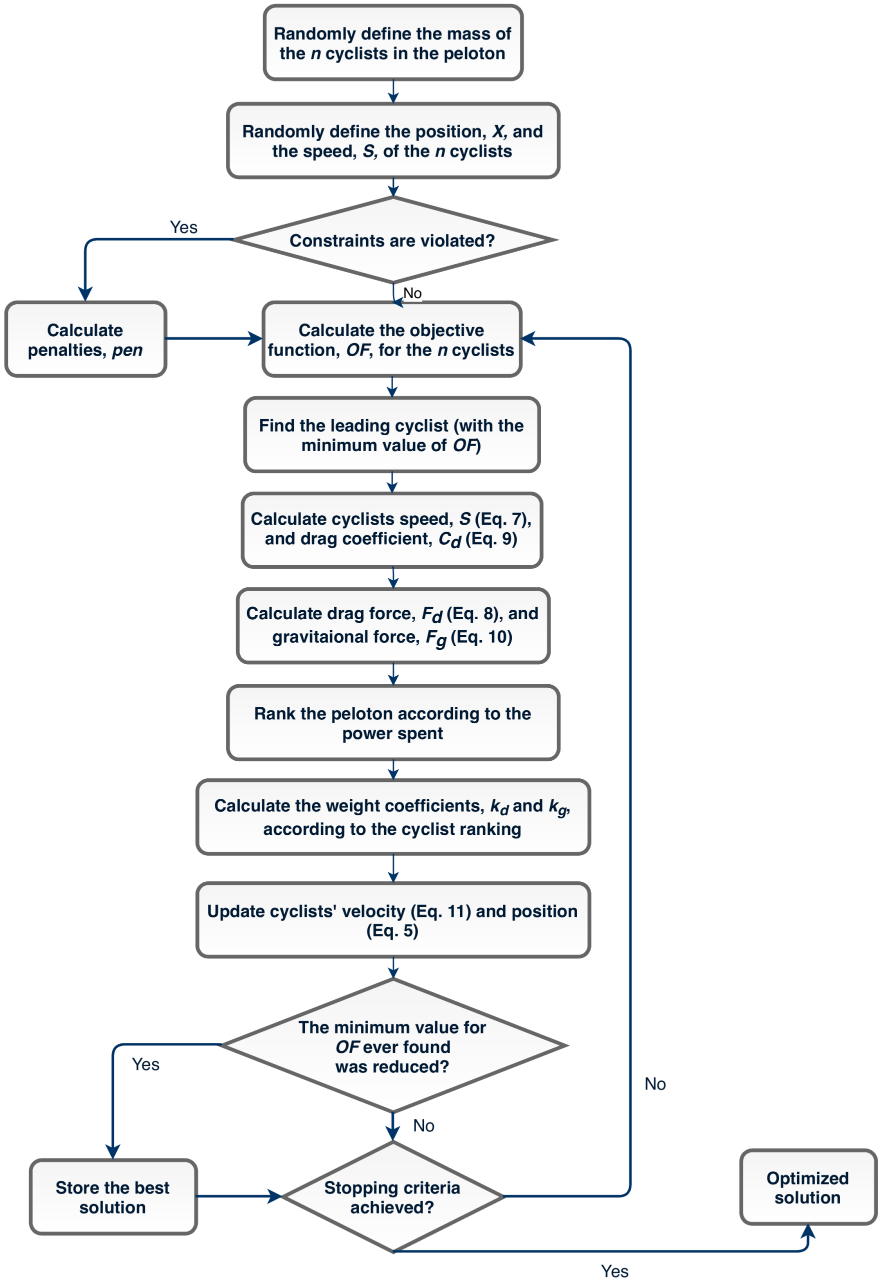
Conclusion

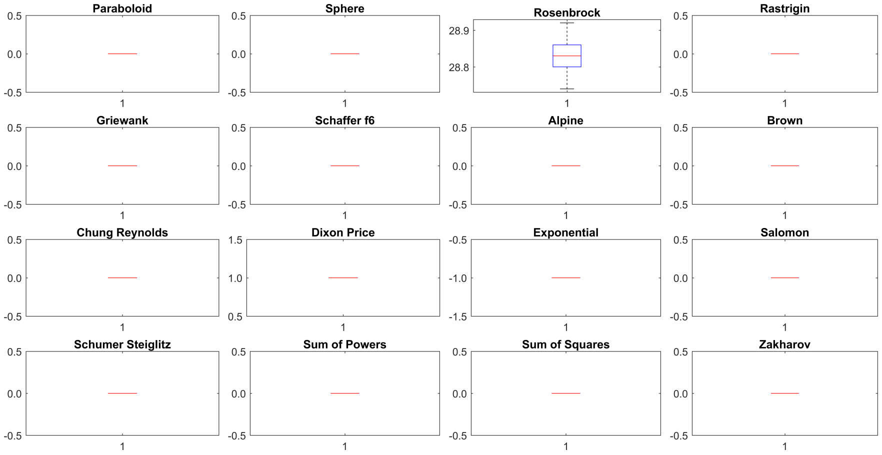
As a swarm-based algorithm, GTA exhibits good results in terms of consistency and speed to find the optimal solution for the benchmarking problems considered. GTA performance is clearly superior to PSO in terms of speed and consistency when functions with a huge (human genome-like) search space (up to 20,000 decision variables) were tested. The great advantage of GTA is its ability to direct the cyclists to search the fastest path for improvement of the objective function. At the same time, points with worst values for the objective functions are locally explored, since they can be just a barrier for a big improvement. The search for the fastest path has a similar goal as the first derivative, hence its fast convergence, despite the number of cyclists and iterations used. Finally, the sensitivity analysis showed the robustness of GTA, since the change in the main parameters had little impact in the final result. This feature results in a user-friendly algorithm, because there is no concern in adjusting the default parameters to perform a good optimization. Therefore, the three main aspects desirable for an optimization algorithm – ease of implementation, speed of convergence and reliability – had good results, confirming the expected improvements of GTA.

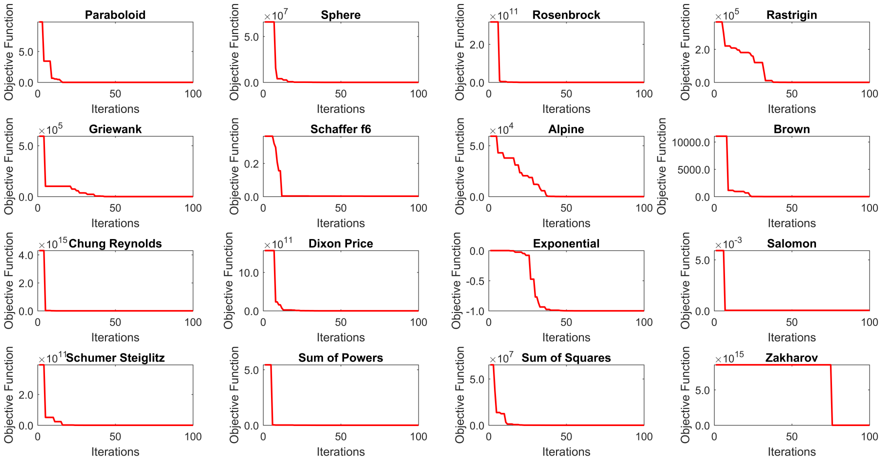
References

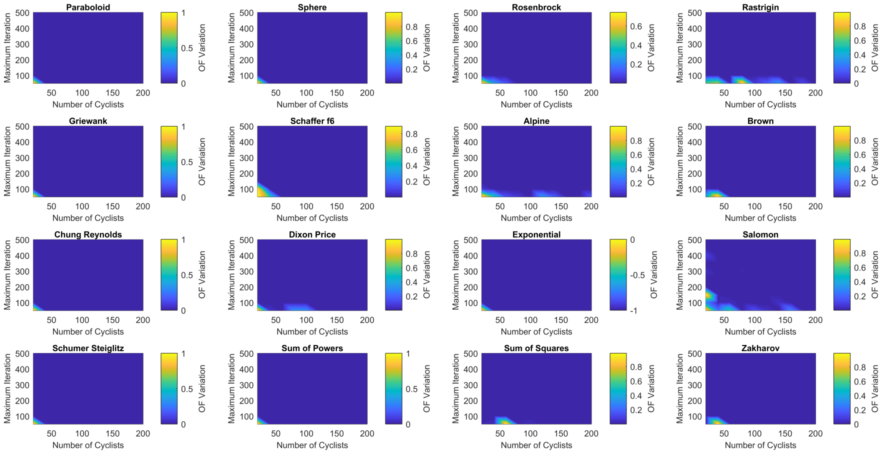
1. Mirjalili S, Lewis A. The whale optimization algorithm. *Adv Engrg Soft.* 2016; 95:51-67.
2. Chatterjee A, Siarry P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Comput Oper Res.*2006; 33(3):859-871.
3. Dorigo M, Blum C. Ant colony optimization theory: A survey. *Theor Comput Sci.* 2005; 344(2-3):243-278.
4. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J Global Optim.* 2007; 39(3):459-471.
5. Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engrg Comput.* 2013; 29(1):17-35.
6. Kennedy J, Eberhart R. Particle swarm optimization (PSO). In *Proc. IEEE Intern Conf Neural Net, Perth, Australia* , 1995 (November) pp. 1942-1948.
7. Montalvo I, Izquierdo J, Pérez-Garcia R, Herrera M. Water distribution system computer-aided design by agent swarm optimization. *Comput Aided Civil Infrastr Engrg.* 2014; 29(6):433-448.
8. Gonzalez-Fernandez Y, Chen S. Leaders and followers—a new metaheuristic to avoid the bias of accumulated information. In *2015 IEEE Congr Evol Comput. (CEC)* 2015 (May) pp. 776-783. IEEE.
9. Parsopoulos KE, Vrahatis MN. Particle swarm optimization method for constrained optimization problems. *Intell Tech Theory Appl: New Trends in Intell Tech.* 2002; 76(1):214-220.
10. Wu ZY, Simpson AR. A self-adaptive boundary search genetic algorithm and its application to water distribution systems. *J Hydr Res.*2002; 40(2):191-203.

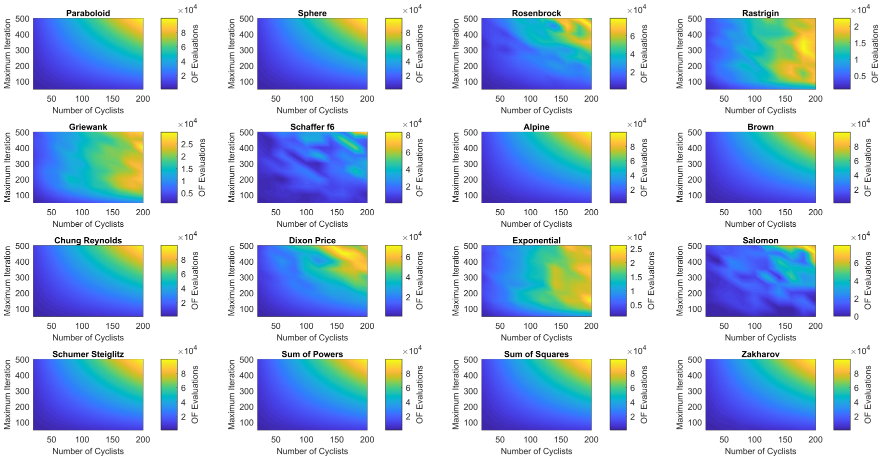
11. Trelea IC. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inform Process Letters*2003; 85(6):317-325.
12. Brentan B, Meirelles G, Luvizotto Jr E, Izquierdo J. Joint operation of pressure-reducing valves and pumps for improving the efficiency of water distribution systems. *J Water Res Plan Manag.*2018; 144(9):04018055.
13. Freire RZ, Oliveira GH, Mendes N. Predictive controllers for thermal comfort optimization and energy savings. *Ener Build.*2008; 40(7):1353-1365.
14. Banga JR, Seider WD. Global optimization of chemical processes using stochastic algorithms. In *State of the art in global optimization* (pp. 563-583). Springer, Boston, MA, 1996.
15. Maringer DG. *Portfolio management with heuristic optimization* (Vol. 8). Springer Science & Business Media 2006.
16. Blocken B, van Druenen T, Toparlar Y, Malizia F, Mannion P, Andrianne T, . . . , Diepens J. Aerodynamic drag in cycling pelotons: new insights by CFD simulation and wind tunnel testing. *J Wind Engrg Ind. Aerod.* 2018; 179:319-337.
17. MATLAB 2018, The MathWorks, Inc., Natick, Massachusetts, United States.
18. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans Evol Comput.* 2002; 6(1):58-73.
19. Eberhart RC, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proc 2000 Congr Evol Comput. CEC00 (Cat. No. 00TH8512)* (Vol. 1, pp. 84-88). IEEE, 2000, July.
20. GAMS World, GLOBAL Library, Available online: http://www.gamsworld.org/global/globallib.html
21. Gould NIM, Orban D, Toint P.L. CUTEr, A Constrained and Un-constrained Testing Environment, Revisited, Available online: http://cuter.rl.ac.uk/cuter-www/problems.html
22. GO Test Problems, Available online: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm
23. Jamil M, Yang XS. A literature survey of benchmark functions for global optimisation problems. *Intern J Math Model Num Optim.*2013; 4(2): 150–194.
24. Sharma G. The Human Genome Project and its promise. *J Indian College Cardiol.* 2012; 2(1):1–3.
25. Li W. On parameters of the human genome. *J Theor Biol.* 2011; 288:92–104.

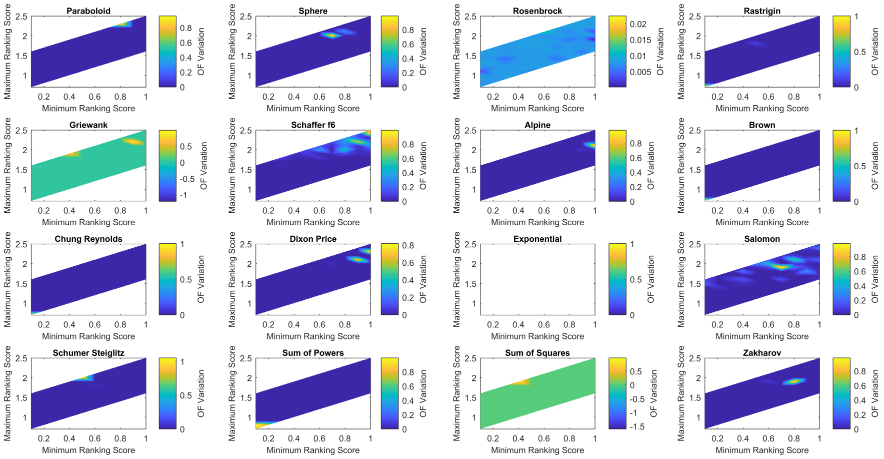

Randomly define the mass of the *n* cyclists in the peloton
Randomly define the position, *X,* and the speed, *S,* of the *n* cyclists
Constraints are violated?
Yes
No
Calculate penalties, *pen*
Calculate the objective function, *OF*, for the *n* cyclists
Find the leading cyclist (with the minimum value of *OF*)
Calculate cyclists speed, *S* (Eq. 7), and drag coefficient, C_d (Eq. 9)
Calculate drag force, F_d (Eq. 8), and gravitaional force, F_g (Eq. 10)
Rank the peloton according to the power spent
Calculate the weight coefficients, k_d and k_g, according to the cyclist ranking
Update cyclists' velocity (Eq. 11) and position (Eq. 5)
The minimum value for *OF* ever found was reduced?
Yes
No
No
Store the best solution
Stopping criteria achieved?
Optimized solution
Yes


Paraboloid
Sphere
Rosenbrock
Rastrigin
Griewank
Schaffer f6
Alpine
Brown
Chung Reynolds
Dixon Price
Exponential
Salomon
Schumer Steiglitz
Sum of Powers
Sum of Squares
Zakharov
28.9
28.8
0.5
0.0
-0.5
1.5
1.0
-1.0
-1.5
1


Paraboloid
Sphere
×10^7
Rosenbrock
×10^11
Rastrigin
×10^5
Griewank
×10^5
Schaffer f6
Alpine
×10^4
Brown
Chung Reynolds
×10^15
Dixon Price
×10^11
Exponential
Salomon
×10^-3
Schumer Steiglitz
×10^11
Sum of Powers
Sum of Squares
×10^7
Zakharov
×10^15
Objective Function
Iterations
0
50
100


Paraboloid
Sphere
Rosenbrock
Rastrigin
Griewank
Schaffer f6
Alpine
Brown
Chung Reynolds
Dixon Price
Exponential
Salomon
Schumer Steiglitz
Sum of Powers
Sum of Squares
Zakharov
Maximum Iteration
Number of Cyclists
OF Variation
50
100
150
200
300
400
500


Paraboloid
Sphere
Rosenbrock
Rastrigin
Griewank
Schaffer f6
Alpine
Brown
Chung Reynolds
Dixon Price
Exponential
Salomon
Schumer Steiglitz
Sum of Powers
Sum of Squares
Zakharov
Maximum Iteration
Number of Cyclists
OF Evaluations
×10^4


Paraboloid
Sphere
Rosenbrock
Rastrigin
Griewank
Schaffer f6
Alpine
Brown
Chung Reynolds
Dixon Price
Exponential
Salomon
Schumer Steiglitz
Sum of Powers
Sum of Squares
Zakharov
Maximum Ranking Score
Minimum Ranking Score
OF Variation