Multiple Particle Collision Algorithm applied to Radiative Transference and Pollutant Localization Inverse Problems

Eduardo Fávero Pacheco da Luz
Post-graduation program in Applied Computing
National Institute for Space Research
São José dos Campos/SP - Brazil
eduardo.luz@lac.inpe.br

José Carlos Becceneri, Haroldo Fraga de Campos Velho
Computing and Applied Mathematics Associated Laboratory
National Institute for Space Research
São José dos Campos/SP - Brazil
{becce, haroldo}@lac.inpe.br

Abstract—The Multiple Particle Collision Algorithm (MPCA) is a nature-inspired stochastic optimization method developed specially for high performance computational environments. Its advantages resides in the intense use of computational power provided by multiple processors in the task of search the solution space for a near optimum solution. This work presents the application of MPCA in solving two inverse problems written as optimization problems, its advantages and disadvantages are also described, so are the obtained results.

Keywords-Metaheuristic, Optimization, High Performance Computing, Inverse Problems.

I. INTRODUCTION

The theory of optimization is a branch of the mathematical sciences that studies the methods for finding an optimum set for a given problem. The practical part of the theory is defined by the collection of techniques, methods, procedures and algorithms that can be used to find the optimum [1].

Optimization problems has the goal of finding the best set within a variable set to maximize or minimize a function, defined as an objective function or cost function. Optimization problems can be classified as [2]:

- Continuous optimization: where variable has real or continuous values;
- Discrete optimization: where variable has integer or discrete values; and
- Mixed optimization: with integer and continuous values at the same time.

The best method for determining the function optimum strongly depends on the nature of the function in study. Two kinds of algorithms can be used: local optimization algorithms, that given a point in a function sub domain are able to find the optimum point in this sub domain (most of this kind of algorithms are deterministic); and global optimization algorithms, that seek the optimum point in the totality of the search space (those are frequently stochastic algorithms, mostly metaheuristics).

Deterministic algorithms used in optimization can be divided into three main types:

- Zero order methods: based on the value of the objective function, e.g., Powell’s conjugated directions;
- First order methods: based on the value of the objective function and its derivative regarding the project’s variables, e.g., Steepest Descent;
- Second order methods: based on the value of the objective function, its derivative and the Hessian matrix, e.g., Quasi-Newton.

Stochastic methods used in optimization are those based on heuristics. The word “heuristic” came from the Greek heuriskein, meaning “to discover”, and describing a method “based on the experience or judgement, that leads to a good solution of a problem, but not assuring to produce the optimum solution” [14]. Metaheuristics can be described as those heuristics strongly based on natural processes.

The main metaheuristics are Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), among others [10].

II. PARTICLE COLLISION ALGORITHM (PCA)

The Particle Collision Algorithm (PCA) was developed by Sacco and co-authors [19], [18], [20], [21], inspired in some basic characteristics of Simulated Annealing [10].

This algorithm was also greatly inspired by two physical behaviours, namely absorption and scattering, that occurs inside a nuclear reactor. The use of the PCA was effective for several test functions and real applications [21].

The PCA starts by selecting an initial solution (Old_Config), that is modified by a stochastic perturbation (Perturbation()), leading to the construction of a new solution (New_Config). The new solution is compared to the old one (the solutions are compared by calculating the fitness of each one with function Fitness()), and the new solution can or cannot be accepted. The main algorithm can be seen in Figure 1.

If the new solution is not accepted, a Metropolis scheme is used (the function Scattering()). The exploration on closer positions is guaranteed by using the functions Perturbation() and Small_Perturbation(). The latter algorithms are shown by Figure 2.
If a new solution is better than the previous one, this new solution is absorbed (absorption is one feature involved in the real collision process). If a worst solution is found, the particle can be sent to a different location of the search space, giving the algorithm the capability of escaping a local minima, this procedure is inspired on the scattering scheme.

PCA is a robust metaheuristic, only a few parameters are required from the user. The main parameter is the number of iterations, and also acts as a stop criteria for the algorithm. According to [20],[21] 10^5 iterations should be a good value for almost every application.

Analysing the main algorithm of PCA, presented at Figure 1, and adopting \( N \) as the number of iterations defined by user, we can see that the first loop induces \( N \) checking operations over the objective function. At this main loop, there are calls to the Exploration() procedure, that by its turn executes more \( N \) operations through a loop inside the procedure. Therefore, \( N \times N \) operations are executed by PCA, leading to a \( O(N^2) \) complexity.

III. MULTIPLE PARTICLE COLLISION ALGORITHM (MPCA)

The new Multiple Particle Algorithm (MPCA) is based on the canonical PCA, but a new characteristic is introduced: the use of several particles, instead of only one particle to act over the search space.

Coordination between the particles was able through a blackboard strategy, where the Best_Fitness information is shared among all the particles in the process.

The pseudo-code for the MPCA is presented by Figure 3, where the new loop, responsible for the control of the new particles is introduced.

Similar to the PCA, MPCA also have only few parameters to be determined, added to the number of particles to be used. But in this case, the total number or iterations is divided by the number of particles which will be used in the process. The division of task is the great distinction of MPCA, which leads to a great reduction of required computing time.

The MPCA was implemented using MPI libraries in a multiprocessor architecture with distributed memory.

The code of MPCA, presented by Figure 3, is very similar to PCA, so there are \( N \times N \) checking operations in the inner loops, but due to the new loop, introduced by the multiple particle technique, that number of checking operations can be increased to a case where \( N \times N \times N \) operations can occur, e.g., the number of particles is equal to the number of iterations.

So, the complexity associated to MPCA is initially \( O(N^3) \), but when we distribute the algorithm through the use of \( p \) processors, making sure that \( p = n \), as the number of particles in use, the complexity can return to \( O(N^2) \), which is the same complexity of the canonical PCA.
IV. INVERSE PROBLEMS

While dealing with the mathematical formulation of a direct problem we start with from an initial situation and came, by the application of some methodology, calculation or equation, to the effects generated by the modelled phenomenon in the direct problem.

To solve an inverse problem is to determine unknown causes taking in consideration desired or observed effects [4]. This is one of the most used definitions, leading to the conclusion that studying inverse problems is consisted of the use of observed results to estimate some parameter values that can characterize the system under investigation.

In this way, if we admit that a given mathematical model can be expressed as \( A(u) = f \), the inverse model related to this problem can be expressed as \( A^{-1}(f) = u \). At Fig. 4 the graphical representation of a direct problem is given by the path that connects the space of causes to the space of effects and the inverse problem connects the space of effects to the space of causes.

Most of the inverse problems belongs to an ill-posed class of problems, i.e., those kind of problems that violates one or more Hadamard condition, which are:

- Existence: the problem must have a solution;
- Uniqueness: the solution must be unique;
- Stability: the dependency between data must be continuous.

Breaking the third condition, stability, leads to situation that can be classified as “ill-conditioning” that can be solved with the use of regularization. More information about the theory of regularization can be obtained in [24].

Now, we are going to present the inverse problems used in this paper in order to validate the MPCA.

A. Radiative transference

The first problem that we present is an inverse problem for radiative transference in a parallel-plane homogeneous media (Fig. 5). More information about the formulation of this problem can be obtained in [23].

The parameters to be estimated are:

\[
\vec{Z} = \{ \tau_0, \omega, \rho_1, \rho_2 \}^T
\]

where \( \tau_0 \) represents the optical thickness, \( \omega \) represents the scattering albedo and \( \rho_1 \) and \( \rho_2 \) represents the diffuse reflectivity.

The optimization problem in this case is written as the minimization of square errors.

\[
Q(\vec{Z}) = \sum_{i=1}^{N_d} [I_i(\vec{Z}) - Y_i]^2
\]

where \( I_i \) corresponds to the value calculated by the model for the candidate solution and \( Y_i \) represents the experimentally measured value.

This is called an implicit formulation, for the parameters does not appear directly on the formulation of the solution, but are included in the formulation of the correspondent direct problem, and its effects are perceived when the direct problem is solved and then participate in the composition of the objective function [23].

B. Localization of polluting sources

To represent the particle dispersion in the atmosphere, in this second problem, we adopted a Lagrangian Model for Buoyant Dispersion in Atmosphere (LAMBDA), which is based in the three-dimensional form of the Langevin equation for a random velocity field, according to Thomson’s derivation [16].

The model for particle dispersion (LAMBDA) simulates a certain quantity of computational particles that emulates the behaviour of real particles for a generic atmospheric contaminant (Fig. 6).

The temporal integration for the Lagrangian dispersion calculates the mean concentration of a contaminant at the position \( \vec{x} \) at time \( t \), giving a certain emission rate in a source \( (S(kgm^{-3}s^{-1})) \), being defined as:

\[
C(\vec{x}, t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S(\vec{x}_0, t_0) P^a(\vec{x}, t|\vec{x}_0, t_0) d\vec{x}_0 dt_0
\]

where \( P^a(\vec{x}, t|\vec{x}_0, t_0) \) is the time forward probability density, defined such as \( P^a(\vec{x}, t|\vec{x}_0, t_0) d\vec{x}_0 \) be the probability that an...
The LAMBDA model implements the calculation of concentration in a given sensor as:

\[ C_j = \sum_{i=1}^{N_f} S_i \frac{V_{f,i}}{V_{s,j}} N_{PEF,i} N_{PVS,i,j} \]  

(4)

where \( C_j \) represents the concentration of the \( j \)-th sensor, \( N_f \) the number of sources, \( S_i \) the intensity of the \( i \)-th source, \( V_{f,i} \) the volume of the \( i \)-th source, \( V_{s,j} \) the volume of the \( j \)-th sensor, \( \Delta t \) the temporal discretization, \( N_{PEF,i} \) the number of emitted particles at the \( i \)-th source and \( N_{PVS,i,j} \) is the number of emitted particles at the \( i \)-th source that can be found in the \( j \)-th sensor.

To solve this problem as an optimization problem, we use a source-receptor model, that reduced the computational effort required by the iterative resolution of the direct model. Therefore, we calculate:

\[ \bar{C} = M \bar{S} \]

where \( \bar{C} \) is a vector of elements that represents the mean concentration in the sensors, \( M \) represents the state transition matrix, \( \bar{S} \) represents the intensity at the emission sources or absorption points. The \( M \) matrix is built based on the Eq. 4, defined as:

\[ M_{ij} = \frac{V_{f,i}}{V_{s,j}} \frac{\Delta t}{N_{PEF,i}} N_{PVS,i,j} \]  

(6)

V. RESULTS

The results presented in this section considers the mean of 10 experiments using distinct random number generation seeds and experimental data artificially generated by the use of the results from the direct model added by Gaussian noise, intending to simulate the use of real instrumentation. The adopted noise level was of 2% seeking to demonstrate the viability of MPCA in the solution of inverse problems.

The used parameter was: 8 particles; 8 processors; 1000 iterations; 1000 local search steps; Gaussian perturbation radius \( N(0,1) \); local search radius corresponding to 20% of the Gaussian perturbation radius. The algorithm was executed in a Cray XT5.

Table I presents the results for the radiative transference inverse problem. We can notice that the parameter \( \rho_1 \) had the worst estimative, but this was an expected result [23].

<table>
<thead>
<tr>
<th>Exact result</th>
<th>( \omega )</th>
<th>( \varphi_1 )</th>
<th>( \varphi_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean result</td>
<td>0.9954</td>
<td>0.6144</td>
<td>0.1620</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>2.51E-2</td>
<td>2.42E-2</td>
<td>8.09E-2</td>
</tr>
</tbody>
</table>

Tables II and III presents the results for the localization of two emission/absorption polluting sources. The sources alternates its behaviour (rate) of emission to absorption and this is properly captured by this optimization algorithm.

<table>
<thead>
<tr>
<th>Exact result</th>
<th>Area 1 - Rate 1</th>
<th>Area 1 - Rate 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean result</td>
<td>0.8875</td>
<td>-0.4114</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.1163</td>
<td>1.18E-2</td>
</tr>
</tbody>
</table>

Table II

<table>
<thead>
<tr>
<th>Exact result</th>
<th>Area 2 - Rate 1</th>
<th>Area 2 - Rate 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean result</td>
<td>0.5127</td>
<td>-0.8820</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1.96E-2</td>
<td>6.59E-2</td>
</tr>
</tbody>
</table>

Table III

VI. CONCLUSION

The MPCA showed itself a viable alternative to the solution of inverse problems, specially when there is high performance computers available.

The actual existence of personal computers with multicore architectures also enables the execution of an algorithm developed for high performance environments without the loss of performance, giving the user a better use of the available resources.

The results also demonstrates the convergence of MPCA for a good solution within a reasonable amount of available resources and opens the possibility of further studies, including hybridization, e.g., substitution of the actual local search method which is solely based in the generation of random neighbour solution within a given radius.

ACKNOWLEDGMENT

The authors would like to thank CAPES that partially funded this work.
REFERENCES


