Reprojective Particle Swarm Optimization

Abstract: Particle Swarm Optimization is a stochastic metaheuristic optimization algorithm, used in various applications such as civil, electrical, and industrial engineering. It's most impressive property is the ability to efficiently search through unknown spaces that may be either non-differentiable,or making the estimation of gradient technically unfeasible. In this paper, we are going to propose a new optimization algorithm, inspired by PSO, with the superior ability to learn and apply the geometry of it's environment, enhancing the group thinking ability by incorporating memory into the hill-climbing mechanics. The simulation results show that our algorithm greatly outperforms all the current versions of PSO,setting new benchmarks.

Keywords: Particle Swarm Optimization, Search, Stochastic Algorithm

Introduction

Optimization problems are fairly common in most complex human endeavors, such as engineering[2][6], logistics[5], or artificial intelligence[8]. Currently, there are lots of methods available for finding either optimal or suboptimal solutions for them, such as nature-based and gradient-based methods. Although gradient-based methods tend to have faster convergence to a solution, requirement of Jacobian, and sometimes even Hessian matrix, becomes a strong disadvantage in real-life applications due to highly complicated objective functions, usually making computation of the derivatives unfeasible, or even physically impossible. Thus, although perceived as slower, due to their ability to optimize non-convex and/or nondifferentiable functions, so-called ,,nature metaphors'' are still popular and actively researched. One of the most popular is PSO, well-known between other nature-based algorithms[4] for its ability of fast convergence and high performance on continuous search spaces.

Preliminaries on PSO

Particle swarm optimization[1] is an iterative stochastic metaheuristic algorithm, based on collective behavior of flocks of birds in search for food. It is applied to many tasks mostly to minimization of non-differentiable functions.

As a description of a swarm-it consists of L particles, each particle possessing coordinates in d dimensional search space and velocity in the same space. The current position of a i-th particle is a vector $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$, while the velocity is presented in $v_i = (v_{i1}, v_{i2}, \dots, v_{id})$. Every iteration, velocity and position vectors are updated as follows

$$(1) \qquad x_i(t+1) = x_i(t) + v_i(t+1)$$
$$(2) \qquad v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot \big(p_i(t) - x_i(t)\big) + c_2 \cdot r_2 \cdot \big(g_i(t) - x_i(t)\big)$$

Where $p_i = (p_{i1}, p_{i2}, \dots, p_{id})$ is personal best position of a particle throughout the iterations, $g_i = (g_{i1}, g_{i2}, \dots, g_{id})$ is the position with the best fitness value in „history'' of the swarm, to which we will later refer as global best. w is the inertia weight, which can also be perceived as personal momentum of a particle, as it depends on velocities from its previous iterations, it is chosen in order to balance the exploration properties of the swarm with the ability to search local regions more precisely. c_1 and c_2 are respectively individual and social factors, while r_1 and r_2 are random independent numbers, generated each iteration from the uniform distribution on $[0,1]$

Throughout iterations, each particle updates itself according to received information, until the termination criteria have been reached.

Proposed approach

The inspired by PSO algorithm we are going to propose consists of 3 main improvements: incorporation of the preserved history into the swarm intelligence,constant reprojection across the best neighbors,and natural selection component

1. Preserved history
   To incorporate the information from the previous iterations in a more effective way, we store the information about the best fitness values in a set of size N_p, which is a custom parameter.Each of the objects which it contains possess 3 attributes-the coordinates, the fitness value, and the ,,age''(the iteration when the point was appended to the set).
2. Reprojection
   The proposed approach involves two kinds or reprojections – involving individual and collective thinking components
   2.1 Individual
      Here are the equations (1,2), updated according to our approach

(3) $$x_i(t+1) = x_i(t) + v_i(t+1) + 2 * \left(l_i(t) - x_i(t)\right)$$

(4) $$v_i(t+1) = -v_i(t) - 2 \cdot r_1 \cdot \left(p_i(t) - x_i(t)\right) - 2 \cdot r_2 \cdot \left(g_i(t) - x_i(t)\right)$$

Firstly- the reprojection across local best of the particle is added
Secondly- custom coefficients are currently more of the structure element.As of now the main development and movement of the whole swarm is more dependent on the other components of our algorithm, the purpose of the individual movement apart from reprojection is ensuring of convergence. The choice of -2 as coefficients for social and cognitive factor is based on the fact that the easiest solution would be simply making the movement negative without any randomness. However, we insist on remaining some stochastic properties to ensure the certain level of diversity in particles behaviour

2.2 Collective

The component of reprojection,based on collective thinking,consists of generating positively and inversely weighted centers from preserved history, using historical information to reproject them across each other, thus searching through the space more efficiently

Firstly, here is the general equation of the weighted center.

(5) $$\mu_w = \frac{\sum_{i=1}^{N_p} w_i * X_i}{\sum_{i=1}^{N_p} w_i}$$

where w_i is the weight of i-th particle, X_i is a coordinates vector of the particle, and N_p is the number of particles in preserved history. Here we see the equation of the weight formed by j-th function for an i-th preserved particle

(6) $$w_{i,j} = \frac{f_j(t_i)}{a_i^{\alpha}}$$

(7) $$t_i = \frac{(c_i - \mu_c)}{\sigma_c}$$

Where c_i is the fitness of i-th particle, μ_c and σ_c are the mean and standard deviation of fitness distribution in preserved history, t_i is a temporary value of i-th weight before the final transformation according to the equation, and a_i^{α} is an age of i-th particle raised to the power α, which is a custom parameter with restriction to be positive-with the increase in it more swarm becomes more sensitive to new particles with high fitness, and as it decreases, the behaviour changes to focusing solely on the fitness values, instead of age.

$f_j(x)$ is one of the functions from the set of choice, corresponding to each mean.The decision on the functions of choice remains purely architectural, although being restricted to the space of functions with growing derivatives. In minimization cases, such as when we are interested in minimizing the error of the model - they are supposed to be raised to the power of -1.

Then,we generate $2 * f_n$ new particles, where f_n is the custom parameter,meaning the number of different weight functions considered. For each function there are 2 particles generated, first one is the weighted average itself,and the second is the one reprojected across it from the common point μ_b-the,,bad mean''

$$(8) \qquad \mu_b = \frac{\sum_{i=1}^{N_p} t_i * X_i}{\sum_{i=1}^{N_p} t_i}$$

Here is the equation for the bad mean, which is essentially the formula for a weighted average from equation, missing the last step of forming the weights via the function. Here we present the minimization formula, in case of maximization problems, t_i should be inversed.

Then, the second particle in pair, is generated by reprojection of ,,bad mean'' across the corresponding weighted average, according to the equation

$$(9) \qquad r = \mu_w - \mu_b$$

$$(10) \qquad x_r = \mu_b + r * (1 + \frac{\sigma}{\|r\|_n})$$

Where μ_b is ,,bad average'', r is the D-dimensional coordinates vector,containing the difference in position between the corresponding weighted average and the mean, $\|r\|_2$ is equivalent to Euclidean distance between them, x_r is a position vector of a new particle, and σ is a span measure for the search space, considered at the moment.

$$(11) \qquad \sigma = [\sigma_1, \sigma_2, \ldots \sigma_{D-1}, \sigma_D]_{max}$$

In equation we can see that it is a maximum entry of a D-dimensional vector,where d-th element is a standard deviation of d-th dimension of coordinates in preserved history.

3. Natural selection
   After each iteration, $2 * f_n$ particles with the worst fitness value are removed from the swarm – according to the number of new particles generated by collective reprojection,thus leaving the size of the swarm the same

Here are the steps of R-PSO

1) Start
2) Initiate the swarm
3) Initiate the positions of the particles
4) Check the termination condition
5) Update the swarm
   1) Update the fitness values
   2) Update the preserved history
   3) Generate the new particles based on preserved history and update their costs
   4) Remove the worst members

5) Update the personal and social information(personal,global,and local best)
6) Update the velocity and coordinates
7) Check the termination condition, if not met, repeat step 5

Algorithm Comparison

As our approach is inspired by the PSO, we are going to compare it’s performance to other versions on a set of benchmark functions. The functions of choice are DeJong’s f4, Cigar, Sphere, Griewank, Ackley, and Rastigrin. In the Table 1 and Table 2 we present the expressions of testing functions and the bounds of the search space for swarm initialization.

Name of the function	Expression
Sphere	$f_1 = \sum_{i=1}^{M} {x_i}^2$
Griewank	$f_2 = \frac{1}{4000}\sum_{i=1}^{M} {x_i}^2 - \sum_{i=1}^{M} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley	$f_3 = 20 + Eu - 20exp(-0.2\sqrt{\frac{1}{M}\sum_{i=1}^{M} {x_i}^2}$)-$exp(\frac{1}{M}\sum_{i=1}^{M}\cos(2\pi x_i)$)
Rastrigrin	$f_4 = \sum_{i=1}^{M} {x_i}^2$-10 cos ($2\pi x_i$) + 10
Cigar	$f_5 = np.abs(x_1) + 10^4 \sum_{i=1}^{M} {x_i}^2$
De Jong’s f4	$f_6 = \sum_{i=1}^{M} i x_i^4$

Table 1: Expressions of testing functions

Function	Initialization variable bounds	Optimum
f1	[−5, 5]M	0
f2	[−500, 500]M	0
f3	[−20, 20]M	0
f4	[−6, 6]M	0
f5	[−6, 6]M	0
f6	[−100, 100]M	0

Table 2: Bounds and optimal solutions of manipulating variables

We are going to use as a reference point the other versions of PSO,such as baseline PSO,HKP-PSO[17],P-PSO, K-PSO, PSOvm[13], PSO-LDIW[12] and W-PSO. Our choice was based mainly on the diversity between the testing algorithms, involving those with higher or lower computational cost, different ways of interaction within swarm involved, different methods for cognitive factor application, like in the case of K-PSO and P-PSO, based on Bayesian filtering, and the ways to control the velocity of individual particles, mostly involving inertia manipulations[12].

In our case, we use two function groups-polynomial and exponential $(e^x)^{-1}$, $(x^3)^{-1}$ – as a basic example of the exponential function, and the cubic polynomial, to emphasize the importance of the power being odd in case of polynomial space. These are the general functions, which were chosen to show the general performance, without tuning the architecture to suit the certain purpose better.

Raising them to the power of -1 directly corresponds to the fact that we will be solving minimization problems

The age is going to have the power of 1 to preserve generality, the set of neighbors will be set to 7, swarm size to 30, dimensionality to 20, and maximum iteration to 6000,making a 100 runs for more thorough analysis.

The data on the performance of other algorithms as a reference, was taken from the article presenting the HKP-PSO, and all the conditions were preserved.

Function	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
F1	14	5316	5896	5117	2566	81	5963	5832
F2	9	4970	5925	4595	2292	79	5908	5695
F3	87	5515	5887	4669	2325	40	5936	4580
F4	9	4999	5995	5023	2396	57	5936	5507
F5	1701	5359	5959	5059	640	92	5967	5651
F6	16	4946	5978	5024	1751	87	5947	5407

Table 3: Convergence rate of comparison algorithms

Function	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	PSO-W
F1	1.87	3.05	3.22	0.96	0.90	1.02	1.11	1.17
F2	1.40	3.21	3.49	1.23	1.13	1.30	1.43	2.18
F3	2.22	2.23	3.82	1.23	1.08	1.21	1.33	1.15
F4	1.87	2.30	3.15	1.10	0.98	1.24	1.36	1.44
F5	2.12	2.54	2.80	1.51	1.47	1.48	1.6	1.43
F6	1.38	1.34	3.04	0.93	0.80	0.98	1.07	0.68

Table 4: Time complexity of comparison algorithms

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
Best	0.0	$2.7*10^{-10}$	$1.8*10^{-7}$	$3.4*10^{-9}$	$1.8*10^{-5}$	$7.2* 10^{-1}$	$1.0 *10^{-4}$	$1.6 * 10^{-2}$
Worst	0.0	$3.6 * 10^{-4}$	$2.0* 10^{-3}$	$3.9*10^{-3}$	$2.5*10^{1}$	$1.3* 10^{1}$	$2.8*10^{0}$	$2.4* 10^{2}$
Average	0.0	$1,9*10^{-5}$	$1,4*10^{-4}$	$5.6*10^{-5}$	$2.1*10^{0}$	$4.1*10^{0}$	$1.0 *10^{-1}$	$2.7* 10^{1}$

Table 5 Final fitness of f1

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
Best	0.0	$2.9*10^{-8}$	$4.7*10^{-6}$	$5.8*10^{-6}$	$4.7 * 10^{-2}$	$2.7 * 10^{0}$	$9.0 * 10^{0}$	$1.6 * 10^{0}$
Worst	0.0	$1.3*10^{-1}$	$5.0 * 10^{-1}$	$1.3*10^{1}$	$1.3*10^{2}$	$4.6 * 10^{1}$	$2.6 * 10^{1}$	$9.5 * 10^{0}$
Average	0.0	$3.8*10^{-2}$	$6.5*10^{-2}$	$2.5*10^{-1}$	$4.9*10^{0}$	$1.2 * 10^{0}$	$1.6 * 10^{1}$	$4.6 * 10^{0}$

Table 6 Final fitness of f2

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
Best	0.0	$9.5*10^{-6}$	$4.0*10^{-3}$	$2.3*10^{-5}$	$3.9*10^{-2}$	$4.7* 10^{0}$	$7.0 * 10^{0}$	$9.6 * 10^{0}$
Worst	0.0	$1.6*10^{-2}$	$2.7 * 10^{-2}$	$4.0*10^{0}$	$1.5*10^{1}$	$1.3* 10^{1}$	$8.2* 10^{0}$	$2.0 * 10^{1}$
Average	0.0	$1.2*10^{-3}$	$8,4*10^{-3}$	$1.0*10^{-1}$	$3.3*10^{0}$	$8.2* 10^{0}$	$7.6 * 10^{0}$	$2.0 * 10^{1}$

Table 7 Final fitness of f3

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
Best	0.0	$1.9*10^{-7}$	$2.0*10^{-4}$	$2.0*10^{0}$	$7.3*10^{0}$	$4.2 * 10^{1}$	$2.3 * 10^{1}$	$1.7* 10^{2}$
Worst	0.0	$1.7*10^{1}$	$1.0 * 10^{1}$	$8.9*10^{1}$	$1.6*10^{2}$	$1.6* 10^{2}$	$3.0 * 10^{2}$	$8.3* 10^{3}$
Average	0.0	$4.5*10^{0}$	$8,6*10^{-1}$	$2.5*10^{1}$	$4.7*10^{1}$	$1.0 * 10^{2}$	$1.6* 10^{2}$	$1.4* 10^{3}$

Table 8 Final fitness of f4

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSOvm	PSO-LDIW	W-PSO
Best	0.0	$7.1*10^{-3}$	$2.1*10^{0}$	$1.8*10^{-2}$	$1.7*10^{2}$	$2.6 * 10^{6}$	$3.1 * 10^{4}$	$1.1 * 10^{4}$
Worst	0.0	$1.0*10^{4}$	$5.5 * 10^{3}$	$6.0*10^{4}$	$1.0*10^{8}$	$9.4 * 10^{7}$	$1.3 * 10^{6}$	$1.2 * 10^{6}$

Average	0.0	$2.6*10^{3}$	$5.1*10^{2}$	$6.0*10^{3}$	$5.0*10^{6}$	$1.5 * 10^{7}$	$3.1 * 10^{5}$	$2.8 * 10^{5}$

Table 9 Final fitness of f5

Indicator	R-PSO	HKP-PSO	P-PSO	K-PSO	PSO	PSO-VM	PSO-LDIW	W-PSO
Best	0.0	$1.6*10^{-15}$	$1.2*10^{-12}$	$4.9*10^{-14}$	$3.5*10^{-9}$	$7.4* 10^{-2}$	$4.0* 10^{-2}$	$2.9* 10^{1}$
Worst	0.0	$9.3*10^{-8}$	$9.3 * 10^{-8}$	$4.3*10^{-7}$	$4.7*10^{2}$	$6.7* 10^{1}$	$9.9* 10^{3}$	$4.8* 10^{4}$
Average	0.0	$4.5*10^{-6}$	$4.5*10^{-6}$	$6.8*10^{-6}$	$1.3*10^{4}$	$1.4* 10^{3}$	$1.0* 10^{3}$	$4.8* 10^{3}$

Table 10 Final fitness of f6

As we can see, our algorithm drastically outperforms every other version, sometimes producing the global minima of the function in very few iterations. It is the first both to converge in every test case only losing in speed to PSOvm in some cases, and has the best final fitness values achieved, while also being computationally cheaper that most of the other versions-taking into account it's speed of convergence. There is not a single case when it has been outperformed. It's worst performance overall dominates the best result of any other algorithms in comparison

Conclusions

In this paper, a new, PSO-inspired metaheuristic optimization algorithm is proposed. It incorporates the memory into it's learning process, applying both positive and negative experiences of the swarm. Also, it proposes a new movement method, that makes use of both local and global social behaviour, processing the search space on the level, unattainable to current versions of PSO, setting a new benchmark for non-differential optimization algorithms.

References

[1] KENNEDY J, EBERHART R. Particle Swarm Optimization [C]//IEEE International Conference on Neu-

ral Networks. Perth, Australia: IEEE, 1995: 1942-1948.

[2] Sarvesh P.S. Rajput, Suprabeet Datta, A review on optimization techniques used in civil engineering material and structure design, Materials Today: Proceedings, Volume 26, Part 2, 2020, 1482-1491,

[3] CHAUHAN P, DEEP K, PANT M. Novel inertia weight strategies for particle swarm optimization [J], Memetic Computing, 2013, 5(3): 229-251.

[4] Nature-inspired optimization algorithms and their significance in multi-thresholding image segmentation: an inclusive review, Springer Nature 2022, 899-945

[5] Zhou, Y., Xia, W., & Dai, J. (2023). The application of nature-inspired optimization algorithms on the modern management: A systematic literature review and bibliometric analysis. Journal of Management and Organisation, 29(4),655-678.

[6] Mounica Nutakki, Srihari Mandava, Review on optimization techniques and role of Artificial Intelligence in home energy management systems, Engineering Applications of Artificial Intelligence,Volume 119,2023

[7] Cuevas, E., Avalos, O., Gálvez, J., IIR System Identification Using Several Optimization Techniques: A Review Analysis. In: Analysis and Comparison of Metaheuristics. Studies in Computational Intelligence, vol 1063. Springer 2023

[8] Abdulkadirov R., Lyakhov P., Nagornov N., Survey of Optimization Algorithms in Modern Neural Networks. *Mathematics* **2023**,

[9] DOUCET A, FREITAS N D, GORDON N. Sequential Monte Carlo methods in practice [M]. New York, USA: Springer-Verlag, 2001: 106.

[12] J. Xin, G. Chen and Y. Hai, "A Particle Swarm Optimizer with Multi-stage Linearly-Decreasing Inertia Weight," *2009 International Joint Conference on Computational Sciences and Optimization*, , 2009, pp. 505-508

[13] ZAHARIS Z D, GRAVAS I P, YIOULTSIS T V, etal. Exponential log-periodic antenna design using im-proved particle swarm optimization with velocity mutation [J]. IEEE Transactions on Magnetics, 2017, 53(6): 7204104.

[14] YAN X H, HE F Z, HOU N, et al. An efficient particle swarm optimization for large-scale hardware/software co-design system [J]. International Journal of Cooperative Information Systems, 2018, 27(1): 1741001.

[15] LI C H, YANG S X, NGUYEN T T. A self-learning particle swarm optimizer for global optimization problems [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012, 42(3): 627-646.

[16] PENG Pai, CHEN Cong, YANG Yongsheng Particle Swarm Optimization Based on Hybrid Kalman Filter and Particle Filter J. Shanghai Jiao Tong Univ. (Sci.), 2020, 25(6): 681-688