SUPER-SAPSO: A New SA-Based PSO Algorithm

Majid Bahrepour, Elham Mahdipour, Raman Cheloi, and Mahdi Yaghoobi

Abstract. Particle Swarm Optimisation (PSO) has been received increasing attention due to its simplicity and reasonable convergence speed surpassing genetic algorithm in some circumstances. In order to improve convergence speed or to augment the exploration area within the solution space to find a better optimum point, many modifications have been proposed. One of such modifications is to fuse PSO with other search strategies such as Simulated Annealing (SA) in order to make a new hybrid algorithm –so called SAPSO. To the best of the authors' knowledge, in the earlier studies in terms of SAPSO, the researchers either assigned an inertia factor or a global temperature to particles decreasing in the each iteration globally. In this study the authors proposed a local temperature, to be assigned to the each particle, and execute SAPSO with locally allocated temperature. The proposed model is called SUPER-SAPSO because it often surpasses the previous SAPSO model and standard PSO appropriately. Simulation results on different benchmark functions demonstrate superiority of the proposed model in terms of convergence speed as well as optimisation accuracy.

1 Introduction

Particle Swarm Optimisation and Simulated Annealing have their own advantages and drawbacks. PSO is related to the birds flocking; PSO seeks inside solution space to find the most optimistic result, however if it starts with inappropriate points, it's possible to get stuck into local optimums because of its high velocity; therefore the most problem with PSO is premature convergence [1, 2].

PSO cannot provide appropriate diversity while exploring the solution space because there is no diversity preservation operator to keep solutions in a diverse manner [3, 4]. Simulated Annealing (SA) is a kind of global optimisation technique based upon annealing of metal that uses random search to find optimal points [5, 6]. As might be expected, finding the global optimum in this way is not

Majid Bahrepour
Pervasive Systems Research Group, Twente University, The Netherlands
e-mail: m_bahrepour@ieee.org

Elham Mahdipour
Khavaran University, Mashhad, Iran

Raman Cheloi
Leiden University, the Netherlands

Mahdi Yaghoobi
Islamic Azad University, Mashhad, Iran

J. Mehnen et al. (Eds.): Applications of Soft Computing, AISC 58, pp. 423 – 430.
springerlink.com

guaranteed, but there is usually appropriate diversity in searching solution space due to the hot temperature that lets particles move in any directions.

The SA-based particle swarm optimisation (SAPSO) fuses PSO with SA and often results in more optimised search than PSO and SA separately [2]. In this paper a novel version of SA-based PSO algorithm is proposed. Empirical results reveal that the proposed approach is highly optimised, as it often outperforms both the previous SAPSO model as well as the standard PSO approach.

This paper is structured as follows. In Section 2, standard PSO algorithm is described briefly. In Section 3, the previous SAPSO model is reviewed. In Section 4, the proposed SA-based algorithm, SUPER-SAPSO, is introduced. In Section 5, the experimental results are demonstrated and compared. Finally Section 6 discusses the results and presents some conclusions.

2 Particle Swarm Optimisation

According to Eberhart and Kennedy [1, 7, 8], PSO is a kind of evolutionary algorithms that discover the best solution by simulating the movement and flocking of birds [1]. Each particle is directed to special coordinate. These particles are moved toward the global optimum after some iteration. PSO optimisation, each particle has the ability to remember its previous best position (P_{Best}) and the global best position (G_{Best}). In addition, a new velocity value (V) is calculated based on its current velocity. The new velocity value is then used to compute the next position of particle in solution space. The original PSO's velocity formula is:

$$\begin{aligned} V[t+1] &= w.v[t] + c_1.rand(1).(G_{Best}[t] - \text{Present}[t]) \\ &\quad + c_2.rand(1).(P_{Best}[t] - \text{Present}[t]) \\ \text{Present}[t+1] &= \text{Present}[t] + V[t+1] \end{aligned} \tag{1}$$

Where $V[]$ is the velocity vector; variables c_1,c_2 are the acceleration constants and positive constants; *rand* is a random number between 0 and 1; Present[] is the position vector; W is the inertia weight.

W is not usually appear in standard PSO algorithm version. Searching inside the solution space using PSO, the exploration area is reduced gradually as the generation increasing which can be considered as a clustering strategy near the optimal point [1, 2].

3 SAPSO Hybrid Algorithm

SAPSO algorithm which is a combination of SA and PSO can avoid the key problem of PSO being premature convergence [2]. The premature convergence happens due to fast clustering of particles near the optimal point whilst the optimal point might be a local optimum. Therefore, PSO algorithm can potentially find the global optimum, but it is also possible to get stuck in local optimums. SA

algorithm can provide a better variety in seeking the solution space due to the hot temperature letting the particles move freely in any direction. Combination of PSO and SA can bear new benefits and compensate drawback of the both [2].

Similar to PSO algorithm, SAPSO searching process is started with random initialisation (dispersion) of particles. Firstly, each particle is moved by SA algorithm to a new position that augments variety of search which is accomplished by the use of Equation (2). Secondly, PSO will help the particles converge to global optimal by the use of Equation (1). This process is then repeated until a minimum error is achieved or maximum iterations are reached.

In the process of real annealing of metals, each new particle randomly is laid around the original particles. In this method variation range of original particles can be determined as a parameter like r_1, Expression (2) is a formulation for the variation of particles [2].

$$\text{Present}[t+1] = \text{Present}[t] + r_1 - r_1.2.rand(1) \tag{2}$$

Where parameter r_1 reduce gradually as the generation is increasing, *rand(*1*)* is random number between 0 and 1.

According to [2] the SAPSO algorithm is as follows:

1. Initialise *n* particles randomly.
2. Compute each particle's fitness.
3. Transform particles with the SA algorithm according to the Expression (2).
4. Compare particle's fitness evaluation with its personal best position (P_{Best}) if its fitness is better; replace P_{Best} with its fitness.
5. Compare particle's fitness evaluation with its global best position (G_{Best}) if its fitness is better; replace G_{Best} with its fitness.
6. Update each particle's velocity and position according to the Expressions (1).

This process continues until either the appropriate fitness is achieved or maximum iterations are reached.

4 Proposed Algorithm (SUPER-SAPSO)

Analogous with SAPSO algorithm, SUPER-SAPSO algorithm fuses SA with standard PSO, but particles' movement is done by the Expression (3) rather than Expression (2).

$$\text{Present}[t+1] = (\text{Present}[t] + v[t+1]).T \tag{3}$$

Where *Present[]* is the location vector, *v[]* is the velocity vector, and *T* is temperature of particle. Where *T* is a function of error and by the growth of error, *T* is increased as well.

SUPER-SAPSO algorithm is as follows:

1. Initialise a population of particles with random positions and velocities.

2. Calculate G_{Best} and assign T=1 to it.
3. For remaining particles evaluate fitness and assign temperature to them ($1 \leq T \leq 4$) so those particles with poorer fitness must have hotter temperature.
4. Transform particles according to the Expression (3).
5. For each particle, compares its fitness and its personal best position (P_{Best}) if current value is better than P $_{Best}$, the set P_{Best} equal to current value.
6. For each particle, compares its fitness and its global best position (G $_{Best}$) if current value is better than G_{Best}, the set G_{Best} equal to current value.
7. Update each particle's velocity and position according to the Expressions (1).
8. Best particle among n particles is recognised as leading particle to advance search process.
9. Go to Step 4, until the termination criterion is satisfied.

Termination criterion is either achievement of appropriate fitness or termination of computational time.

The main differences between SUPER-SAPSO algorithm and SAPSO algorithm are:

1. SUPER-SAPSO assigns the temperatures locally to the each particle.
2. The temperature is a function of error.

SUPER-SAPSO in comparison with SAPSO and standard PSO is investigated on benchmark functions and the experimental results are reported in the next section.

5 Experimental Results

Seven numeric optimisation problems were chosen to compare the relative performance of SUPER-SAPSO algorithm to SAPSO and PSO. These functions are standard benchmark test functions and are all minimisation problems.

The first test function is the generalised Rastrigin function:

$$F_1(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \tag{4}$$

The second function is the Foxholes function given by:

$$F_2(x) = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}} \tag{5}$$

Where a_{1j} and a_{2j} is:

$$a_{1j} = \begin{cases} -32 \rightarrow \mathrm{mod}(j,25) = 1 \\ -16 \rightarrow \mathrm{mod}(j,25) = 2 \\ 0 \rightarrow \mathrm{mod}(j,25) = 3 \\ 16 \rightarrow \mathrm{mod}(j,25) = 4 \\ 32 \rightarrow \mathrm{mod}(j,25) = 5 \end{cases}$$

$$a_{2j} = \begin{cases} -32 \rightarrow j > 0 \wedge j \leq 5 \\ -16 \rightarrow j > 5 \wedge j \leq 10 \\ 0 \rightarrow j > 10 \wedge j \leq 15 \\ 16 \rightarrow j > 15 \wedge j \leq 20 \\ 32 \rightarrow j > 20 \wedge j \leq 25 \end{cases}$$

The third test function is the generalised Griewangk function:

$$F_3(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{6}$$

The fourth function is the Sphere function:

$$F_4(x) = \sum_{i=1}^{n} x_i^2 \tag{7}$$

The fifth function is the Ackley function:

$$F_5(x) = 20 + e - 20e^{-0.2\left[\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right]} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} \tag{8}$$

The sixth function is the Step function:

$$F_6(x) = 6n + \sum_{i=1}^{n} \lfloor x_i \rfloor \tag{9}$$

The final function is the Schwefel's Double Sum function:

$$F_7(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2 \tag{10}$$

Table 1 and Figures (1-7) show the results of SUPER-SAPSO in comparison with SAPSO and PSO algorithms respectively.

In all of examinations the numbers of particles are the same and equal to 30. Each examination is repeated 20 times and the average value is reported.

Table 1 Performance results of SUPER-SAPSO, SAPSO and PSO algorithms on benchmark functions

FUNCTION NAMES	ALGORITHM	DIMENSIONS (n)	NUMBER OF ITERATIONS	AVERAGE ERROR
Rastrigin	PSO	20	2989	0.097814
	SAPSO		2168	0.07877
	SUPER-SAPSO		5	0.0
Foxholes	PSO	2	652	0.004846
	SAPSO		536	0.002848
	SUPER-SAPSO		6	0.0
Griewangk	PSO	20	2004	0.082172
	SAPSO		1517	0.075321
	SUPER-SAPSO		3	0.0
Sphere	PSO	20	805	0.094367
	SAPSO		503	0.085026
	SUPER-SAPSO		4	0.0
Ackley	PSO	20	4909	0.099742
	SAPSO		3041	0.099461
	SUPER-SAPSO		5	0.0
Step	PSO	20	10	0.0
	SAPSO		8	0.0
	SUPER-SAPSO		3	0.0
SchwefelDoubleSum	PSO	20	1964	0.086542
	SAPSO		847	0.074965
	SUPER-SAPSO		4	0.0

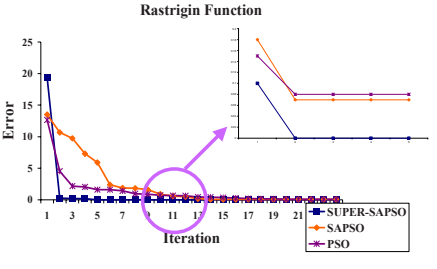

Fig. 1 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Rastrigin function

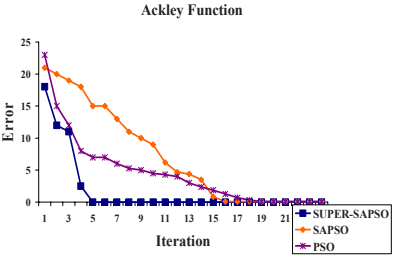

Fig. 2 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithm on Ackley function

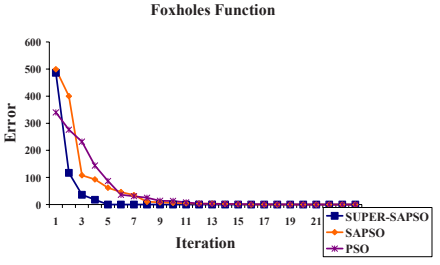

Fig. 3 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Foxholes function

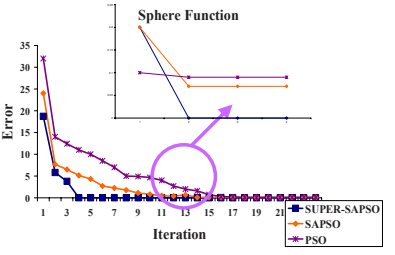

Fig. 4 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Sphere function

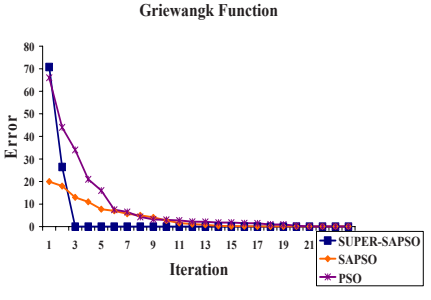

Fig. 5 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Griewangk function

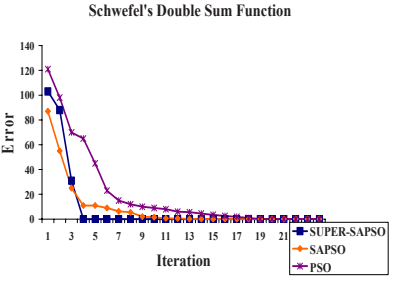

Fig. 6 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Schwefel's Double Sum function

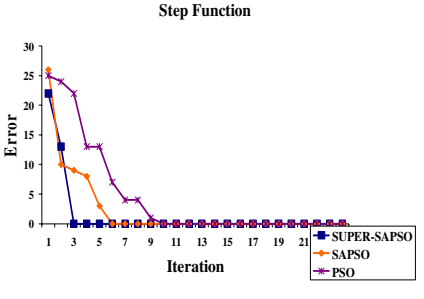

Fig. 7 Convergence comparison of SUPER-SAPSO, SAPSO and PSO algorithms on Step function

6 Conclusions and Future Works

In this paper a new SA-based PSO algorithm, namely SUPER-SAPSO, is proposed. Various tests are carried out on different bench mark functions and

superiority of the proposed model demonstrated. The empirical results show that convergence ratio of SUPER-SAPSO algorithm is almost 282 times faster than SAPSO and 435 times faster than standard PSO algorithm in average. The proposed model not only augments convergence speed but it reduces the optimisation error as well. Therefore the proposed algorithm can move the particles faster towards global optimum with bearing less error.

The results are promising; we are about to develop appropriate multi-objective version of SUPER-SAPSO and solve some applied engineering problems with SUPER-SAPSO and report the results in near future.

References

1. Kennedy, J., Eberhart, R.: Swarm Intelligence. The Morgan Kaufmann Series in Evolutionary Computation (2000)
2. Wang, X.-H., Li, J.-J.: Hybrid Particle Swarm Optimization with Simulated Annealing. In: Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai (2004)
3. Bayraktar, Z., Werner, P.L., Werner, D.H.: Array Optimization via Particle Swarm Intelligence. In: Antennas and Propagation Society International Symposium (2005)
4. Russell, S.J., Norving, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education, Inc., London (2003)
5. Eglese, R.W.: Simulated Annealing: A Tool for Operational Research. European Journal of Operational Research 76(3), 271–281 (2000)
6. Ingber, L.: Simulated Annealing: Practice Versus Theory. Mathl. Comput. Modelling (2001)
7. Oliveira, L.S.: Proving Cascading Classifiers with Particle Swarm Optimization. In: Eighth International Conference on Document Analysis and Recognition (ICDAR 2005) (2005)
8. Oliveira, L.S., Britto, A.S., Sabourin Jr., R.: Optimizing Class-related Thresholds with Particle Swarm Optimization. In: 2005 IEEE International Joint Conference on IJCNN 2005 Proceedings (2005)