Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

Multi-Particle Collision Algorithm with Reflected Points

Reynier Hernández Torres, **Eduardo Fávero Pacheco da Luz**, **Haroldo Fraga de Campos Velho**
Laboratório Associado de Computação e Matemática Aplicada (LAC)
Instituto Nacional de Pesquisas Espaciais (INPE)
12227-010, São José dos Campos, SP
E-mail: reynierhdez@gmail.com, haroldo@lac.inpe.br, eduardo.luz@lac.inpe.br

Resumo: *New versions for the MPCA (Multi-Particle Collision Algorithm) meta-heuristic are presented. In order to provide more effective candidate solutions for an optimization problem, the concept of opposition and reflection is introduced to improve the capacity of search space for the MPCA. Four different strategies to compute the reflected and/or opposited points are implemented. The performance of all implementation are evaluated with four objective functions.*

Key-words: *stochastic optimization, opposition, reflection, particle collision algorithm*

Introduction

Optimization is an area of the Applied Mathematics that studies the theory and techniques to finding better available values to minimize or maximize some objective function, given a defined domain. Many real problems in science involves optimization of any type. These problems can be classified according the nature of their variables and parameters as continuous (variables have real or continuous values), discrete (variables have discrete or integer values) or mixed optimization (variables can be both continuous or discrete).

Optimization can be divided in two large areas: *deterministic* (exact methods) and *stochastic* optimization (uses random processes). Stochastic optimization facilitates the exploration of the search space, while an exploitation (intense search) is made. The search is made by randomly generated candidate solution visiting the entire search space, while a local search is made in a small neighbourhood for a such candidate solution. The objective of this strategy is to avoid to converge to a local optimum and to continue searching the global optimum.

With the advances of the computation, many algorithms have been developed in the sub-area of the stochastic optimization. Those algorithms are called to improve the exploration, making it more efficient, converging quickly to the global optimum.

This work introduces a new variant of the Multi-Particle Collision Algorithm that exploits the advantages of the parallel computation and the opposition and reflection concepts.

Multi-Particle Collision Algorithm (MPCA)

The MPCA [2] is based on the canonical Particle Collision Algorithm (PCA), introduced by Sacco[5].

The PCA algorithm is inspired by the physics of nuclear particle traveling inside of a nuclear reactor, particularly the scattering and the absorption phenomena. In this algorithm, the Perturbation function performs a random variation of the solution within a defined range (this allows the *visit* on different regions in the search space), while the Exploration function performs a local search (applying an small perturbation on the candidate solution). When the

DOI: 10.5540/03.2015.003.01.0433

new candidate solution has a worse performance (the cost function is enhanced), the Scattering process is activated. The particle (the candidate solution) is replaced by a new random solution, according a probability computed from: $[1 - \text{cost\_function}/(\text{best solution})]$ [2, 5].

The MPCA increases a simple characteristic to the original PCA: the use of more than one particle to explore the search space. Also, a Blackboard strategy is implemented, where the best particle is overcopied for all other particles. The process is re-started at each $n_{blackboard}$ iterations, as seen in Algorithm 1.

Algorithm 1 MPCA

```
for i ← 1, n_processors do
    for j ← 1, n_particles do
        Generate an initial solution OldConfig_{i,j}
    end for
end for
while Stopping criteria not yet met do
    for i ← 1, n_processors do
        if it is time to update the blackboard (each n_blackboard iterations) then
            UpdateBlackboard()
        end if
        for j ← 1, n_particles do
            Perturbation()
            if Fitness(NewConfig_{i,j}) > Fitness(OldConfig_{i,j}) then
                if Fitness(NewConfig_{i,j}) > BestFitness_i then
                    BestFitness_i = Fitness(NewConfig_{i,j})
                    BestConfig_i = NewConfig_{i,j}
                end if
                OldConfig_{i,j} = NewConfig_{i,j}
                Exploration()
            else
                Scattering()
            end if
        end for
    end for
end while
UpdateBlackboard()
return BestConfig
```

In the MPCA, the number of evaluation of the functions is divided by the number of particles ($n_{particles}$) in the population. That is an advantage is this variant of the algorithm, due to a considerable reduction of computing time, by the use of $n_{processors}$ processors.

The MPCA is implemented in FORTRAN using MPI libraries in a multiprocessor architecture with distributed memory machine.

MPCA using Opposition Based Learning

In this section, four new variants of the MPCA are introduced. These are the Opposite MPCA (O-MPCA), Quasi Opposite MPCA (QO-MPCA), Quasi Reflective MPCA (QR-MPCA) and the Center Based Sampling MPCA (CB-MPCA).

These new versions of the algorithm make possible to do a more intense exploration of the search space, using the oppositions concepts working together with randomness. Opposition is applied in certain time, in order to reduce the computational time, since it doubles the function evaluation number each iteration.

DOI: 10.5540/03.2015.003.01.0433

Algorithm 2 MPCA with opposition (quasi-opposition, quasi-reflection or center-based sampling)

```
for i ← 1, n_processors do
    for j ← 1, n_particles do
        Generate an initial solution OldConfig_{i,j}
        Determinate opposite solution QOldConfig_{i,j}
        if Fitness(QOldConfig_{i,j}) < Fitness(OldConfig_{i,j}) then
            OldConfig_i ← QOldConfig_i
        end if
    end for
end for
UpdateBlackboard()
while Stopping criteria not yet met do
    for i ← 1, n_processors do
        for j ← 1, n_particles do
            Perturbation()
            if Fitness(NewConfig_{i,j}) < Fitness(OldConfig_{i,j}) then
                if Fitness(NewConfig_{i,j}) < BestFitness_i then
                    BestFitness_i = Fitness(NewConfig_{i,j})
                    BestConfig_i ← NewConfig_{i,j}
                end if
                OldConfig_{i,j} ← NewConfig_{i,j}
                Exploration()
            else
                Scattering()
            end if
            if it is time to get the opposite population (each n_opposition iterations) then
                Determinate opposite solution QOldConfig_{i,j}
                if Fitness(QOldConfig_{i,j}) < Fitness(OldConfig_{i,j}) then
                    OldConfig_i ← QOldConfig_i
                end if
            end if
        end for
    end for
    if it is time to update the blackboard (each n_blackboard iterations) then
        UpdateBlackboard()
    end if
end while
BestConfig_overall ← UpdateBlackboard()
return BestConfig_overall
```

Opposition Based Learning and derivates

Opposition-based learning (OBL) was proposed in [6]. OBL has been applied to many evolutionary algorithms such as Differential Evolution [4], Particle Swarm Optimization [7], Ant Colony Optimization [3] and Biogeography-Based Optimization [1].

The opposite point $P_o(x_{o1}, x_{o2}, \cdots, x_{on})$ of a point $P(x_1, x_2, \cdots, x_n)$, in a $\mathbb{R}^n$ space is completely defined by its components

$$x_{oi} = L_i + U_i - x_i \tag{1}$$

where $x_i \in \mathbb{R}, L_i \leq x_i \leq U_i, \forall i \in \{1, 2, \cdots, n\}$. L_i and U_i are the lower and upper limits for the variable x_i.

Also, a quasi-opposite point P_{qo}, a quasi-reflective point P_{qr}, and a center-based sampling point P_{cb}, are defined below:

$$P_{qo}(x_{qo1}, x_{qo2}, \cdots, x_{qon}) \mid x_{qoi} = \text{rand}(M_i, x_{oi}) \tag{2}$$

$$P_{qr}(x_{qr1}, x_{qr2}, \cdots, x_{qrn}) \mid x_{qri} = \text{rand}(M_i, x_i) \tag{3}$$

$$P_{cb}(x_{cb1}, x_{cb2}, \cdots, x_{cbn}) \mid x_{cbi} = \text{rand}(x_i, x_{oi}) \tag{4}$$

where M_i is the center of the interval $[L_i, U_i]$ and can be calculated as $M_i = (L_i + U_i)/2$, and $rand(\cdot)$ is a is a random number uniformly distributed between the first and second number in the argument.

Opposite-based population initialization and generation frequency

Random number generation is commonly the most used choice to create an initial population. By utilizing opposition working together with randomness, it may obtain better starting candidates even when there is no a priori knowledge about the solution. In this phase, the first step is to create the initial solution for each particle as usual. Next, the opposite solution is calculated within the original search space $[L_i, U_i]$. The original solution is substituted by the opposite solution if the second one has a better fitness.

After applying Perturbation, Exploration and Scattering functions to generate the new solution of a particle, the same procedure is used to get the opposite solution and replacing if the solution is better. That process is made each $n_{opposition}$ iterations, as seen in Algorithm 2.

Empirical Analysis

Benchmark Functions

Experiments were made in a server with a Intel(R) Xeon(R) X5570 microprocessor at 2.93GHz, with 16 cores and 32 GB of RAM memory. OpenMPI libraries were used for parallel processing.

In this experiment set, was used a configuration of eight processes, each one with a particle, resulting eight particles in the population each iteration.

Four benchmark functions with ten dimensions ($n = 10$) were implemented to compare the performance of MPCA and their variants with opposition. Information about these functions is shown in Table 1.

Function	Domain	Argmin	min $f(x)$
Ackley	$(-32.768, 32.768)^n$	0^n	0
Griewank	$(-600, 600)^n$	0^n	0
Rastrigin	$(-5.12, 5.12)^n$	0^n	0
Sphere	$(-10, 10)^n$	0^n	0

Table 1: Benchmark functions

All the algorithms are terminated when the number of function evaluation exceeds the predetermined maximum number $100 \times n_{dimensions}$.

A trial is considered successfully if $\mid f^* - \widehat{f} \mid < \epsilon \mid f^* \mid + \epsilon$, where f^* is the best function value obtained by the algorithm, $\widehat{f}$ is the known exact global minimum and ϵ is an small positive number, here set to 10^{-4}.

Statistic	Objective function			
	Ackley	Rastrigin	Griewank	Sphere
		MPCA		
mean	$1{,}2204\times10^{-14}$	$2{,}6716\times10^{-14}$	$4{,}5295\times10^{-16}$	$1{,}6733\times10^{-44}$
median	$1{,}3767\times10^{-14}$	$2{,}8422\times10^{-14}$	$*4{,}4407\times10^{-16}$	$1{,}5812\times10^{-44}$
minimum	$6{,}6613\times10^{-15}$	$1{,}4211\times10^{-14}$	$4{,}4406\times10^{-16}$	$7{,}4263\times10^{-45}$
maximum	$1{,}3767\times10^{-14}$	$5{,}6843\times10^{-14}$	$6{,}6607\times10^{-16}$	$2{,}4595\times10^{-44}$
std. dev.	$2{,}3115\times10^{-15}$	$*8{,}5265\times10^{-15}$	$4{,}4400\times10^{-17}$	$4{,}8439\times10^{-45}$
		O-MPCA		
mean	$1{,}2488\times10^{-14}$	$3{,}0695\times10^{-14}$	$4{,}7070\times10^{-16}$	$1{,}5908\times10^{-44}$
median	$1{,}3767\times10^{-14}$	$2{,}8422\times10^{-14}$	$*4{,}4407\times10^{-16}$	$1{,}6832\times10^{-44}$
minimum	$6{,}6613\times10^{-15}$	$1{,}4211\times10^{-14}$	$4{,}4402\times10^{-16}$	$3{,}9793\times10^{-45}$
maximum	$1{,}3767\times10^{-14}$	$5{,}6843\times10^{-14}$	$6{,}6609\times10^{-16}$	$2{,}5590\times10^{-44}$
std. dev.	$2{,}2656\times10^{-15}$	$8{,}8747\times10^{-15}$	$7{,}3617\times10^{-17}$	$5{,}3073\times10^{-45}$
		QO-MPCA		
mean	$*2{,}6823\times10^{-15}$	$*1{,}7053\times10^{-14}$	$*4{,}4407\times10^{-16}$	$*4{,}0569\times10^{-45}$
median	$*3{,}1086\times10^{-15}$	$*1{,}4211\times10^{-14}$	$*4{,}4407\times10^{-16}$	$3{,}9177\times10^{-45}$
minimum	$*4{,}4409\times10^{-16}$	$*0{,}0000$	$*4{,}4402\times10^{-16}$	$*6{,}0764\times10^{-46}$
maximum	$*3{,}1086\times10^{-15}$	$*2{,}8422\times10^{-14}$	$*4{,}4409\times10^{-16}$	$7{,}6090\times10^{-45}$
std. dev.	$1{,}1783\times10^{-15}$	$1{,}2307\times10^{-14}$	$*1{,}4617\times10^{-20}$	$*1{,}9541\times10^{-45}$
		QR-MPCA		
mean	$*2{,}6823\times10^{-15}$	$2{,}1600\times10^{-14}$	$4{,}7070\times10^{-16}$	$4{,}2618\times10^{-45}$
median	$*3{,}1086\times10^{-15}$	$2{,}8422\times10^{-14}$	$*4{,}4407\times10^{-16}$	$3{,}8554\times10^{-45}$
minimum	$*4{,}4409\times10^{-16}$	$*0{,}0000$	$4{,}4404\times10^{-16}$	$8{,}7317\times10^{-46}$
maximum	$*3{,}1086\times10^{-15}$	$8{,}5265\times10^{-14}$	$6{,}6603\times10^{-16}$	$*7{,}0481\times10^{-45}$
std. dev.	$1{,}1783\times10^{-15}$	$1{,}8401\times10^{-14}$	$7{,}3608\times10^{-17}$	$1{,}9665\times10^{-45}$
		CB-MPCA		
mean	$2{,}8244\times10^{-15}$	$2{,}1032\times10^{-14}$	$4{,}7070\times10^{-16}$	$4{,}1577\times10^{-45}$
median	$*3{,}1086\times10^{-15}$	$2{,}8422\times10^{-14}$	$*4{,}4407\times10^{-16}$	$*3{,}4423\times10^{-45}$
minimum	$*4{,}4409\times10^{-16}$	$*0{,}0000$	$4{,}4404\times10^{-16}$	$1{,}3082\times10^{-45}$
maximum	$*3{,}1086\times10^{-15}$	$2{,}8422\times10^{-14}$	$6{,}6608\times10^{-16}$	$8{,}5136\times10^{-45}$
std. dev.	$*9{,}8370\times10^{-16}$	$1{,}0946\times10^{-14}$	$7{,}3610\times10^{-17}$	$2{,}0061\times10^{-45}$

Table 2: Mean, minimum, maximum and median values of the objective function for 100000 function evaluations for 25 trials.

Table 2 shows the statistics results for 25 trials of each algorithm, each one with different initial populations and random seeds. In almost the cases, the variant with quasi-opposition (QO-MPCA) obtained the best results for those statistics analysed, closely followed by the QR-MPCA and CB-MPCA. The O-MPCA does not bring results of significant relevance over the MPCA. All trials for all algorithms were successful.

Conclusions

In this preliminary study, the algorithms with opposition and reflection mixed with randomness have obtained better results than MPCA and MPCA with simple opposition, over four functions of simple and median complexity in a space with 10 dimensions.

In further studies, non-parametric statistic testes will be made to analyse the relevance of the results, more objective function will be used, increasing the complexity, and some new variants and configurations of the algorithm MPCA will be implemented.

References

[1] M. Ergezer, D. Simon, and D. Du, Oppositional Biogeography-Based Optimization, Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, USA, 2009

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

[3] A. R. Malisia, Investigating the application of opposition-based ideas to ant algorithms, Master's thesis, University of Waterloo, 2007.

[4] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, Quasi-Oppositional Differential Evolution, IEEE Congress on Evolutionary Computation, CEC 2007. Pattern Anal. & Machine Intelligence (PAMI), Waterloo, 2007.

[5] W. F. Sacco, and C. R. de Oliveira, A new stochastic optimization algorithm based on a particle collision metaheuristic. In Proceedings of 6th World Congress of Structural and Multidisciplinary Optimization, Rio de Janeiro. WCSMO, 2005

[6] H. R. Tizhoosh, Opposition-Based Learning: A New Scheme for Machine Intelligence, International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05), vol. 1, pp. 695–701, 2005.

[7] H. Wang, Y. Liu, S. Zeng, H. Li, and C. Li, Opposition-based particle swarm algorithm with cauchy mutation, in IEEE Congress on Evolutionary Computation, Singapore, pp. 4750–4756, 2007.

DOI: 10.5540/03.2015.003.01.0433