Genetic Roach Infestation Optimization - Optimization Algorithm based on Cockroaches' social life

António Gustavo Lança Brites

antonio.brites@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

November 2017

Abstract

This paper presents a novel adaptation of Particle Swarm Optimization (PSO) algorithm based on Roach Infestation Optimization (RIO). This algorithm is inspired on cockroaches' scientifically demonstrated behaviours. Cockroaches have been able to survive over the centuries and that is only possible due to their capacity to live in community and share information, such like dark spots to hide and food locations. Genetic Roach Infestation Optimization (GRIO) is more tied to roaches' life than RIO. It uses a multi-swarm and self-adaptive strategy to avoid falling into local optimum and converging too soon. Using a self-adaptive inertia weight, it's possible to explore a global search, or a local one, according to the algorithm feedback. This algorithm allows, in the same space, the presence of two different families of cockroaches that behave independently. To keep the information balanced between both families, reproduction behaviour (based on a regular Genetic Algorithm) is carried out. In that reproduction, babies will have a crossover of parents' information. To preserve diversity some mutations are enabled. Afterwards, the proposed algorithm is tested in some multimodal and hard benchmark functions in comparison to other algorithms. Results clearly support that cockroaches' life adapted into PSO algorithm greatly improves optimization results.

I. Introduction

Groups of birds, insects and fishes are able to synchronize velocities, turning manoeuvres and to perform several group activities (landing for example). Such behaviour has intriguing several authors that have been trying to understand the way that kind of social behaviour works.

Craig [1] was the first author presenting some work in flocks simulation. He presented this phenomena as three simple rules: collision avoidance - since it is impossible that two particles (bird, fish, insect among others) exists in the same position, at the same time, every single one avoids to crash with other flock mates; velocity matching - each flock's member attempts to match his own velocity with nearby mates; and flock centering - each particle tries to keep close to nearby mates.

Kennedy and Eberhart, in 1995 [2], pioneered an entire new approach to optimization problems when, inspired in real world swarm intelligence, the authors developed a new optimization algorithm based on some behaviours of birds flocking, Particle Swarm Optimization (PSO). In order to improve its performance, a lot of PSO variations were developed and studied (some examples of those variations are available in the Master Thesis *Genetic Roach Infestation Optimization)*. The aim goal is always to achieve global optima in the less possible iterations, to escape from local optima and to avoid premature convergence.

Due to advances in roaches' studies, a lot of characteristics (listed further in this paper) regarding these animals were disclosed. Understanding that roaches' social behaviour was even more complete than the birds one, in 2008, Havens *et al* developed the first cockroaches' based algorithm, Roach Infestation Optimization (RIO) [3]. This work was the inspiration to adapt even more roaches' behaviour to these kind of algorithms.

II. PSO Description

Kennedy and Eberhart [3] defined a world whose dimensions were the variables of the chosen objective function, based on several studies mentioned above. So, if our objective function, Z, has D variables, there's a swarm (in a D-dimensional world) where each i^{th} particle exists on a certain position $X_i=[x_{i1}\ x_{i2}\ \dots\ x_{iD}]^T$ and moves by a certain velocity $V_i=[v_{i1}\ v_{i2}\ \dots\ v_{iD}]^T$ (limited to V_{max}), during t iterations, trying to find the best possible position. The best location is defined proportionally to the fitness function. It means that if one wants to minimize Z function, the best position will be defined by the smaller value that a particle's position achieves.

In this optimization approach, particle's velocity on instant k, V^k, depends on the previous velocity V^{k-1}, on the particle's best position ever achieved, *Pbest* (individual component), and on the position of the best positioned particle in all swarm *Gbest* (social component). That way, it is possible to calculate the next position using the following equations:

$$v_i^{k+1} = wv_i^k + C_1 r_1 (Pbest_i^k - x_i^k) + C_2 r_2 \left(Gbest_i^k - x_i^k\right) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2)$$

On these equations, r_1 and r_2 are D-dimensional vectors with random values between 0 and 1. These random vectors help to keep diversity on swarm. C_1 and C_2 are cognitive and social factor, respectively. These are the parameters that define the influence of cognitive and social components on particle's velocity. Usually they are set, as suggested by Kennedy [4], in a

way that C_1 plus C_2 equals 4. However, in 2001, Carlisle and Dozier [5], developed their research on cognitive and social parameters influence and found out that, for some functions, a sum lower than 4 was the best choice (like Schaffer F6). They also found out that, the values usually used on common sense of 2.05 to both C_1 and C_2 weren't the bests ones. They came to the conclusion that the best choice should be a ratio of 2.8 of cognitive parameter to 1.3 of social one. w is inertia weight, defined by Eberhart and Shi, in 1998 [6]. It receives a value between 0 and 1 and it defines the importance of particle's previous velocity. If it is a lower value there is a preference in local search once the previous velocity it is not valorised and the final velocity will be smaller. If it is a higher value, there is a preference in global search, because the previous value is highly taken into account.

Later, in order to keep all the advantages from a global search in the beginning and a local search in the end a PSO with linear decreasing inertia weight was introduced [7].

$$w(k) = w_{max} - \left(\frac{w_{max} - w_{min}}{k_{max}}\right) k \tag{3}$$

In equation 3, k is iteration number and w_{max} and w_{min} are, respectively, maximum and minimum possible values to inertia weight. There are three types of inertia weight: constant and random; time varying (like the one represented on equation 3); and adaptive ones, where inertia weight is adjusted according to some feedback parameters.

III. RIO Description

Roach Infestation Optimization (RIO) [3] stands for the first algorithm based on cockroaches' social life. Havens *et al*, understanding that cockroaches enjoy family company and socialize between them, changed the social component in PSO's velocity update equation.

Instead of learning only from best-positioned particle, in RIO, cockroaches learn from neighbours if there's socialization. Therefore, when a cockroach can find friends in its detection radius d_r (equation 4), there's the probability (table 1) that those roaches socialize.

$$d_r = \frac{\sum_{i=1}^{N-1}\left(\sum_{j=i}^{N}\|x_i - x_j\|\right)}{\sum_{i=1}^{N-1}\left(\sum_{j=i}^{N} 1\right)} \tag{4}$$

That socialization is carried out by an exchange of knowledge according to equation 5:

$$bk_1 = bk_2 = arg\ \min\{Z(bk_j)\}, j = \{1,2\} \tag{5}$$

Where bk is best known position (for roach 1 and roach 2) and Z is the objective function. So, every single time a roach meets another one, and they socialize, both roaches will remember the same best position ever achieved by a roach.

Finally, velocity equation on RIO algorithm will be as equation 6 displays:

$$v_i^{k+1} = wv_i^k + C_1 r_1 (bv_i^k - x_i^k) + C_2 r_2 \left(bk_i^k - x_i^k\right) \tag{6}$$

Where w is inertia weight, C_1 and C_2 are cognitive and social component, respectively, r_1 and r_2 are D-dimensional vectors with random values between 0 and 1 and bv is the best visited position (same as past best (*Pbest*) position from PSO).

Table 1: Probability of stop when finding N friends [8].

Number of Neighbours	Probability of socialize
1	0.49
2	0.63
≥3	0.65

IV. GRIO Description

In spite of the evolution from PSO to RIO, one still can't avoid efficiently premature convergence and local optima. Understanding such drawback, in this work, a novel optimization algorithm developed to continuous problems is proposed, Genetic Roach Infestation Optimization (GRIO).

GRIO is more tied to cockroaches' behaviour than RIO and it's highly founded on the following characteristics:

1. Cockroaches always seek the darkest possible spot to hide [9].
2. Cockroaches socialize when meeting friends [10].
3. Cockroaches don't reproduce with family members [11].
4. All cockroaches have the same potential to reproduce (no alpha male for example) [12].
5. Cockroaches hide their eggs in small dark shelters [12].
6. When a spot isn't big enough to all the roaches, they split themselves equally into several spots [13].
7. Cockroaches' genetic code is very suitable for mutations, facilitating adaptation to several environments and survival [14].
8. Isolated cockroaches are less likely to explore and leave their shelters [15].
9. Cockroaches have many natural predators [16].
10. Cockroaches have memory [17].

In the presented approach, the darkness level at some location is proportional to fitness function Z at that location (characteristic one). If one wants to minimize function Z at location $x \in \mathbb{R}^D$, darkness increases as much as $Z(x)$ decreases. All cockroaches keep in memory (characteristic ten) the best location ever visited and the best location ever achieved by a friend. The best location ever achieved by a friend is updated when they socialize (characteristic two).

In the beginning, it's believed that there aren't dark spots big enough to all swarm. So, a population consists on two equal sub-swarms seeking a dark spot (characteristic six). Those swarms are two different families and behave independently from the other family. Using this independent multi-swarm approach one can escape from local optima and increase solution variability.

Sometimes, a cockroach is able to reproduce (characteristic four) and finds a spot good enough to leave eggs (characteristic five). When it happens, it chooses a partner on the other family to have kids (characteristic three). Childs rise with a combination of parent's knowledge. In order to promote a global search, it is possible that some random mutation occurs (characteristic seven). Population size is controlled by predators (characteristic nine). If one allows population to keep growing, then the algorithm execution time would be excessive.

A tool to control global and local search is Cockroaches' speed. Velocity is affected by the number of neighbours in the moment and the distance to the best solution.

An isolated cockroach velocity is smaller than a surrounded one (characteristic eight). Another velocity controller is the distance to the best location ever achieved. The closer a cockroach is to the best solution the smaller will be its velocity to promote a more local search in that area.

GRIO's search behaviour is equal to RIO's one. Before movement, it's necessary to verify if cockroach have friends inside its detection radius (defined by user). If there are friends in that detection radius, there is a probability (table 1) that these cockroaches will stop and socialize. Socialization is performed in algorithm by a comparison between roaches best-known position. So, when two roaches meet, both roaches best-known position is updated according to equation 5.

Then, once the best-known position is updated, roach's position is calculated using equations 6 and 2, respectively.

a. Adaptive Inertia Weight

An important parameter in all PSO-based algorithms is inertia weight. This parameter controls velocity. If it is a lower value there is a preference in local search once the previous velocity is not valorised and the final velocity will be smaller. If it is a higher value, there is a preference in global search, because the previous value is highly taken into account.

As inertial weight is a major parameter to proper convergence and it is difficult to choose a suitable fixed inertia weight, several authors have been developing new techniques to introduce a variable inertial weight that adapts while algorithm is running.

In this work, an adaptive linear decreasing inertia weight, adjusted to each particle/roach, is proposed. That way one can achieve better results once linear weight is adapting itself according to each situation. On figure 1, linear weight adaptive system is represented by a flowchart.

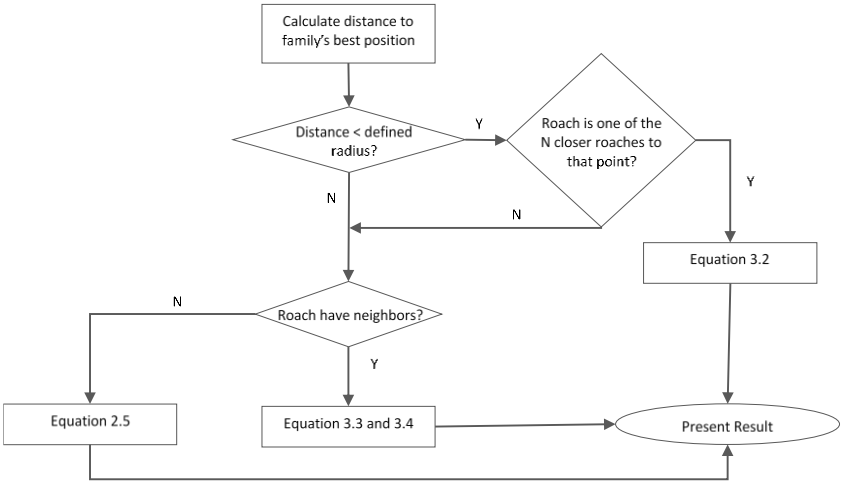

Figure 1: Adaptive behaviour's flowchart

It's important to promote a local search in final iterations and near optimum point. So, when a cockroach is about to be moved, his distance to the family's best position ever achieved is checked (equation 7).

$$d = \sqrt{\sum_{j=1}^{D} (bp(j) - x(j)^2)} \quad (7)$$

In that equation, D represents problem dimension, bp is the best position ever achieved and x is the roach position.

If that distance is smaller than the maximum distance (user defined), roach is inside the $\frac{1}{4}N$ closest roaches to family's best position and iterations are bigger than $\frac{3}{4}i_{max}$ (i_{max} is the highest iteration), then inertia weight is updated using equation 8, where N is the number of roaches in the swarm.

$$w = -\frac{0.45}{round\left(\frac{N}{4}\right)} distanc\, e^2 + 0.45 \tag{8}$$

When roaches don't meet criteria to use equation 6, inertia weight follows a linear decreasing method. So, as long as iterations are increasing, inertia weight is decreasing (equation 3).

As accompanied cockroaches are more active then those isolated, it was defined a parameter (hyper-activity, equation 9) that is summed to regular linear decreasing inertia weight multiplied by number of neighbours (equation 10).

$$ha = \left(-0.1\frac{i}{i_{max}} + 0.105\right) Nn \tag{9}$$

$$w = \left(w_{max} - i\frac{w_{max} - w_{min}}{i_{max}}\right) + hyper - activity \tag{10}$$

As we use those equations, where Nn is the number of neighbours, instead of common inertia weights, one can adapt inertia weight and obtain more suitable results. As one applies this adaptive inertia weight, a guideline equal to linear decreasing method is kept. However, sometimes it is bigger (when a cockroach has neighbours) or smaller (when a cockroach is near the best position ever achieved). In the final iterations there's more variations from the linear guideline because there is more probability of having roaches groups and roaches near best position.

b. Roaches Reproduction

GRIO have a cooperative component between the two different and independent swarms. While reproducing, there is a share of the knowledge from one family to another.

Reproduction is performed like a regular genetic algorithm [18]. First of all, one must be eligible to reproduce and that is possible when two conditions are satisfied:

- Roach is on fertile window (emulated with a user defined random number).
- Roach finds a secure spot to leave eggs (emulated when best known position isn't changed during five iterations).

If these conditions are fulfilled at the same time, roach is able to reproduce and chooses the best-positioned roach in the opposite family. Once two reproducing roaches are selected, it's necessary to choose the "genetic code" (vector with coordinates). This vector is generated creating one hundred different combinations of best-visited position vector (bv) and best-known position vector (bk). That way, one achieves a vector with information from two different memories of each roach. For each roach involved in the reproduction process, are chosen the best combination vector obtained and then a new crossover process is done and a kid is generated (the kid vector will represent the baby roach's actual position, best known position and best visited position). One repeats this procedure until all kids (number decided by user) are born. Once it's necessary to keep swarm size to avoid extra running time, after born, one kills N random extra roaches. In figure 2 a flowchart of this process is represented.

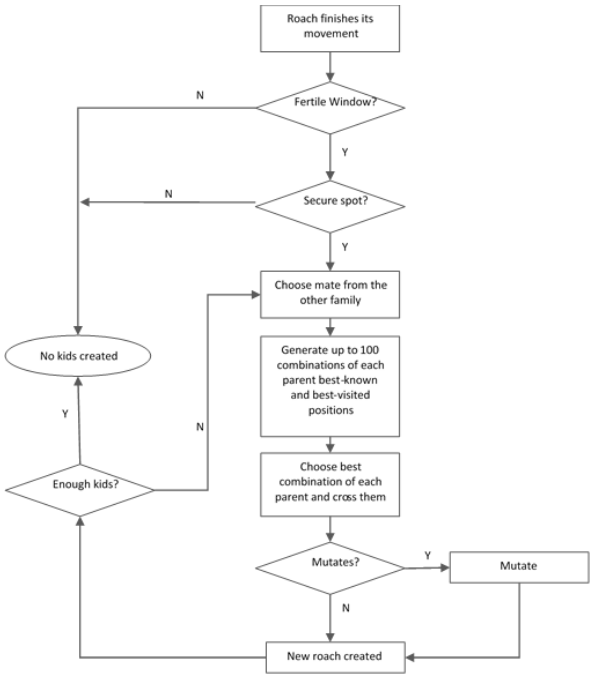

Figure 2: Reproduction behaviour flowchart

c. Roaches' Mutations

A major issue in PSO based algorithms is premature convergence. In order to avoid it is necessary to keep diversity in search space. Otherwise solutions will converge into a single spot (a good one or not) and there will be a huge unsearched and unknown space.

Understanding that, there were introduced two kinds of mutations: replacement mutation and frameshift mutation. Every time a baby roach rises, there's the opportunity of being a healthy one, or a mutant one (however only one kind of mutation will occur).

- Replacement mutations: Some random values from kid's vector are replaced by new values (figure 3, a)).
- Frameshift mutations: A random sequence of values is allocated in a new place (figure 3, b)).

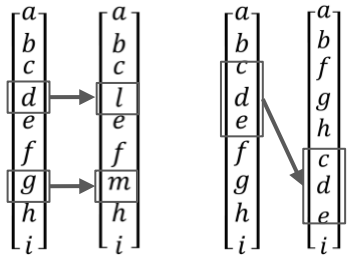

Figure 3: Mutations. a) Replacement mutation. b) Frameshift mutation.

V. Results and Discussion

In this section, GRIO algorithm's performance is tested in comparison to four other algorithms. In order to do it, there were selected some functions with known solutions.

a. Benchmarks

Benchmark functions are very useful once they allow one to test new algorithms using functions with pre-known solutions and characteristics. So, in order for a good test to be made to GRIO algorithm, one must choose carefully all the test functions. There are some characteristics one must be careful about:

- Dimensionality: Measures the number of dimensions a function affords. Functions with more dimensions/parameters are more difficult to solve once the search space increases exponentially.
- Modality: Measures the number of peaks. As a function has more peaks, the probability of being trapped in local optima increases.
- Separability: Measures the difficulty of optimize a function. If a dimension depends on the other the function is independent from the other one and the problem is easier to solve than one with dimensions related to each other.
- Valleys: This phenomenon occurs when a narrow area is surrounded by steep descents. If an algorithm is trapped into such region it can be difficult to leave and slowed.

In this work, twenty three benchmark functions were tested and results analysed. All those functions were tested along with other four algorithms (one of them from another work as reference that can be consulted in references chapter). In master thesis *Genetic Roach Infestation Optimization* all results can be consulted and analysed, however, to keep size in this paper, only the hardest ones, according to all characteristics presented previously will be presented:

Table 2: Benchmark functions specifications.

Function	Dimension	Initialization Range	Minimum	Reference
Ackley	30	$[-32,32]$	0	[19]
Griewank	30	$[-600,600]$	0	[20]
Langerman	10	$[0,10]$	-1.4	[21]
Michalewicz	10	$[-\pi,\pi]$	-9.66015	[3]
Paviani	10	$[2.001,9.999]$	-45.778	[21]
Rastrigin	30	$[-5.12,5.12]$	0	[20]
Rosenbrock	30	$[-2.048,2-048]$	0	[20]

b. Experimental Tests

Three PSO variants and reference algorithm were chosen to compare to GRIO algorithm. Therefore, used algorithms and conditions are listed:

- Genetic Roach Infestation Optimization (GRIO)
  - Number of evaluations: 20
  - Maximum Iteration: 1000
  - Number of Particles in each swarm: 10
  - C_1 and C_2: 1.43
  - Maximum inertia weight (when using linear decreasing): 0.9
  - Minimum inertia weight (when using linear decreasing): 0.4
  - $Prob\_rep$: 20%
  - $Prob\_mut\_rep$: 25%
  - $Prob\_mut\_fram$: 25%
- PSO with random inertia weight (RN-PSO)
  - Number of evaluations: 20
  - Maximum Iteration: 1000
  - Number of Particles: 20
  - C_1 and C_2: 1.43
  - Maximum inertia weight: 1
  - Minimum inertia weight: 0
- Reference one (with conditions optimal for that algorithm according to authors)
- PSO with linear decreasing inertia weight (LW-PSO)
  - Number of evaluations: 20
  - Maximum Iteration: 1000
  - Number of Particles: 20
  - C_1 and C_2: 1.43
  - Maximum inertia weight: 0.9
  - Minimum inertia weight: 0.4
- Roach Infestation Optimization (RIO)
  - Number of evaluations: 20
  - Maximum Iteration: 1000
  - Number of Particles: 20
  - C_1 and C_2: 1.43
  - Inertia weight: 0.7

As one can see, with the exception of reference, number of evaluations, iterations and particles are exactly the same for a fair comparison. Actually, it was attempted to keep all variables on the same values to have a better comparison. Final results for all tests can be found on table 3. In this table, best results achieved are bolded.

Table 3: Complete Results (bolded best results).

Function		LW-PSO	RW-PSO	RIO	GRIO	Reference
Ackley	Best	1.244e+02	1.197e+01	4.424	**2.841e-09**	1.146e-04
	Worst	3.124e+02	1.991e+01	1.970e+01	**4.589e-07**	3.901e-04
	Mean	2.227e+02	1.709e+01	1.020e+01	**1.482e-07**	2.005e-04
	Std.	4.753e+01	2.695	4.548	**1.422e-07**	3.772e-05
Griewank	Best	9.476e-04	1.048e-01	1.351e-12	**0**	1.480e-02
	Worst	4.000e-02	1.014e+00	1.970e-02	**2.929e-13**	7.870e-02
	Mean	1.600e-02	5.668e-01	7.500e-03	**5.256e-14**	4.91e-02
	Std.	1.120e-02	2.702e-01	8.400e-03	**8.535e-14**	1.78e-02
Langerman	Best	**-1.4**	**-1.4**	**-1.4**	**-1.4**	-2.128e-02
	Worst	-2.223e-13	-1.673e-13	-1.912e-13	**-1.4**	Na
	Mean	-1.015	-9.351e-01	-8.428e-01	**-1.4**	Na
	Std.	5.574e-01	6.503e-01	0.671	**3.482e-09**	Na
Michalewicz	Best	-9.362	-2.686	-9.2799	**-9.66015**	Na
	Worst	-5.640	-1.111	-5.8654	**-9.6552**	Na
	Mean	-7.300	-1.890	-7.9299	**-9.6579**	-5.8
	Std.	0.8607	0.482	0.9422	**0.0024**	Na
Paviani	Best	**-45.778**	**-45.778**	**-45.778**	**-45.778**	-14.053
	Worst	**-45.778**	**-45.778**	**-45.778**	**-45.778**	Na
	Mean	**-45.778**	**-45.778**	**-45.778**	**-45.778**	Na
	Std.	**1.124e-14**	2.824e-14	1.310e-14	5.427e-10	Na
Rastrigin	Best	9.155e+01	3.097e+02	1.244e+02	**0**	8.512e-12
	Worst	2.817e+02	2.843e+03	3.124e+02	**1.810e-10**	1.990
	Mean	1.757e+02	7.733e+02	2.227e+02	**4.296e-11**	6.985e-01
	Std.	5.051e+01	5.325e+02	4.753e+01	**4.969e-11**	7.284e-01
Rosenbrock	Best	1.227e+02	4.517e+03	4.629e+00	**2.684e-05**	6.687e-05
	Worst	5.476e+02	8.719e+05	7.724e+01	**7.904e-02**	2.218e+01
	Mean	2.710e+02	2.261e+05	2.899e+01	**1.290e-02**	2.835
	Std.	1.079e+02	2.1432+05	1.976e+01	**2.280e-02**	3.583

Time of processing and calculation is directly related to the number of times the objective function needs to be evaluated. Ideally an algorithm must be able to find the optimal point in the less time possible. For that reason, sometimes, is not

profitable to have an algorithm that solves the problem in a time that doesn't fit our needs [21]. The Number of function evaluations is presented on table 4.

It is also important to note that the number of times an algorithm needs to evaluate the function is dependent on the way the algorithm was programed. If the algorithm doesn't achieve the global optima, then the number of functions evaluations is the correspondent to the 1000 iterations done (maximum). If it achieves, then the result presented in table 4 is the number of evaluations that the algorithm took until got there. The algorithms that achieved the best result using the less function evaluations are bolded.

Table 4: Function evaluations for each tested algorithm. Bolded the ones that achieved the best result.

Function	LW-PSO	RW-PSO	RIO	GRIO
Ackley	840e+03	840e+03	880e+03	**1840e+03**
Griewank	840e+03	840e+03	880e+03	**1722240**
Langerman	276360	**58800**	80960	427e+03
Michalewicz	840e+03	840e+03	880e+03	**732908**
Paviani	213360	**66360**	73040	107800
Rastrigin	840e+03	840e+03	880e+03	**1766400**
Rosenbrock	840e+03	840e+03	880e+03	**1840e+03**

c. Results Discussion

In order to better discuss and analyse results for all functions, some graphs are presented. For a better distribution analyses, aiming to determine if an algorithm is achieving results consistently or not, box plots are presented on figure 4. For convergence analysis, a path of best value is traced so that one can check the ability of all algorithms to escape from local optima (figure 5). Some convergence graphics have logarithmic or exponential scale for a better visual experience.
Important to note that the following graphics do not include reference's results (as tables presented previously) once all the test results weren't available.

Analysing all the tables above, one notes that GRIO outperforms all other algorithms with the exception of function Paviani. Actually, GRIO manages to get optimal value but in a less consistent way than other algorithms as can be seen in figure 4. Still on figure 4, as expected, GRIO algorithm results have mean value below all the other algorithms. Bigger boxes support that algorithm fell into local optima more often. That could be concluded from standard deviation values represented on table 2. In Ackley's function, despite GRIO's best result is an outlier one, GRIO's worst result is much better than all other algorithms best value.

Regarding convergence, interpretation from figure 5 is quite easy. If we check functions where GRIO wasn't the only one to achieve the optimum point, we can see that all algorithms got a fast convergence to that point (even before the iteration 100). However, in functions where other algorithms failed, due to lack of capacity to avoid local minima points, GRIO could overcome those points and keep improving its score. If we look to GRIO's line on Ackley and Rosenbrock functions (the only ones that GRIO couldn't find the minimum in 1000 iterations) it is easy to spot that with more iterations the line should keep decreasing, once the algorithm wasn't stuck in local points like other tested algorithms, and global minimum should, eventually, be found.

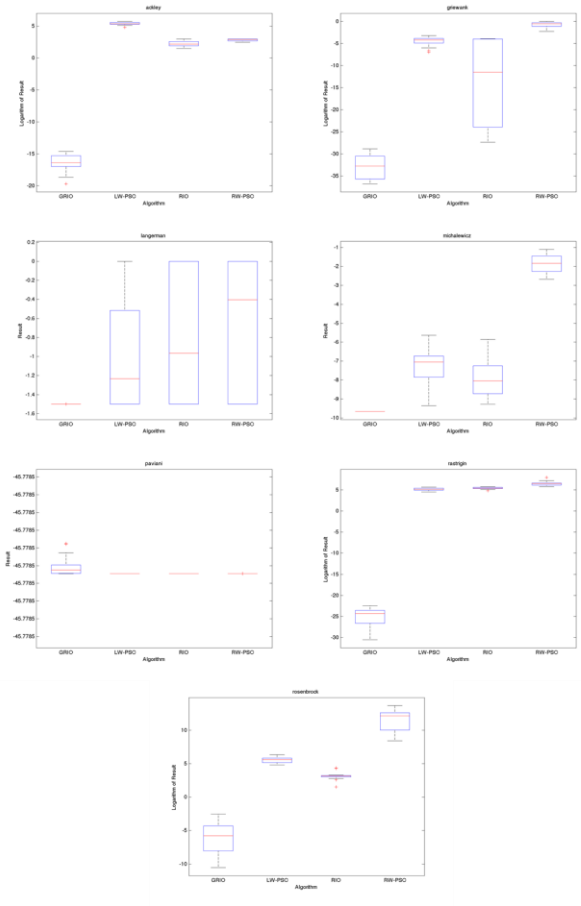

Figure 4: Boxplots for hard functions.

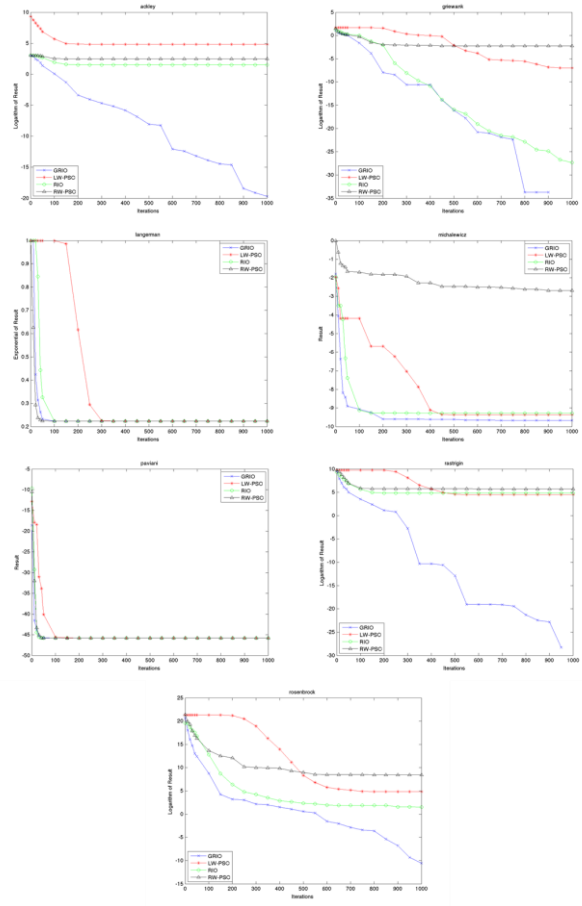

Figure 5: Convergence graphs for hard functions

VI. Conclusions

This work aimed to develop an algorithm capable to solve the most difficult continuous functions. Seeing an opportunity to improve RIO algorithm taking into account cockroaches social life, GRIO appeared.

GRIO is an algorithm capable of escape from local optima and keep diversity in the search. This characteristics led GRIO to outperform other algorithms while optimizing very hard benchmark functions as can easily be seen in results and discussion chapter (on convergence analysis we can see that on functions were other algorithms couldn't find optima soon, only GRIO could keep improving its result).

On the Master Thesis *Genetic Roach Infestation Optimization* is proved that GRIO doesn't outperform other algorithms in functions with few dimensions. In fact, is proved that GRIO's performance is as better as complexity increases, because most of anti-premature convergence strategies from GRIO are based in changing information between dimensions. Meaning that if there are few dimensions, few information is changed as well.

On the hardest functions, represented in this paper, due to its escape from local optima techniques and convergence velocity regulation among other details, GRIO was able to achieve results not seen with any other algorithm.

Concluding, one may say that GRIO allows optimization of hard, high-dimensional and continuous functions.

Finally, now that one knows about this algorithm's potential, some steps should be taken in order to transport GRIO algorithm to real applications.

A discrete version of the algorithm must be made once that all real-life problems are modelled as discrete. One of those problems could be related to logistics' industries in order to optimize delivery routes taking into account the kind of vehicle, transportation, deadlines, fuel's consumption, etc.

As a final point, analysing all the work done and previous conclusions, one may conclude that this work was concluded with success once that the main objective was achieved: produce an algorithm capable of solving the most difficult continuous functions. Also it was amazing to produce an algorithm with results never seen before for some functions, while learning how to create, test and compare a new approach. Competences that are highly valorised in a young engineer in the industrial world.

VII. Acknowledgements

This work is the final of an academic path. However, for me, it is even more special once it only come true due to all the support my family always gave me while I was already working. I will ever forget it and I will be thankful all my entire life.

Also to my thesis advisor for all the patience and wise advises despite my lack of availability. It was really appreciated.

Finally to every single student and teacher that crossed with me and, more or less, always contributed with a piece of who I am and where I am.

Thank you all.

VIII. References

[1] Craig, W.R., *Flocks, herds and schools: A distributed behavioral model.* SIGGRAPH Comput. Graph. %@ 0097-8930, 1987. **21**(4): p. 25-34.

[2] Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Neural Networks, 1995. Proceedings., IEEE International Conference on*. 1995.

[3] Havens, T.C., et al. *Roach Infestation Optimization*. in *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*. 2008.

[4] James, K., *The Behavior of Particles*, in *Proceedings of the 7th International Conference on Evolutionary Programming VII %@ 3-540-64891-7*. 1998, Springer-Verlag. p. 581-589.

[5] Carlisle, A. and G. Dozier, *An Off-The-Shelf PSO*. 2001.

[6] Yuhui, S. and R. Eberhart. *A modified particle swarm optimizer*. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998.

[7] Shi, Y. and R. Eberhart, *Parameter selection in particle swarm optimization*, in *Evolutionary Programming VII*, V.W. Porto, et al., Editors. 1998, Springer Berlin Heidelberg. p. 591-600.

[8] Jeanson, R., et al., *Self-organized aggregation in cockroaches.* Animal Behaviour, 2005. **69**(1): p. 169-180.

[9] Halloy, J., et al., *Social Integration of Robots into Groups of Cockroaches to Control Self-Organized Choices.* Science, 2007. **318**(5853): p. 1155-1158.

[10] Gautier, J.Y., P. Deleporte, and C. Rivault, *Relationships between ecology and social behavior in cockroaches.* The ecology of social behavior, 1988: p. 335-351.

[11] Lihoreau, M., C. Zimmer, and C. Rivault, *Kin recognition and incest avoidance in a group-living insect.* Behavioral Ecology, 2007. **18**(5): p. 880-887.

[12] Gillott, C., *Entomology*. 2005: Springer Netherlands.

[13] Amé, J.-M., et al., *Collegial decision making based on social amplification leads to optimal group formation.* Proceedings of the National Academy of Sciences, 2006. **103**(15): p. 5835-5840.

[14] Wada-Katsumata, A., J. Silverman, and C. Schal, *Changes in Taste Neurons Support the Emergence of an Adaptive Behavior in Cockroaches.* Science, 2013. **340**(6135): p. 972-975.

[15] Lihoreau, M., L. Brepson, and C. Rivault, *The weight of the clan: even in insects, social isolation can induce a behavioural syndrome.* Behavioural Processes, 2009. **82**(1): p. 81-84.

[16] Roth, L.M. and E.R. Willis, *The Biotic Associations of Cockroaches*. 1960: Smithsonian Institution.

[17] Sakura, M. and M. Mizunami, *Olfactory Learning and Memory in the Cockroach Periplaneta americana.* Zoological Science, 2001. **18**(1): p. 21-28.

[18] Golberg, D.E. and M.P. Samtani. *Engineering Optimization via Genetic Algortihm*. in *9th Conf. Eletronic Computation*. 1986.

[19] Jie, J., et al., Knowledge-based cooperative particle swarm optimization. Applied Mathematics and Computation, 2008. 205(2): p. 861-873.

[20] Niu, B., et al., MCPSO: A multi-swarm cooperative particle swarm optimizer. Applied Mathematics and Computation, 2007. 185(2): p. 1050-1062.

[21] Custódio, A.L. and J.F.A. Madeira, GLODS: Global and Local Optimization using Direct Search. Journal of Global Optimization, 2015. 62(1): p. 1-28.