Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Rotation-Based Multi-Particle Collision Algorithm with Hooke Jeeves

Reynier Hernández Torres[1]
Haroldo F. de Campos Velho[2]
Instituto Nacional de Pesquisas Espaciais (INPE) São José dos Campos, SP

Abstract. A new variant of the hybrid metaheuristic MPCA-HJ (Multi-Particle Collision Algorithm with Hooke-Jeeves method) is presented. Multi-Particle Collision Algorithm is a metaheuristic algorithm that performing a traveling on the search space. The addition of the Rotation-Based Learning mechanism to the exploration search enhances the possibility to cover a larger area in the search space. The Hooke-Jeeves direct search method exploites the best solution found by the MPCA, allowing to achieve better solutions. The performance of all implementation are evaluated over twenty-two well known benchmark functions.

Keywords. Hybrid metaheuristic, rotation-based learning, opposition-based learning, multi-particle collision algorithm

1 Introduction

Optimization refers to a large area of the Applied Mathematics that studies the theory and methods for finding optimal values from all the possible values to minimize or maximize some objective function, given constraints and defined domain. Optimization is involved in many real problems of all fields of study.

Optimization techniques can be divided in two large areas: *deterministic* and *stochastic* optimization (uses random processes). Metaheuristic algorithms belong to a class of stochastic optimization. Metaheuristics have mechanisms for facilitating the exploration of the search space, while an exploitation of the best solutions is made. Such schemes have strategies to scape of a local optima, following the exploration of the search space.

A hybridization with the Multi-Particle Collision Algorithm, where the Rotation-Based Learning (RBL) mechanism is applied, coupled with the Hooke-Jeeves (MPCA-HJ), will be described in the next section.

2 Algorithms

2.1 Multi-Particle Collision Algorithm (MPCA)

MPCA is a populational optimization algorithm inspired in the physics of nuclear particle collision reactions [1,2]. In a nuclear reactor, an incident particle could be scattered

[1]reynier.torres@inpe.br
[2]haroldo@lac.inpe.br

by a target nucleus, phenomenum called scattering, or the particle could be absorbed by the target nucleus, kwown as absorption.

MPCA is an algorithm consisting in a set of particles (candidate solution) travelling inside a nuclear reactor (search space). This particles are perturbed, and the resulting particles could be absorbed (if its fitness is improved, they will substitute the *old* particles), or scattered (if its fitness is worst, a new particle is created in another place of the space search). Also, the particles behave cooperatively, i.e., after some iteration, the best particle overall is over-copied for all particles in the set, through a blackboard strategy, and a new interation is started.

The MPCA starts with a initial set of N_p particles randomly created over the search space. Then, the particles traveling process is started, involving three main functions: Perturbation, Exploitation, and Scattering.

The PERTURBATION function performs a random variation of a particle within a defined range. The d-th variable of each perturbated particle $newP$ is calculated by the equation:

$$new - P_d = P_d + ((UB_d - P_d) \cdot R) - ((P_d - LB_d) \cdot (1 - R)), \tag{1}$$

where P is the particle to be perturbed, UB and LB are the upper and the lower bounds in the search space, respectively, and R is a random number uniformly generated between 0 and 1.

If the new particle $newP$ is better that the current particle P, then the EXPLOITATION function is activated. Else, if the new particle $newP$ is worse than the current particle P, the SCATTERING function is activated.

The EXPLOITATION function intensify the search on the solution found, by means of $NFE_{exploit}$ small perturbations. This intensification process finds the dth variable of each new exploited particles ($exploitedP$) using the following equation:

$$exploitedP_d = P_d + ((u_d - P_d) \cdot R) - ((P_d - l_d) \cdot (1 - R)), \tag{2}$$

where $u_d = P_d \cdot rand(1, \mathrm{SL})$ and $l_d = P_d \cdot rand(\mathrm{IL}, 1)$ are the upper and lower limits computed from the current particle P, using the superior (SL) and inferior (IL) limits for the random numbers generated. Each new exploited particle $exploitedP$ is compared with the original particle P, and will substitute it if is better.

The SCATTERING function is based on a Metropolis scheme: there are two options selected with a defined probability: (i) the current particle P is replaced by a new random solution P, or (ii) a series of small perturbations are performed on it (as in the Exploitation function).

This principal loop will be stopped after a maximum number of function evaluations (NFE_{mpca}) defined by user.

The blackboard updating is applied at three times: (i) after the initial set of particles is created, (ii) ending of each iteration –if a number of function evaluations (NFE_{bb}) was reached after the last blackboard updating, and (iii) just after the principal loop is stopped.

The current version MPCA was implemented in C++.

2.2 Opposition-Based Learning (OBL)

The OBL concept was introduced in 2005 by Tizhoosh [3]. The idea of OBL is to evaluate the candidate solution and its opposite solution, for getting the better solution among them. Later, other variants, such as Quase-Opposition Based Learning, Quase-Reflective Based Learning, Center-Based Sampling Learning, Rotation-Based Learning have been appeared, giving more success in the exploration/exploitation of the search space and improving the convergence [4].

OBL and their extensions have been applied to improve the performance of various computational intelligence methods, such as artificial neural networks, fuzzy logic, metaheuristic algorithms, and miscellaneous applications [5].

Mathematically, the opposite number z_o of a real number $z \in [a, b]$ is defined by:

$$z_o = a + b - z \ . \tag{3}$$

The opposite point $Z_o = (z_{o_1}, z_{o_2}, \cdots, z_{o_D})$ of a point $Z = (z_1, z_2, \cdots, z_D)$, with D dimensions, is completely defined by its coordinates as show in equation (4).

$$z_{o_d} = a_d + b_d - z_d \tag{4}$$

where $z_d \in \mathbb{R}$, with $a_d \leq z_d \leq b_d \ \forall d \in \{1, 2, \cdots, D\}$.

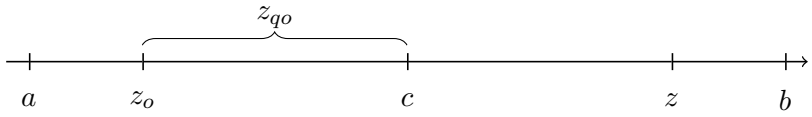

Figura 1: Opposition-Based and Quasi-Opposition Based Learning.

Another mechanism, the Quasi-opposition learning (QOL) reflects a point to a random point between the center of the domain and the opposite point.

$$Z_{qo}(z_{qo_1}, z_{qo_2}, \cdots, z_{qo_D}) \mid z_{qod} = \text{rand}(c_d, z_{o_d}) \ . \tag{5}$$

The concept of opposition number could be placed on a two-dimensional space as shown in Figure 2: a circle of center c and radius $(b - a)/2$ can be drawn (dotted in Figure). A line from z is drawn perpendicular to the x-axis and intersecting with the circle on poinT l. Then, the point l is rotated π radians counterclockwise around the circle (that is called reflection angle (β)), and a point m will be reached. The projection of m on x-axis is the opposite number (z_o).

Also, if the deflection angles β are carefully selected so that the rotation points are localized between the opposition point Z_o and the center C, the QOL is represented.

2.3 Rotated-Based Learning (RBL)

The RBL concept is an extension of the OBL and QOL mechanisms [4].

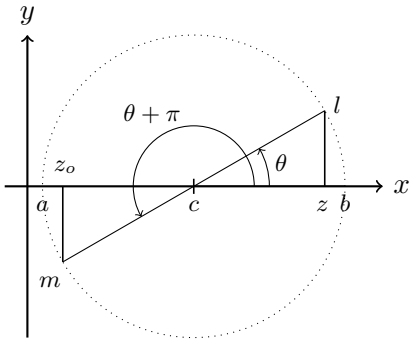

Figura 2: Geometric interpretation of the opposite number in 2D.

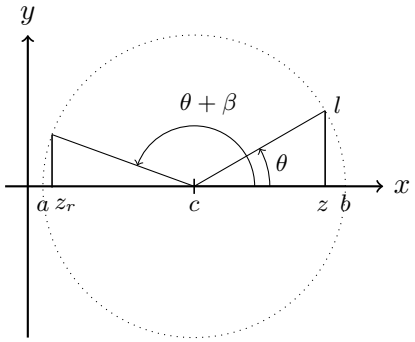

Figura 3: Geometric interpretation of the rotation number in 2D.

Let $Z = (z_1, z_2, \cdots, x_D)$ be a vector with D variables, $A = (a_1, a_2, \cdots, a_D)$ is the lower boundary, and $B = (b_1, b_2, \cdots, b_D)$ is the upper boundary of the search space, respectively. The center point of the search space in each dimension is denoted by $C = (c_1, c_2, \cdots, c_D)$ and $c_i = (a_i + b_i)/2$.

The rotation point $Z_r = (z_{r1}, z_{r2}, \cdots, z_{rD})$ can be calculated by

$$z_{ri} = c_i + u_i \times \cos\beta - v_i \times \sin\beta\,, \tag{6}$$

where u_i is the quantity from the z_i to the center ($u_i = z_i - c_i$), and v_i is the lenght of the point to the corresponding intersection point l_i on the circle ($v_i = \sqrt{(z_i - a_i)\,(b_i - z_i)}$).

The deflection angle β is a random value with Gaussian distribution of one mean and δ as its standard devaition, and it is defined in the equation (7).

$$\beta = \beta_0 \cdot \mathcal{N}(1, \delta)\ . \tag{7}$$

2.4 Hooke-Jeeves Direct Search Method

The Hooke-Jeeves direct search method [6] consists of the repeatedly application of exploratory moves around a base point which, if successful, is followed by pattern moves. In a D-dimensional problem, a candidate solution is denoted as a vector s of length D.

The exploratory movement consists adding (and substracting) the column v_d of the search directions matrix V, scaled by a step size h, to the solution s. This process is repeated for all the dimensions of the problem. The new solution s^n will be accepted if is better than the previous s.

If the exploratory movement was successful, it will return an improved solution s^n. Later, a pattern move s^{n*} is done adding a search direction to s^n: $s^{n*} = s^n + (s^n - s^c)$. The solution s^c will be replaced by s^{n*} if the result of the pattern movement is better; if not s^n will become the new solution s^c. If the exploratory movement was not successful, the step size h is reduced in ρ times. If the step size reach a minimum value h_{min}, the algorithm will stop. Another stopping criterium is to reach a maximum number of function evaluations (NFE_{hj}) defined.

3 Numerical Experimentation

3.1 Experimental configuration

Experiments were made in a personal computer with 8x Intel® Core™ i7-4790 CPU @ 3.60GHz, with 8 GB of memory, operating with Ubuntu 14.04.3 LTS.

The number of experiments was set in 25. MPCA parameters were set in $N_p = 40$, $NFE_{\text{bb}} = 10000$, $NFE_{exploit} = 500$, $IL = 0.7$, and SL: 1.2. HJ parameters were set in $\rho = 0.8$, and $h_{\min} = 1 \times 10^{-10}$. RBL parameters were set in $\beta_0 = 60$, and $\delta = 0.25$.

3.2 Benchmark Functions

Twenty-two benchmark functions with $D = 30$ and different characteristics were implemented to evaluate the performance of the algorithms. Those functions are commonly used in the literature, and are selected based on different properties of separability and modality, representing a varied range of difficulty. In this selection, we have six Unimodal Separable (Sphere (f_1), Powell sum (f_2), Sum squares (f_3), Quartic with noise (f_4), Schwefel 2.21 (f_5), Step (f_6)), and four Non-separable (Schwefel 1.2 (f_7), Dixon & Price (f_8), Schwefel 2.22 (f_9), Rosenbrock (f_{10})); five Multi-modal Separable (Schwefel 2.26 (f_{11}), Michalewicz (f_{12}), Rastrigin (f_{13}), Alpine (f_{14}), Levy (f_{15})), and seven Non-separable (Exponential (f_{16}), Rana (f_{17}), Griewank (f_{18}), Ackley (f_{19}), Zakharov (f_{20}), Salomon (f_{21}), Egg holder (f_{22})). More information on these benchmark function can be found on the vast literature [7, 8].

All the algorithms are terminated when the number of function evaluation exceeds the predetermined maximum number of $10^5 \times D$.

3.3 Results of the experiments

The table 1 shows the statistics results for 25 trials for all the variants with and without RBL and HJ (MPCA, RMPCA, MPCAHJ, RMPCAHJ) over 22 benchmark functions, each one with different initial populations and random seeds. Functions f_4, f_{11}, f_{17} and f_{22} could not be solved by none of the algorithms. Functions $f_5, f_6 f_8 - f_{10}, f_{15}$ and f_{19} were solved only by the variants with HJ. It is important to stand out that just the RMPCA-HJ could solve the f_{13} and f_{14} problems.

4 Final remarks

In this study, the hybrid mataheuristic RMPCA-HJ was proposed as an alternative for solving optimization problems. It takes advantages of the exploration mechanism of the MPCA, with the complement of the Rotation-Based Learning, and the power of intensification of the HJ method.

The performance of the algorithm was compared with the canonical MPCA, the hybrid MPCA-HJ, and the RMPCA. The latter could solve problems that the others algorithms fail, and obtained better results over some other problems.

Acknowledgements

The authors acknowledge the financial support from by the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), a Brazilian agency for research support.

Referências

[1] W. F. Sacco and C. R. E. Oliveira. A new stochastic optimization algorithm based on a particle collision metaheuristic. *Proceedings of 6th WCSMO*, 2005.

[2] E. F. P Luz, J. C. Becceneri, and H. F. Campos Velho. A new multi-particle collision algorithm for optimization in a high performance environment. *Journal of Computational Interdisciplinary Sciences*, 1(1):3–10, 2008.

[3] H. R. Tizhoosh. Opposition-Based Learning: A New Scheme for Machine Intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 695–701. IEEE, 2005.

[4] Huichao Liu, Zhijian Wu, Huanzhe Li, Hui Wang, Shahryar Rahnamayan, and Changshou Deng. *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings*, chapter Rotation-Based Learning: A Novel Extension of Opposition-Based Learning, pages 511–522. Springer International Publishing, Cham, 2014.

[5] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao. A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, 29:1–12, 2014.

[6] R. Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[7] Marcin Molga and Czesław Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 2005.

[8] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

Tabela 1: Statistical for the number of function evaluations needed to reach the VTR. Best and mean values are shown. Values between parenthesis represent the success rate. Asteriscs denote that the functions could not reach the VTR before the maximum number of function evaluation set (NFE). Bold values respresent the best values for each function

Function	Statistic	MPCA	RMPCA	MPCAHJ	RMPCAHJ
f_1	best	6036	5562	**5110**	6150
	mean	8197 (1.00)	8781 (1.00)	7716 (1.00)	8664 (1.00)
f_2	best	2915	2229	2595	**1826**
	mean	3882 (1.00)	3928 (1.00)	3751 (1.00)	3721 (1.00)
f_3	best	6205	5798	**5383**	**5319**
	mean	**8612** (1.00)	10622 (1.00)	8981 (1.00)	9977 (1.00)
f_4	best	*	*	*	*
	mean	* (0.00)	* (0.00)	* (0.00)	* (0.00)
f_5	best	*	*	**16147**	21277
	mean	* (0.00)	* (0.00)	209124 (1.00)	**150134** (1.00)
f_6	best	*	*	**12596**	24217
	mean	* (0.00)	* (0.00)	**126939** (1.00)	182692 (1.00)
f_7	best	49985	*	**10086**	14290
	mean	862397 (0.24)	* (0.00)	**95996** (1.00)	114785 (1.00)
f_8	best	*	*	**132843**	175272
	mean	* (0.00)	* (0.00)	**788699** (0.44)	920488 (0.20)
f_9	best	*	*	**16205**	17378
	mean	* (0.00)	* (0.00)	158860 (1.00)	**136105** (1.00)
f_{10}	best	*	*	**23382**	52948
	mean	* (0.00)	* (0.00)	**190145** (1.00)	291331 (1.00)
f_{11}	best	*	*	*	*
	mean	* (0.00)	* (0.00)	* (0.00)	* (0.00)
f_{12}	best	**5978**	7686	6226	6802
	mean	**8988** (1.00)	11656 (1.00)	9176 (1.00)	11828 (1.00)
f_{13}	best	*	*	*	**20275**
	mean	* (0.00)	* (0.00)	* (0.00)	**922024** (0.08)
f_{14}	best	*	*	*	**102039**
	mean	* (0.00)	* (0.00)	* (0.00)	**877135** (0.16)
f_{15}	best	*	*	**11148**	17392
	mean	* (0.00)	* (0.00)	124478 (1.00)	**100087** (1.00)
f_{16}	best	4203	**3948**	4224	4463
	mean	6537 (1.00)	7576 (1.00)	**6328** (1.00)	7573 (1.00)
f_{17}	best	*	*	*	*
	mean	* (0.00)	* (0.00)	* (0.00)	* (0.00)
f_{18}	best	8976	11111	**7556**	10134
	mean	13172 (1.00)	20266 (1.00)	**12848** (1.00)	20800 (1.00)
f_{19}	best	*	*	**19115**	51458
	mean	* (0.00)	* (0.00)	**123442** (1.00)	170829 (1.00)
f_{20}	best	**8062**	15179	**8083**	12154
	mean	**14471** (1.00)	52799 (1.00)	**14106** (1.00)	60998 (1.00)
f_{21}	best	7744	4841	6191	**4042**
	mean	21707 (1.00)	8132 (1.00)	33921 (1.00)	**7991** (1.00)
f_{22}	best	*	*	*	*
	mean	* (0.00)	* (0.00)	* (0.00)	* (0.00)