Comparing Border Strategies for Roaming Particles on Single and Multi-swarm PSO

Tomas Kadavy(✉), Michal Pluhacek, Adam Viktorin, and Roman Senkerik

Faculty of Applied Informatics, Tomas Bata University in Zlin, T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{kadavy, pluhacek, aviktorin, Senkerik}@fai.utb.cz

Abstract. In this paper, the methods for handling particles that violate available search spaces are compared using a single and multi-swarm technique. The methods are soft borders and hypersphere universe. The goal is to compare this approaches and its combination. The comparisons are made on CEC'17 benchmark set functions. The experiments were carried out according to CEC benchmark rules and statistically evaluated.

Keywords: Particle swarm optimization · PSO · Search space boundaries · Multi-swarm · Roaming particles

1 Introduction

Despite the fact that Particle Swarm Optimization (PSO) was first proposed in 1995 [1], its well-known weaknesses are still actual and many researches are working on improving that. Classical PSO have numerous adjustment options. These can be: learning factors, inertia weight, maximum velocity and others. In this paper, the strategies for handling roaming particles are compared with their impact on solutions quality. The boundaries of the search space are defined by particular optimization problem and define the minimally acceptable and maximal acceptable value of the solutions in dimensions. The compared strategies in this paper are: soft borders and hypersphere universe. For the hyperspace method, the velocity formula can by changed to match a unique characteristic of this method. This velocity update is categorized as a different strategy among to hypersphere universe using only classical velocity formula. The strategies in this paper are compared on CEC'17 benchmark set [2]. The strategies are also combined together using multi-swarm technique [3].

The paper is structured as follows. The PSO and its multi-swarm modification are described in Sect. 2. The methods for handling the roaming particles are described in Sect. 3. The experiment setup is detailed in Sect. 4. Section 5 contains statistical overviews of results and performance comparisons obtained during the evaluation on benchmark set. Discussion and conclusion follow.

2 Particle Swarm Optimization

A typical representative of swarm intelligence algorithms is Particle Swarm Optimization (PSO). This algorithm was published in 1995 by Ebenhart and Kennedy in [1]. This algorithm mimics the social behavior and movement of swarm members. Originally it was bird flocking and fish schooling. Because it is a quite long time from his first appearance, the numerous versions appeared. These new versions mostly want to improve the fact that PSO has well know weakness, the premature convergence to local optima.

In this PSO algorithm, the individuals (also called particles) are moving in space of possible solutions of the defined particular problem. This movement is determined by two factors. One of them is the current position of a particle, labelled as x. The second one is the velocity of a particle, labelled as v. Each particle also remembers his best position (solution of the problem) obtained so far. This solution is tagged as the *pBest*, personal best solution. Also, each particle has access to the global best solution, *gBest*, which is selected from all *pBests*. These variables set the direction for every particle and then even a new position in next iteration. PSO usually stops after a number of iterations, or a number of FEs, fitness evaluations, defined by the user.

The particle position is represented as coordinates in n-dimensional space of solutions. These coordinates are parameters of the optimized problem. In every step of the algorithm, the new positions of particles are calculated based on previous positions and velocities. The new position of a particle is checked if it still lies in space of possible solutions. The function of the optimized problem is called Cost Function (CF). The position of a particle is used as input parameters in CF. If the value of CF is better than the value saved in *pBest* of a particle, then the particle saves this new position as his new *pBest*. Also, if this *pBest* has better CF value than the previous *gBest*, then this *pBest* is saved as new *gBest*.

The position of particle x is calculated according the formula (1)

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{1}$$

where $t + 1$ is actual iteration, t is then previous iteration, x_{ij} is the position of a i-particle in j-dimension, v is the velocity of a particle.

The velocity of a particle v is calculated according to (2).

$$v_{ij}^{t+1} = w \cdot v_{ij}^{t} + c_1 \cdot r_1 \cdot \left(pBest_{ij} - x_{ij}^{t}\right) + c_2 \cdot r_2 \cdot \left(gBest_{j} - x_{ij}^{t}\right) \tag{2}$$

where w is inertia weight [4], c_1 and c_2 are learning factors and r_1 and r_2 are random numbers of unimodal distribution in range <0,1>.

2.1 Multi-swarm

Due to premature convergence of classical PSO, some techniques were developed. One of them is called multi-swarm [3]. The basic idea of this mechanism is splitting the swarm population into, typically two, subpopulations. Each group has own *gBest*. These subgroups exchange their *gBest* (*gBest* of better fitness value is copied to others subpopulations) every n-iterations, defined by the user. Each subpopulation can use different behaviors.

3 Strategies

When the position of a particle is updated, the particle has to be checked if its new position is in the appropriate boundaries (inside a space of possible solutions). If the particle is not in the appropriate boundaries (roaming particle), some corrections have to be made. There are several different strategies for this purpose. For this paper two of them are selected: soft borders and hypersphere universe.

3.1 Soft Borders

The particle can travel outside of the search space, there is only one restriction applied on that particle. His cost function is not calculated and therefore his *pBest* is unchanged. If a certain condition is met [5], the particle will eventually return inside a search space by itself.

3.2 Hypersphere Universe

This method simulates an endless spherical universe. For example, if a particle violates upper boundary of the search space, the particle than appear in the search space from lower boundary. The upper boundary is neighbouring the lower one of corresponding dimension and vice versa. This approach is explained in Fig. 1.

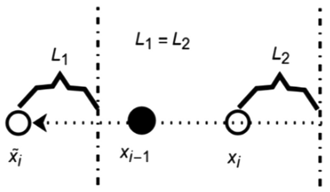

Fig. 1. Explanation of hypersphere universe method. x_{i-1} is the particle position in last iteration, $\overline{x_i}$ is uncorrected position and x_i is the final correct position.

Using this method, a typical way to compute velocity (2) of a particle becomes less efficient, because a particle can choose a longer way (vector of particle position and his *pBest* or *gBest*). The hypersphere universe offers the second option, a particle can travel through a boundary and reach the final destination using shorter vector. In Fig. 2 the *L1* is vector computed using the standard velocity update (2) and vector *L2* is a new vector that appears when the hyperspace method is used. The sum of these vectors is the range of available search space.

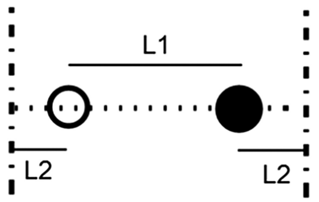

Fig. 2. Two possible velocity vectors in hypersphere universe.

The new velocity formula for this method, which can choose the better (smaller) vector, is defined as (3).

$$v_{ij}^{t+1} = w \cdot v_{ij}^{t} + c_1 \cdot r_1 \cdot L_{P,ij}^{t} + c_2 \cdot r_2 \cdot L_{G,ij}^{t} \tag{3}$$

where $L_{P,ij}^{t}$ and $L_{G,ij}^{t}$ are defined in formula (4).

$$\begin{gathered} L_{P,ij} = \begin{cases} \widehat{L}_{P,ij}, & if \left|\widehat{L}_{P,ij}\right| \leq d \\ \widehat{L}_{P,ij}\, mod(-d), & if\left(\left|\widehat{L}_{P,ij}\right| > d \wedge \left|\widehat{L}_{P,ij}\right| > 0\right) \\ \widehat{L}_{P,ij}\, mod(+d), & if\left(\left|\widehat{L}_{P,ij}\right| > d \wedge \left|\widehat{L}_{P,ij}\right| \leq 0\right) \end{cases} \\ L_{G,ij} = \begin{cases} \widehat{L}_{G,ij}, & if \left|\widehat{L}_{G,ij}\right| \leq d \\ \widehat{L}_{G,ij}\, mod(-d), & if\left(\left|\widehat{L}_{G,ij}\right| > d \wedge \left|\widehat{L}_{G,ij}\right| > 0\right) \\ \widehat{L}_{G,ij}\, mod(+d), & if\left(\left|\widehat{L}_{G,ij}\right| > d \wedge \left|\widehat{L}_{G,ij}\right| \leq 0\right) \end{cases} \end{gathered} \tag{4}$$

The d is computed by formula (5) where b^u and b^l stand for upper bound limit and lower bound limit of the search space. The $\widehat{L}_{P,ij}$ and $\widehat{L}_{G,ij}$ are defined in (6).

$$d = \frac{\left|b^u - b^l\right|}{2} \tag{5}$$

$$\begin{gathered} \widehat{L}_{P,ij} = pBest_{ij} - x_{ij} \\ \widehat{L}_{G,ij} = gBest_{j} - x_{ij} \end{gathered} \tag{6}$$

4 Experimental Setup

The experiments were performed for dimension dim = 10 on CEC'17 benchmark functions set [2]. The maximal number of cost function evaluations is set to 10000 · dim according to the definition for this benchmark set. The population size (NP) is set to 40 for all dimensions. The inertia weight is set w = 0.729 and learning factors are

$c_1 = c_2 = 1.49445$ according to [5]. Every test function is repeated for 51 independent runs and the results are statistically evaluated. The benchmark set includes 30 functions separated into four categories: unimodal, multimodal, hybrid and composite. Each function has search space defined in $[-100{,}100]^{Dim}$ and global minimum is $100 \cdot f_i$, where i is an order of test function f. In Table 1 is shown the summary of tested functions.

Table 1. Tested functions

Test function f_i	Category	Global minimum
f_1	Unimodal	100
f_2		200
f_3		300
f_4	Multimodal	400
f_5		500
f_6		600
f_7		700
f_8		800
f_9		900
f_{10}		1000
f_{11}	Hybrid	1100
f_{12}		1200
f_{13}		1300
f_{14}		1400
f_{15}		1500
f_{16}		1600
f_{17}		1700
f_{18}		1800
f_{19}		1900
f_{20}		2000
f_{21}	Composition	2100
f_{22}		2200
f_{23}		2300
f_{24}		2400
f_{25}		2500
f_{26}		2600
f_{27}		2700
f_{28}		2800
f_{29}		2900
f_{30}		3000

The six variants were tested. Three of them are multi-swarms where each of them is composed of two subpopulations of 20 particles and every 100 iterations the subpopulations compare their *gBest* and both of them use the better one (*gBest* with

Table 2. Variant description

Variant	Type	Border strategy	
A1	Single-swarm	Soft border	
A2	Single-swarm	Hypersphere	
A3	Single-swarm	Hypersphere with velocity formula (3)	
A4	Multi-swarm	Soft border	Hypersphere
A5	Multi-swarm	Soft border	Hypersphere with velocity formula (3)
A6	Multi-swarm	Hypersphere	Hypersphere with velocity formula (3)

smaller fitness value). Each variant uses different border strategy for its subpopulations. In Table 2 is the detailed list of this setting.

5 Results

The Friedman test [6] was used for statistical comparison of used variants. To compute the statistics, the JAVA package from 'http://sci2s.ugr.es/keel/multipleTest.zip' was used. The results of each test function from CEC benchmark set were averaged from their 51 independent runs. These average results were used for the Friedman test.

The p-value computed by Friedman test is 4.13E−8, the critical value of the Friedman statistic is at $\alpha = 0.05$ [7] so the ranking of Friedman statistics is valid.

The non-parametric Friedman test ranking of the variants is in Table 3. The adjusted Holm p-values among A6 variant and others are shown in Table 4.

Table 3. Friedman ranking of variants

Variant	Ranking
A6	2.28
A5	2.45
A1	3.48
A3	3.62
A4	4.43
A2	4.73

Table 4. Adjusted p-values of A6 variant and others variants

Variant	p-value
A5	0.73E0
A1	0.03E0
A3	0.02E0
A4	3.42E−5
A2	1.97E−6

Furthermore, selected examples of mean *gBest* value history are shown in Figs. 3, 4, 5 and 6.

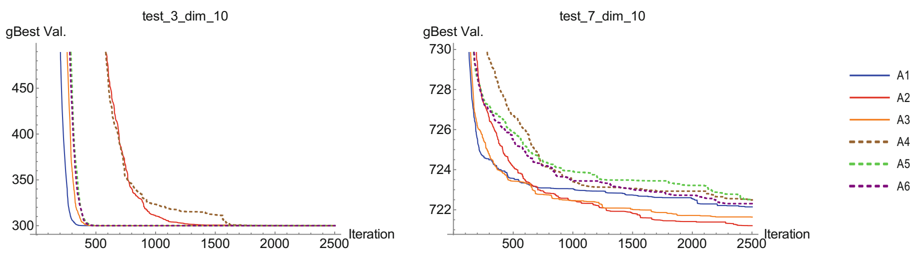

Fig. 3. Comparisons of gBests mean history over 51 runs

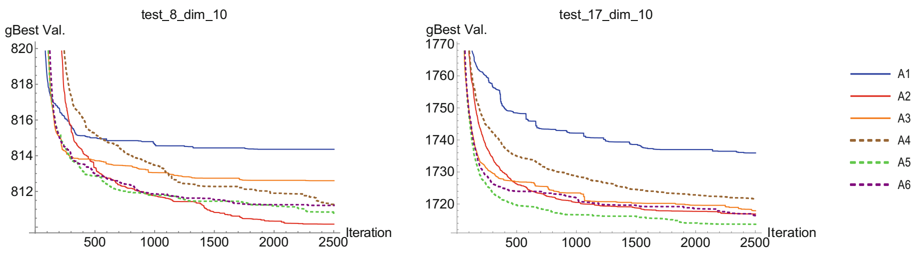

Fig. 4. Comparisons of gBests mean history over 51 runs

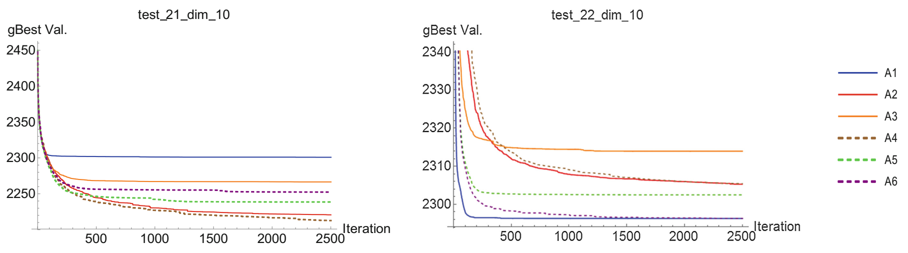

Fig. 5. Comparisons of gBests mean history over 51 runs

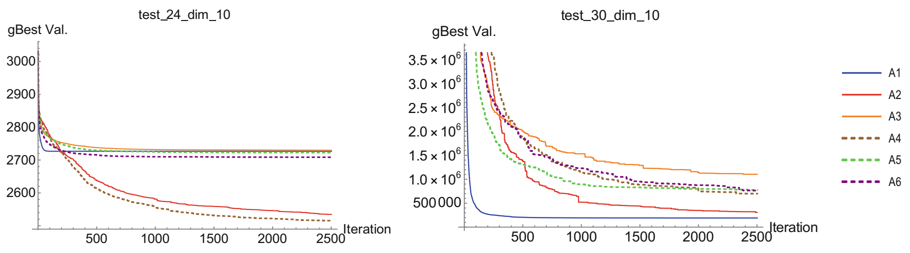

Fig. 6. Comparisons of gBests mean history over 51 runs

6 Results Discussion

The results in Table 3 are sorted in ascending order and the first variant is, therefore, the best performing one. According to Table 4, if the adjusted p-value is smaller than value 0.1 for Holm test, the compared version is significantly better in performance. With this values, the variants A1, A2, A3 and A4 are significantly different from the best performing variant A6. Between variants A6 and A5 there is no significant difference. With this knowledge, the best performing variants are A6 and A5 on tested benchmark functions.

This trend can be seen also in Figs. 4 and 5. In few cases, the *gBest* mean history show the opposite trend, but the difference with others variants are small. In the figures (Figs. 4 and 6), the variants A6 and A5 are close to each other on most test functions.

In Fig. 3, for test function 3 (unimodal), the convergence speed seems to be the slowest for the A2 and A4 variants, this trend can be caused due to the high velocity of particles.

7 Conclusion

In this paper, the results of six possible variants of handling particles position in n-dimensional solution space were presented. Three methods (soft borders, hypersphere and hypersphere with new velocity formula) were tested on the single swarm and their combination on multi-swarm techniques. The methods were tested on classical PSO algorithm. For comparison, the benchmark set CEC'17 was used. The results were presented and tested for statistical significance using Friedman test. Based on this results, some conclusions can be made.

The best performing variants were A6 and A5. Both variants are multi-swarm which have one common border strategy, the hypersphere universe with updated velocity equation.

The goal of this study was to show and compare differences in performance of the selected methods with their combinations using the multi-swarm technique. The results of this study will be further used in future studies to suggest possible improvements for controlling the position of particles that violates search space boundaries.

Acknowledgements. This work was supported by Grant Agency of the Czech Republic – GACR P103/15/06700S, further by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014. Also by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2017/004.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks 1995, pp. 1942–1948 (1995)
2. Awad, N.H. et al.: Problem definitions and evaluation criteria for CEC 2017 special session and competition on single-objective real-parameter numerical optimization (2016)
3. Pluhacek, M., Senkerik, R., Viktorin, A., Zelinka, I.: Single swarm and simple multi-swarm PSO comparison. In: 2016 9th EUROSIM Congress on Modelling and Simulation, Oulu, pp. 498–502 (2016)
4. Kennedy, J.: The particle swarm: social adaptation of knowledge. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 303–308 (1997)
5. Eberhart, R.C., Shi, A.Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation, CEC 2000, pp. 84–88. IEEE (2000)
6. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**, 675–701 (1937)
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)