PARTICLE SWARM OPTIMIZER IN NONSMOOTH PROBLEMS

K.E. PARSOPOULOS & M.N. VRAHATIS
Department of Mathematics,
University of Patras Artificial Intelligence
Research Center (UPAIRC),
University of Patras,
GR–26110 Patras, Greece
email: {kostasp, vrahatis}@math.upatras.gr

1. SUMMARY

In Mechanics applications, nonsmooth/nonconvex functions that need to be minimized, are frequently encountered. These functions are not differentiable, and consequently cannot be directly addressed through classical optimization algorithms. A very common approach for solving such problems is the implementation of convex optimization methods applied on piecewise linear approximations of the objective function. However, most techniques in this category incorporate assumptions on the Lipschitz continuity of the objective function. Recently, Evolutionary Algorithms (EAs) have been applied on nonsmooth/nonconvex problems, and exhibited very promising results. A study of the performance of the Particle Swarm Optimization (PSO) method for solving nonsmooth/nonconvex problems forms the core of this paper. Several well–known and widely used test problems are considered and experimental results are reported. Ideas for further work and applications of PSO on Mechanics problems are discussed.

2. INTRODUCTION

We consider the following nonsmooth/nonconvex optimization problem

$$\min_x f(x), \quad x \in S \subset \mathbb{R}^n, \tag{1}$$

where the objective function, $f : \mathbb{R}^n \to \mathbb{R}$, is nonsmooth and nonconvex. Problems like this, arise very often in Mechanics. Consider, for example, the inverse problem in flaw identification in elastomechanics. This problem is formulated as a minimization problem of a nonsmooth and in general nonconvex objective function [26]. Moreover, the solution of a Hemivariational inequality, has proved to be a substationary point of a nonsmooth/nonconvex functional [13]. Since the objective function is not differentiable, these problems cannot be addressed directly using deterministic optimization methods. The most common approach is the implementation of convex minimization algorithms on convex approximations of the objective function [17], [27]. Bundle methods are considered as the most promising approach for solving nonsmooth problems [13]. Their origin is the classical cutting plane method [3], [7], and they are based on a piecewise linear approximation of the objective function. Numerical experiments indicate that Bundle methods are effective [11], [14], [20]. However, some assumptions on the objective function are still needed. Specifically, $f(x)$ needs to be locally Lipschitz continuous [13].

Evolutionary and Swarm Intelligence algorithms are stochastic methods for global optimization. They draw from natural evolution and insects' social behavior, exploiting a population of

points to probe different areas of the search space simultaneously. They do not require derivatives information, but only function values. In EAs, each individual of the population is encoded either in real (Evolution Strategies [22], [23]) or binary (Genetic Algorithms [6], [15]) format, and operators inspired by natural evolution are applied on it. Thus, the population evolves through time towards the global minimum of the objective function. A fitness function is used for the evaluation of individuals. In unconstrained global optimization problems, the objective function under consideration is used as fitness function. EAs have been successfully applied on numerous, diverse, scientific fields, including Mathematics, Computer Science, Physics, Computational Biology, etc. PSO is a Swarm Intelligence algorithm, rooted in a simulation of social behavior [5], [10]. As in EAs, a population of potential solutions is used in PSO. However, each individual shares information with the rest, and can thus profit from the discoveries of all other companions.

The performance of PSO on widely used nonsmooth/nonconvex test problems is investigated in this paper. In the next section, the PSO algorithm is described. In Section 4, the test problems and the experimental results are reported. The paper closes with a discussion for further application of PSO on Mechanics problems.

3. THE PARTICLE SWARM OPTIMIZATION METHOD

PSO's predecessor was a simulator of social behavior, that was used to visualize the movement of a birds' flock. Several versions of the simulation model were developed, incorporating concepts such as nearest–neighbor velocity matching and acceleration by distance [5], [9]. When it was realized that the simulation could be used as an optimizer, several parameters were omitted, through a trial and error process, resulting in the first simple version of PSO [5].

In PSO, each individual of the population has an *adaptable velocity* (position change), which dictates its motion in the search space. Moreover, each individual is characterized by a *memory*, which stores the best position of the search space it has ever visited [5]. Thus, its movement is an aggregated acceleration towards its best previously visited position and towards the best individual of a topological neighborhood. Since the "acceleration" term was mainly used for particle systems in Particle Physics [19], the developers of this technique decided to use the term *particle* for each individual, and the name *swarm* for the population as a whole, thus, coming up with the name *Particle Swarm* for their algorithm [9].

Two variants of the PSO algorithm were developed. One with a global neighborhood, and one with a local neighborhood. According to the global variant, each particle moves towards its best previous position and towards the best particle in the whole swarm. On the other hand, in the local variant, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood [5]. Since the global variant performs better in most cases, it was selected for the experiments conducted in this paper.

Suppose that the search space is D–dimensional, then the i–th particle of the swarm can be represented by a D–dimensional vector, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^\top$. The *velocity* (position change) of this particle, can be represented by another D–dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^\top$. The best previously visited position of the i–th particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^\top$. Defining g as the index of the best particle in the swarm (i.e. the g–th particle is the best), and letting the superscripts denote the iteration number, then the swarm is manipulated according to the following two equations [5]:

$$v_{id}^{n+1} = \chi \left(w v_{id}^n + c_1 r_1^n (p_{id}^n - x_{id}^n) + c_2 r_2^n (p_{gd}^n - x_{id}^n) \right), \quad (2)$$

$$x_{id}^{n+1} = x_{id}^n + v_{id}^{n+1}, \quad (3)$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$, and N is the size of the swarm; w is called *inertia weight*; c_1, c_2 are two positive constants, called *cognitive* and *social* parameter respectively; χ is a *constriction factor*, which is used to limit velocity; r_1, r_2 are random numbers, uniformly distributed in $[0, 1]$; and $n = 1, 2, \ldots$, determines the iteration number. The role of these parameters is discussed in the next section.

In the early versions of PSO, neither an inertia weight nor a constriction factor was used. Thus, no actual mechanism controlled the velocity of a particle. This lack of a control mechanism resulted in low efficiency for PSO, compared to EAs [1]. This problem was initially addressed by using a maximum value V_{max} for the velocity. If the velocity exceeded this threshold, it was set equal to V_{max}. This parameter proved to be crucial, because large values could result in particles moving past good solutions, while small values could result in insufficient exploration of the search space. Specifically, PSO located the area of the optimum faster than EA techniques, but once in the region of the optimum, it was unable to adjust its velocity stepsize to continue the search at a finer grain. The aforementioned problem was addressed by incorporating either an inertia weight or a constriction factor (in some cases both parameters are included).

The role of the *inertia weight* w, in Eq. (2), is considered very important for PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter w regulates the trade–off between the global (wide–ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine–tuning the current search area. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. Initially, the inertia weight was set to a constant. However, experimental results indicated that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions [24], [25]. Thus, an initial value around 1.2 and a gradual decline towards 0 can be considered as a good choice for w .

The parameters c_1 and c_2, in Eq. (2), are not critical for PSO's convergence. However, proper fine–tuning may result in faster convergence and alleviation of local minima. An extended study of the acceleration parameter in the first version of PSO, is given in [8]. As default values, $c_1 = c_2 = 2$ were proposed. Experimental results indicate that $c_1 = c_2 = 0.5$ also consist a good choice. Recent work reports that it might be even better to choose a larger cognitive parameter, c_1, than a social parameter, c_2, but with $c_1 + c_2 \leqslant 4$ [2].

The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range $[0, 1]$. The constriction factor χ controls the magnitude of the velocities, in a manner similar to the V_{max} parameter, applied in the first versions of PSO. In general, the variants of PSO that incorporate a constriction factor are usually faster, but do not exhibit the same good global convergence rates, compared to the versions using an inertia weight.

The PSO method appears to adhere to the five basic principles of swarm intelligence, as defined in [5], [16]:

(a) *Proximity*, i.e. the swarm must be able to perform simple space and time computations;

(b) *Quality*, i.e. the swarm should be a able to respond to quality factors in the environment;

(c) *Diverse response*, i.e. the swarm should not commit its activities along excessively narrow channels;

(d) *Stability*, i.e. the swarm should not alter its behavior every time the environment changes; and finally

(e) *Adaptability*, i.e. the swarm must be able to change its behavior, when the computational cost is not prohibitive.

Indeed, the swarm in PSO performs space calculations for several time steps. It responds to the quality factors implied by each particle's best position and the best particle in the swarm, allocating the responses in a manner that ensures diversity. Moreover, the swarm alters its behavior (state) only when the best particle in the swarm (or in the neighborhood, in the local variant of PSO) changes, thus, it is both adaptive and stable [5].

In EAs, three main operators are involved. The *recombination*, the *mutation* and the *selection* operator. PSO does not have a direct recombination operator. However, the stochastic acceleration of a particle towards its previous best position, as well as towards the best particle of the swarm (or towards the best in its neighborhood in the local version), resembles the recombination procedure in Evolution Strategies [4], [18], [21], [23]. In PSO the information exchange takes place only among the particle's own experience and the experience of the best particle in the swarm, instead of being carried from fitness dependent selected "parents" to descendants as in EAs.

Moreover, PSO's directional position updating operation resembles the mutation in EAs, with a kind of memory built in. This mutation–like procedure is multidirectional both in PSO and EAs, and it includes control of the mutation's severity, utilizing factors such as the V_{max} and χ.

PSO is actually the only evolutionary algorithm that does not incorporate the "survival of the fittest" concept. It does not utilize a direct selection function. Thus, particles with lower fitness value can survive during the optimization and potentially visit any point of the search space [4].

4. EXPERIMENTAL RESULTS

The following well–known and widely used test problems of various dimensions were considered and they are defined as:

TEST PROBLEM 1 [12], [14]: Crescent function (2–dimensional)

$$F_1(x) = \max\{x_1^2 + (x_2 - 1)^2 + x_2 - 1, -x_1^2 - (x_2 - 1)^2 + x_2 + 1\},$$

with optimum value $F_1^* = 0$.

TEST PROBLEM 2 [12], [14]: CB2 function (2–dimensional)

$$F_2(x) = \max\{x_1^2 + x_2^4, (2 - x_1)^2 + (2 - x_2)^2, 2e^{-x_1+x_2}\},$$

with optimum value $F_2^* = 1.9522245$.

TEST PROBLEM 3 [12], [14]: Rosen–Suzuki function (4–dimensional)

$$\begin{aligned}
F_3(x) &= \max\{f_1(x), f_1(x) + 10f_2(x), f_1(x) + 10f_3(x), f_1(x) + 10f_4(x)\},\\
f_1(x) &= x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4,\\
f_2(x) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8,\\
f_3(x) &= x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10,\\
f_4(x) &= x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5,
\end{aligned}$$

with optimum value $F_3^* = -44$.

TEST PROBLEM 4 [12], [14]: Shor function (5–dimensional)

$$F_4(x) = \max_{1 \leqslant i \leqslant 10} \left\{ b_i \sum_{j=1}^{5} (x_j - \alpha_{ij})^2 \right\},$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 3 \\ 1 & 2 & 1 & 1 & 2 \\ 1 & 4 & 1 & 2 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 5 \\ 10 \\ 2 \\ 4 \\ 3 \\ 1.7 \\ 2.5 \\ 6 \\ 3.5 \end{pmatrix},$$

with optimum value $F_4^* = 22.600152$.

TEST PROBLEM 5 [12]: El–Attar function (6–dimensional)

$$\begin{aligned} F_5(x) &= \sum_{i=1}^{50} \left| x_1 e^{-x_2 t_i} \cos(x_3 t_i + x_4) + x_5 e^{-x_6 t_i} - y_i \right|, \\ y_i &= 0.5e^{-t_i} - e^{-2t_i} + 0.5e^{-3t_i} + 1.5e^{-1.5t_i} \sin 7t_i + e^{-2.5t_i} \sin 5t_i, \\ t_i &= 0.1(i-1), \quad 1 \leqslant i \leqslant 50, \end{aligned}$$

with optimum value $F_5^* = 0.5598131$.

TEST PROBLEM 6 [12]: Steiner 2 function (12–dimensional)

$$\begin{aligned} F_6(x) &= \sqrt{x_1^2 + x_{1+m}^2} + \sqrt{(\bar{\alpha}_{21} - x_m)^2 + (\bar{\alpha}_{22} - x_{2m})^2} + \\ &\quad + \sum_{j=1}^{m} p_j \sqrt{(\alpha_{j1} - x_j)^2 + (\alpha_{j2} - x_{j+m})^2} + \\ &\quad + \sum_{j=1}^{m-1} \tilde{p}_j \sqrt{(x_j - x_{j+1})^2 + (x_{j+m} - x_{j+m+1})^2}, \end{aligned}$$

$m = 6, \quad \bar{\alpha}_{21} = 5.5, \quad \bar{\alpha}_{22} = -1,$

$$A = [\alpha_{ij}] = \begin{pmatrix} 0.0 & 2.0 \\ 2.0 & 3.0 \\ 3.0 & -1.0 \\ 4.0 & -0.5 \\ 5.0 & 2.0 \\ 6.0 & 2.0 \end{pmatrix}, \quad p = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 5 \\ 1 \\ 1 \end{pmatrix}, \quad \tilde{p} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 2 \end{pmatrix},$$

with optimum value $F_6^* = 16.703838$.

TEST PROBLEM 7 [12], [14]: Maxq (20–dimensional)

$$F_7(x) = \max_{1 \leqslant i \leqslant 20} x_i^2,$$

with optimum value $F_7^* = 0$.

TEST PROBLEM 8 [12], [14]: Maxl (20–dimensional)

$$F_8(x) = \max_{1 \leqslant i \leqslant 20} |x_i|,$$

with optimum value $F_8^* = 0$.

TEST PROBLEM 9 [12], [14]: Goffin (30–dimensional)

$$F_9(x) = 30 \max_{1 \leqslant i \leqslant 30} x_i - \sum_{i=1}^{30} x_i,$$

with optimum value $F_9^* = 0$.

Two variants of PSO were used. One utilizes an inertia weight only, and the other utilizes only the constriction factor, denoted as Pso-In and Pso-Co respectively. For all experiments, the initial swarm and the initial velocities were randomly distributed within the range $[-10, 10]^D$, where D is the corresponding problem dimension. For the parameters of PSO, values which are considered as default, were used: $c_1 = c_2 = 2$; $\chi = 0.73$ in Pso-Co; w was gradually decreased from 1 toward 0.1, in Pso-In; the maximum allowed number of function evaluations was 10^5; $V_{max} = 4$; the size of the swarm was problem dependent. In any case, PSO was not allowed to exceed 10^5 function evaluations. A function tolerance of 10^{-4} was used as stopping criterion. For each test problem, 20 experiments were performed for both variants of PSO. The success rate, the mean, the standard deviation and the median of the required function evaluations were recorded and they are reported in Table 1.

Test Problem	Algorithm	Success Rate	Mean Func. Eval.	St.D. Func. Eval.	Median Func. Eval.
1	Pso-Co	100%	51.60	4.52	51
(2-dim.)	Pso-In	100%	793.20	59.32	780
2	Pso-Co	100%	141.00	71.56	123
(2-dim.)	Pso-In	100%	948.20	111.60	1007
3	Pso-Co	95%	377.00	560.37	79.5
(4-dim.)	Pso-In	95%	766.60	430.75	612
4	Pso-Co	95%	104.40	135.04	65.5
(5-dim.)	Pso-In	90%	281.10	137.12	232
5	Pso-Co	95%	224.75	148.95	192.5
(6-dim.)	Pso-In	74%	385.20	135.59	310
6	Pso-Co	100%	147.50	13.02	148.5
(12-dim.)	Pso-In	100%	443.75	15.45	439.5
7	Pso-Co	100%	255.20	32.65	246
(20-dim.)	Pso-In	100%	662.30	57.37	646
8	Pso-Co	100%	265.60	29.89	260.5
(20-dim.)	Pso-In	100%	702.15	75.65	686
9	Pso-Co	100%	499.35	30.86	501.5
(30-dim.)	Pso-In	100%	857.90	57.58	855

Table 1: The success rate, the mean, the standard deviation and the median of the required function evaluations for all experiments.

The experimental results indicate that PSO is effective in solving nonsmooth/nonconvex optimization problems. The success rates are high in all test problems, with the constriction factor variant always outperforming the inertia weight variant. The success rates are high even in high–dimensional test problems, although only default values for the PSO's parameters were used. Proper fine tuning may result in even faster convergence.

5. CONCLUSIONS

The performance of two variants of the PSO algorithm, in solving nonsmooth/nonconvex optimization problems, was investigated. Nonsmooth/nonconvex optimization problems are frequently encountered in Mechanics applications and they are usually addressed through deterministic techniques for convex optimization, applied on piecewise convex approximations of the objective function. However, this approach is indirect and the computational cost is heavy. Moreover, an assumption on the Lipschitz condition of the objective function needs to be made.

Evolutionary and Swarm Intelligence algorithms, can be applied on discontinuous test functions and disjoint search spaces. Thus, they appear to be a good alternative for solving nonsmooth/nonconvex problems. They require only function values, they are easily implemented, and they are noise tolerant. Since they are stochastic optimization algorithms, several parameters have to be defined. On the other hand, our experimental results on well–known and widely used problems indicate that high success rates can be achieved even by using the default parameters values of PSO. Of course, fine–tuning may result in faster convergence.

In future work, the performance of PSO directly on Mechanics applications will be investigated and compared with results reported in relative literature, obtained through different EAs.

References

[1] Angeline P.J., Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences, In: V.W. Porto, N. Saravanan, D. Waagen and A.E. Eiben (eds.) *Evolutionary Programming VII*, Springer, (1998), 601–610.

[2] Carlisle A. and Dozier G., An Off-The-Shelf PSO, In: *Proceedings of the Particle Swarm Optimization Workshop*, (2001), 1–6.

[3] Cheney E.W. and Golstein A.A., Newton's Method for Convex Programming and Tchebychef Approximation, *Numerische Mathematik*, 1, (1959), 253–268.

[4] Eberhart R.C. and Shi Y., Comparison Between Genetic Algorithms and Particle Swarm Optimization, In: V.W. Porto, N. Saravanan, D. Waagen and A.E. Eiben (eds.) *Evolutionary Programming VII*, Springer, (1998), 611–616.

[5] Eberhart R.C., Simpson P.K. and Dobbins R.W., *Computational Intelligence PC Tools*, Academic Press Professional, Boston (1996).

[6] Holland J.H., *Adaptation in Natural and Artificial Systems*, MIT Press (1992).

[7] Kelley J.E., The Cutting Plane Method for Solving Convex Programs, *SIAM J.*, 8, (1960), 703–712.

[8] Kennedy J., The Behavior of Particles, In: V.W. Porto, N. Saravanan, D. Waagen and A.E. Eiben (eds.) *Evolutionary Programming VII*, Springer, (1998), 581–590.

[9] Kennedy J. and Eberhart R.C., Particle Swarm Optimization, In: *Proc. IEEE Int. Conf. Neural Networks*, IEEE Service Center, Piscataway, NJ, (1995), 1942–1948.

[10] Kennedy J. and Eberhart R.C., *Swarm Intelligence*, Morgan Kaufmann (2001).

[11] Kiwiel K.C., Proximity Control in Bundle Methods for Convex Nondifferentiable Optimization, *Mathematical Programming*, 46, (1990), 105–122.

[12] Lukšan L. and Vlček J., Test Problems for Nonsmooth Unconstrained and Linearly Constrained Optimization, Technical Report No. 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, (2000).

[13] Mäkelä M.M., Miettinen M., Lukšan L. and Vlček J., Comparing Nonsmooth Nonconvex Bundle Methods in Solving Hemivariational Inequalities, Technical Report 10/1997, University of Jyväsklylä, (1997).

[14] Mäkelä M.M. and Neittaanmäki, *Nonsmooth Optimization. Analysis and Algorithms with Applications to Optimal Control*, World Scientific, Singapore (1992).

[15] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer–Verlag, New York (1992).

[16] Millonas M.M., Swarms, Phase Transitions, and Collective Intelligence, In: M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel and T. Fukuda (eds.) *Computational Intelligence: A Dynamic System Perspective*, Piscataway, NJ: IEEE Press, (1994), 137–151.

[17] Mistakidis E.S., Baniotopoulos C.C. and Panagiotopoulos P.D., On the Numerical Treatment of the Delamination Problem in Laminated Composites under Cleavage Loading, *Composite Structures*, 30, (1995), 453–466.

[18] Rechenberg I., Evolution Strategy, In: J.M. Zurada, R.J. Marks II and C. Robinson (eds.) *Computational Intelligence: Imitating Life*, Piscataway, NJ: IEEE Press (1994).

[19] Reeves W.T., Particle Systems–A Technique for Modelling a Class of Fuzzy Objects, *ACM Transactions on Graphics*, 2(2), (1983), 91–108.

[20] Schramm H. and Zowe J., A Version of the Bundle Idea for Minimizing Nonsmooth Functions: Conceptual Idea, Convergence Analysis, Numerical Results, *SIAM J. Optimization*, 2, (1992), 121–152.

[21] Schwefel H.–P., *Evolutionsstrategie und numerische Optimierung*, Technical University of Berlin, Department of Process Engineering, Dr.–Ing. Thesis (1975).

[22] Schwefel H.–P., *Numerical Optimization of Computer Models*, Wiley (1981).

[23] Schwefel H.–P., *Evolution and Optimum Seeking*, Wiley (1995).

[24] Shi Y. and Eberhart R.C., Parameter Selection in Particle Swarm Optimization, In: V.W. Porto, N. Saravanan, D. Waagen and A.E. Eiben (eds.) *Evolutionary Programming VII*, Springer, (1998), 611–616.

[25] Shi Y. and Eberhart R.C., A Modified Particle Swarm Optimizer, In: *Proc. IEEE Conf. Evolutionary Computation*, (1998).

[26] Stavroulakis G.E. and Antes H., Flaw Identification in Elastomechanics: BEM Simulation with Local and Genetic Optimization, *Structural Optimization*, 16(2/3), (1998), 162–175.

[27] Tzaferopoulos M.A., Mistakidis E.S., Bisbos C.D. and Panagiotopoulos P.D., Comparison of Two Methods for the Solution of a Class of Nonconvex Energy Problems Using Convex Minimization Algorithms, *Computers & Structures*, 57(6), (1995), 959–971.