A Multi-objective Version of the Multiple Particle Collision Algorithm

Eduardo F. P. Luz*, Amarisio da S. Araujo†, Juliana A. Anochi* and Haroldo F. Campos Velho*
*Associated Laboratory for Computing and Applied Mathematics (LAC)
National Institute for Space Research (INPE), São José dos Campos, SP, Brazil, 12227-010
Email: {eduardo.luz, juliana.anochi, haroldo}@lac.inpe.br
†Department of Mathematics
Federal University of Viçosa, Viçosa, MG, Brazil, 36570-000
Email: amarisio@ufv.br

***Abstract*—A multi-objective version of a meta-heuristic, loosely inspired on the behaviour of particles inside a nuclear reactor, is presented. The Multi-objective Multiple Particle Collision Algorithm (MMPCA) uses the Pareto based fitness assignment, which uses the concept of dominance, to generate new solutions and build the Pareto set. The original population is duplicated, with purpose of classification and for applying the crowding distance approach. The latter procedures are also used in the NSGA-II.**

I. Introduction

Almost every real-world problem involves the optimization of several incommensurable and often conflicting objectives. For this reason, multi-objective optimization has become an important topic in optimization theory and, in general, a topic very challenging for researchers from several applied sciences [1].

Several techniques have been proposed to solve multi-objective problems optimization by using meta-heuristics and evolutionary procedures. The increasing use of these procedures, during the last decade, can be explained by some of their important characteristics, for example, the ability to work without derivatives, find multiple solutions in a single run, escape local optimum, high convergence rate, amongst others.

In this paper we present a new meta-heuristic for multi-objective problems. The Multiple Particle Collision Algorithm (MPCA) [2] is a population-based method, loosely inspired in the collision behaviour of particles inside a nuclear reactor. Here we merge the MPCA with some procedures introduced at the genetic algorithm NSGA-II [3] to obtain a good approximation of the Pareto front.

II. Multi-objective Optimization

A Multi-objective Optimization Problem (MOP) can be formally stated as: find the points $\bar{x} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$ of decision variables $x \in \Omega$ that simultaneously optimize the m objective functions $f_1, f_2, ..., f_m$, where $\Omega \subseteq R^n$ is the space of feasible solutions to the problem, resulting from some constraints on the variables, if any.

Unlike single objective optimization, the solution to the MOP is not a single point $\bar{x}$, but a set of points known as Pareto set [4]. Such points are considered to be equally important solutions to the MOP in the sense that they have the property that moving from one point to another results in the improvement of one objective function while causing to degradation in the value of at least one of the remaining objective functions. The points in the Pareto set constitute the global optimum solutions to the MOP. Once identified the Pareto set, a decision maker, based on the characteristics of the problem, can select a "compromise" solution satisfying the objective functions as best possible.

An important concept in this context, used to define the Pareto set, is that of domination [4]. It is said that the solution x^* dominates the solution x when x^* is not worse than x for all the objective functions, and x^* is better than x for at least one of the objective functions. For a minimization problem, for example, the solution x^* dominates the solution x if $\forall$ $i \in \{1, 2, ..., m\}$, $f_i(x^*) \leq f_i(x)$, and $\exists$ $k \in \{1, 2, ..., m\}$ such that $f_k(x^*) < f_k(x)$. Considering a set of solutions P, those which are not dominated by any member of the set P constitute a subset $P^{'} \subseteq P$ of non-dominated solutions. When P is the entire search space, $P^{'}$ constitutes the Pareto set. Mapping this set in the objective space we have the surface called Pareto front.

When solving a MOP, the goal is, therefore, to obtain the Pareto set (equivalently, the Pareto front). For many practical problems, to obtain explicit analytic solutions to the Pareto set is a difficult task and, in general, an impossible task. Several techniques have been proposed to obtain a good approximation of the Pareto set by a finite number of solutions. Some of them consist in reducing the problem to a single objective problem by introducing different weights for each objective function. The techniques that have shown the most efficient are those based on the concept of the Pareto's ranking, which allows to find several points in the Pareto set in only one step. The idea of using this Pareto-based fitness assignment was first proposed by Goldberg [5].

Several evolutionary algorithms have been successfully using the concepts presented here. One of the most known is the Non-dominated Sorting Genetic Algorithm (NSGA-II) [3].

III. Multiple Particle Collision Algorithm

The Multiple Particle Collision Algorithm (MPCA) is a population-based meta-heuristic [2]. It is based on the single-solution version PCA [6]. They are greatly inspired by two physical behaviours, namely absorption and scattering, that occurs inside a nuclear reactor. In both algorithms, there are similarities with basic characteristics of Simulated Annealing [7], [8]. In MPCA, coordination between the multiple particles is implemented using a blackboard (or mailbox) strategy, where the information on the best solution is shared among all the particles in the process.

The MPCA starts by selecting a set of initial solutions (size N), that are modified by a stochastic perturbation, leading to the construction of a new set of solutions. The new solutions are compared to the old ones (the solutions are compared by calculating the fitness of each one), and the new solutions can or cannot be accepted.

If the new solutions are not accepted, a Metropolis scheme is used. The exploration on closer positions is guaranteed by using local perturbation functions.

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 send to a different location of the search space, giving the algorithm the capability of escaping a local optimum, this procedure is inspired on the scattering mechanism.

The MPCA is intended to be implemented using Message Passing Interface (MPI) libraries in a multiprocessor architecture with distributed memory.

IV. Multi-objective MPCA

Taking as example the procedures adopted by NSGA-II [3] to double the size of the initial population (size N), the Multi-objective Multiple Particle Collision Algorithm (MMPCA) generates a new set of solutions (size N), using the absorption and scattering mechanisms (from mono-objective MPCA) and the dominance concept from MOP theory.

Once generated, the two sets of solutions are joined, creating a new set containing $2N$ solutions which are sorted by Pareto's ranking [5] generating several fronts, one for each rank. On a second step we calculate the crowding distance [3] within a rank and sort these solutions based on this distance.

With the $2N$ solutions, sorted by rank and crowding distance, the algorithm iterates over the top N solutions (discarding the bottom N solutions).

At the end of execution, in the final set of top N solutions, we will have p solutions with the best rank which will constitute our approximation of the Pareto set. The pseudo-code for MMPCA is presented in Fig. 1.

```
Master-process generates p-solutions
Update ParetoBlackboard
  Each Slave-process receives initial Old_Config from Master -process
  Best_Fitness = Fitness(Old_Config)
For n=0 to # of iterations
  For p=0 to # of particles
    Update ParetoBlackboard, on comm cycle
      Each Slave-process send Best_Config to Master -process
      Master-process receives and sort 2p solutions
      Master-process resends the best p solutions to Slave-process
    Perturbation()
    If Fitness(New_Config) < Fitness(Old_Config)
      If Fitness(New_Config) < Best_Fitness
        Best_Fitness = Fitness(New_Config)
      End-If
      Old_Config = New_Config
      Exploration()
    Else
      Scattering()
    End-IF
  End-For
End-For
Store the Pareto front

Perturbation()
  For i=0 to (Dimension-1)
    New_Config[i] = Old_Config[i] + ((Upper_Limit[i] - Old_Config[i]) *
      Rand(0,1)) - ((Old_Config[i] - Lower_Limit[i]) * (1-Rand(0,1))
    If (New_Config[i] > Upper_Limit[i])
      New_Config[i] = Upper_Limit[i]
    Else-If (New_Config[i] < Lower_Limit[i])
      New_Config[i] = Lower_Limit[i]
    End-If
  End-For
Return

Exploration()
  For n=0 to # of iterations
    Small_Perturbation()
    If Fitness(New_Config) < Fitness(Old_Config)
      If Fitness(New_Config) < Best_Fitness
        Best_Fitness = Fitness(New_Config)
      End-If
      Old_Config = New_Config
    End-If
  End-For
Return

Small_Perturbation()
  For i=0 to (Dimension-1)
    Upper = Rand(1.0,1.2) * Old_Config[i]
    If (Upper > Upper_Limit[i])
      Upper = Upper_Limit[i]
    End-If
    Lower = Rand(0.8,1.0) * Old_Config[i]
    If (Lower < Lower_Limit[i])
      Lower = Lower_Limit[i]
    End-If
    New_Config[i] = Old_Config[i] + ((Upper - Old_Config[i]) *
      Rand(0,1)) - ((Old_Config[i] - Lower) * (1-Rand(0,1))
  End-For
Return

Scattering()
  pscat = 1 - (Best_Fitness / Fitness(New_Config))
  If (pscat > Rand(0,1))
    Old_Config = Random Solution
  Else
    Exploration()
  End-If
Return
```

Fig. 1. Pseudo-code for MMPCA.

V. Final remarks

Initial tests using benchmark functions reveals great potential for the algorithm presented here. Further tests are been conducted to ensure the competitiveness of MMPCA.

Since MMPCA is a Metropolis-based method, we are also exploring different ways to enable the scattering procedure.

Finally, the application of MMPCA in real-world problems, such as: calibration of prediction models employed on climate prediction, climate change, and hydrology, will be our future applications.

Acknowledgement

The authors would like to thank CNPq, LAC-INPE and Federal University of Viçosa.

References

[1] E. Zitzler. PhD thesis. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Shaker Verlag, ETH Zurich, Switzerland(1999).

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

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*, IEE Transactions on Evolutionary Computation, 6 (2) (2002) 182-197

[4] W. Stalder, *A survey of multicriteria optimization or the vector maximum problem, part I: 1776-1960* Journal of Optimization Theory and Applications 29 (1) (1979) 1-52

[5] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* Reading, Massachusetts: Addison-Wesley, 1989

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

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. Science, New Series, 220 (4598) (1983) 671-680.

[8] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics, 21 (6) (1953) 1087-1092.