# Final Exam
# Applied Computational Intelligence -7650

# SA Embedded Within PSO Algorithm

**Mohaimen Rahman**
**Student# 7890305**

**Abstract:**

Sometimes Particle swarm optimization (PSO) reaches a local optimal solution prematurely, and all particles gather around it. Escaping from these local optima becomes difficult. To avoid premature convergence of PSO, some parts of simulated annealing can be used with it and a hybrid algorithm can be generated called SA embedded within PSO to escape premature local minima and obtain global optimal solution. Two standard benchmark functions (Rastrigin and Griewank) have been used here to evaluate the performance of this algorithm.

**Introduction:**

Particle swarm optimization (PSO) is a swarm intelligent optimization technique where all the particles in the swarm communicate with each other and tend to move towards the global optima updating their position through the velocity. The velocity helps them to update their positions in every iteration. Though PSO is an intelligent optimization technique but it sometimes get stuck into local minima and failed to reach the global optimal solution. Simulated annealing can play a very important role here to not get stuck in any local minima, as it is a probabilistic technique that employs a degree of randomness as part of its logic. So if Simulated annealing (SA) is embedded within Particle swarm optimization (PSO) then the PSO will not stuck in any local minima rather it aims to reach the global minima with better performance.

To compare and check the performance of the simulated annealing (SA) embedded within Particle Swarm optimization (PSO) algorithm, Travelling Salesman problem and two standard benchmark functions (Rastrigin and Griewank) were implemented through this algorithm.

**Related Work:**

There is a good number of optimization algorithms have been developed over last few decades. Among them, there are many heuristics like Simulated Annealing (SA) [1] and other optimization algorithms, which make use of social or evolutionary behaviors like Particle Swarm Optimization (PSO) [2, 3]. Particle Swarm Optimization (PSO) has been developed on the basis of social behavior of each particle living together in groups. Each particle aims to improve itself by observing other group members and imitating the better ones. This way, the particles of the group are performing an optimization procedure.

Particle Swarm Optimization (PSO) was introduced in 1995 by social psychologist James Kennedy and professor and chairman of electrical and computer engineering Russell C. Eberhart [4]. Their purpose was to simulate the natural swarming behavior of birds as they search for food. They developed a simulation of multiple particles flying around a cornfield with the goal of determining if the swarm of particles would flock towards the food.

As the PSO has a pretty good probability to get stuck in local minima, so simulated annealing (SA) is added which significantly reduces the chances of early convergence [5]using a temperature ramping schedule which prevents swarm particles from only relying on improved evaluations of the objective function [6].

Modified particle swarm optimization was previously used to find the travelling salesman problem [8]. To evaluate the performance of the algorithm (SA embedded within PSO), benchmark functions were used [7]. These functions play a very important role for testing the credibility of an optimization algorithm. Among a good number of benchmarks, the newly generated algorithm has been tested with two of them.

**System description:**

The PSO updates the position of each particle through updating their velocity ($v_i$). If vector of the particles is $X_i = (x_{i,1}, x_{i,2}, x_{i,3}, \dots \dots \dots, x_{i,n})$ then their velocity vector will be $V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \dots \dots \dots, v_{i,n})$ respectively for a N dimensional space. The personal best($p_{best}$) and global best ($g_{best}$) are found among the particles. The value of the parameters $\omega, c_1, c_2$ are 0.5 , $r_1$ and $r_2$ are randomly distributed in the range [0,1] and then the velocity becomes

$$\boldsymbol{v_{i,j}(m+1) = \omega_m v_{i,j}(m) + c_{1,m} r_1 \left(p_{i,j}^{best} - x_{i,j}(m)\right) + c_{2,m} r_2 \left(g_{best} - x_{i,j}(m)\right)}$$

The updated position of each particle becomes

$$\boldsymbol{x_{i,j}(m+1) = x_{i,j}(m) + v_{i,j}(m)}$$

Particle swarm optimization (PSO) updates the position with the above-mentioned process and reaches the global optima. But to compare the basic PSO with the SA embedded within PSO travelling salesman problem (TSP) has been developed besides benchmark functions. Here everything was same but the velocity has been calculated by swapping instead of arithmetic operations.

Suppose there are 5 cities in 2-d space (Euclidean space). The cities are numbered from 1,2,3,4,5. Now let a particle P be [2,4,5,1,3] and Q be [3,1,2,5,4].In the velocity equation there are (-) operator and (+) operators. So the meaning of the (-) operator has been changed as follows.

To determine (P - Q), the concept of Swap Sequence has been introduced where S meaning that S=P - Q by which it mean the swapping sequence to go from the particle Q to P. S will be
1 3 $\Rightarrow$ meaning to swap between the 1st and 3rd element in Q $\Rightarrow$ [2,1,3,5,4]
2 5 $\Rightarrow$ meaning to swap between the 2nd and 5th element in Q $\Rightarrow$ [2,4,3,5,1]
3 4 $\Rightarrow$ meaning to swap between the 3rd and 4th element in Q $\Rightarrow$ [2,1,5,3,4]
4 5 $\Rightarrow$ meaning to swap between the 4th and 5th element in Q $\Rightarrow$ [2,1,5,4,3] = P

Thus applying the swap sequence S to Q, P from Q can be got.
Now for the (+) operator, ordered union of Swap Sequence is made. Let T be another sequence where
T is
1 5
2 4

Then V = S+T will be
1 3
2 5
3 4
4 5
1 5
2 4

Now let X = [4,2,5,1,3].

To to determine a new particle [as updating the particle position by X $\leftarrow$ X+V], Applying V being the whole Swap Sequence to determine the new X.
Thus applying
1 3 $\Rightarrow$ meaning to swap between the 1st and 3rd element in X $\Rightarrow$ [5,2,4,1,3]
2 5 $\Rightarrow$ meaning to swap between the 2nd and 5th element in X $\Rightarrow$ [5,3,4,1,2]
3 4 $\Rightarrow$ meaning to swap between the 3rd and 4th element in X $\Rightarrow$ [5,3,1,4,2]
4 5 $\Rightarrow$ meaning to swap between the 4th and 5th element in X $\Rightarrow$ [5,3,1,2,4]
1 5 $\Rightarrow$ meaning to swap between the 1st and 5th element in X $\Rightarrow$ [4,3,1,2,5]
2 4 $\Rightarrow$ meaning to swap between the 2nd and 4th element in X $\Rightarrow$ [4,2,1,3,5]

The new X is created as [4,2,1,3,5].

Thus the velocity equation works and update position of PSO in case of TSP problem.

But PSO sometimes gets stuck in a local optimal solution prematurely. To avoid it SA has been added. The cost of the vector X was sent to SA. The Boltzmann probability function has been used here

$$\mathbf{P = exp}\left(\frac{\Delta f}{T}\right)$$

After adding the SA with PSO the new algorithm has become able to not get stuck in a premature local minima and reach a global optimal solution.

**Algorithm:**

$\mathrm{d}$ = Dimension as the size of the set of values for the benchmark function
$\mathrm{value_{min}}$ = Minimum value of the range of the benchmark function
$\mathrm{value_{max}}$ = Maximum value of the range of the benchmark function
$\mathrm{p}$ = Number of particles
$\mathrm{M}$ = Maximum number of iterations, $\mathrm{M} \geq 10$
$\omega$ = Inertial constant of the velocity
$\mathrm{P_S}$=Probability of accepting worse solution at the start
$\mathrm{P_F}$=Probability of accepting worse solution at the end
$\mathrm{T_S} = \frac{-1}{\ln \mathrm{P_S}}$
$\mathrm{T_F} = \frac{-1}{\ln \mathrm{P_F}}$
$\mu = \left(\frac{\mathrm{T_F}}{\mathrm{T_S}}\right)^{\frac{1}{\mathrm{M}-1}}$
$\mathrm{T_{k+1}} = \mu \mathrm{T_k};\ \mathrm{k} \geq 0$

**procedure** PSOwithSA(d, $\mathrm{value_{min}}$, $\mathrm{value_{max}}$, p, M, $\omega$, $\mathrm{c_1}$, $\mathrm{c_2}$, T)
Initialize $\mathrm{P_S}, \mathrm{P_F}$ with proper values
$\mathrm{T_S} \leftarrow \frac{-1}{\ln \mathrm{P_S}}$
$\mathrm{T_F} \leftarrow \frac{-1}{\ln \mathrm{P_F}}$
$\mu \leftarrow \left(\frac{\mathrm{T_F}}{\mathrm{T_S}}\right)^{\frac{1}{\mathrm{M}-1}}$
Let x, pBest be **p** particles each having **d** dimension.
Let sBest be the swarm's best particle having **d** dimension.
Let v be the velocity of p particles each having **d** dimension.

**for** $i = 1, 2, ..., p$ **do**
    **for** $j = 1, 2, ..., d$ **do**
        $\mathrm{x}_{i,j} \leftarrow \mathrm{rand}[\mathrm{value_{min}}, \mathrm{value_{max}}]$
        $\mathrm{pBest}_{i,j} \leftarrow \mathrm{x}_{i,j}$
    **end for**
    **if** $i = 1$ **then**
        $\mathrm{sBest} \leftarrow \mathrm{pBest}_i$, $\mathrm{Cost\_sBest} \leftarrow \mathrm{Cost}(\mathrm{pBest}_i)$
    **end if**
    **if** $\mathrm{Cost}(\mathrm{x}_i) < \mathrm{Cost\_sBest}$ **then**
        $\mathrm{sBest} \leftarrow \mathrm{x}_i$, $\mathrm{Cost\_sBest} \leftarrow \mathrm{Cost}(\mathrm{x}_i)$
    **end if**
**end for**

**for** $k = 1, 2, ..., M$ **do**
    **for** $i = 1, 2, ..., p$ **do**
        $\mathrm{r_p} \leftarrow \mathrm{rand}[0, 1]$, $\mathrm{r_g} \leftarrow \mathrm{rand}[0, 1]$
        $\mathrm{v}_i \leftarrow \omega \mathrm{v}_i + \mathrm{c_1 r_p}\left(\mathrm{pBest}_i - \mathrm{x}_i\right) + \mathrm{c_2 r_g}\left(\mathrm{sBest} - \mathrm{x}_i\right)$
        $\mathrm{newX} \leftarrow \mathrm{x}_i + \mathrm{v}_i$
        $\mathrm{x}_i \leftarrow \mathrm{newX}$
        $\mathrm{pX} \leftarrow \mathrm{Perturb}(\mathrm{d}, \mathrm{value_{min}}, \mathrm{value_{max}}, \mathrm{newX})$
        $\mathrm{D_b} \leftarrow \mathrm{Cost}(\mathrm{x}_i)$
        $\mathrm{D_j} \leftarrow \mathrm{Cost}(\mathrm{pX})$
        $\Delta \leftarrow |\mathrm{D_b} - \mathrm{D_j}|$
        **if** $\mathrm{D_j} > \mathrm{D_b}$ **then**
            **if** $\mathrm{k} = 1$ and $\mathrm{i} = 1$ **then**
                $\Delta_{\mathrm{average}} = \Delta$
            **end if**
            $\mathrm{p} \leftarrow \mathrm{e}^{-\frac{\Delta}{\Delta_{\mathrm{average}} \mathrm{T}}}$

```
            if p > rand[0, 1] then
                accept ← true
            else
                accept ← false
            end if
        else
            accept ← true
        end if

        if accept = true then
            x_i ← pX
            D_b = D_j
            numAcceptedSolutions = numAcceptedSolutions + 1
            Δ_average ← (Δ_average(numAcceptedSolutions−1)+Δ) / numAcceptedSolutions
        end if

        if Cost(x_i) < Cost(pBest_i) then
            pBest_i ← x_i
            Cost(pBest_i) ← Cost(x_i)

            if Cost(pBest_i) < Cost_sBest then
                sBest ← pBest_i
                Cost_sBest ← Cost(pBest_i)
            end if
        end if
    end for
    T ← μT
end for
end procedure
```

**Experiments and Results:**

Let the number of particles t=15, $\omega$ =0.5, $c_1 = 0.5, c_2 = 0.5$, maximum number of iterations = 1000 and got results with the following experiments.

**1. PSO and SA embedded within PSO (Solving Travelling salesman problem):**

The Travelling salesman problem has been solved through the particle swarm optimization (PSO) and Simulated annealing (SA) embedded within particle swarm optimization (PSO) and the below results were found:

| **Travelling salesman problem** | **Results** |
|---|---|
| Solved by **PSO** | **17074.4102** |
| Solved by **PSO-SA** | **7838.7712** |

**Table 1: Travelling Salesman Problem by PSO and SA embedded within PSO**

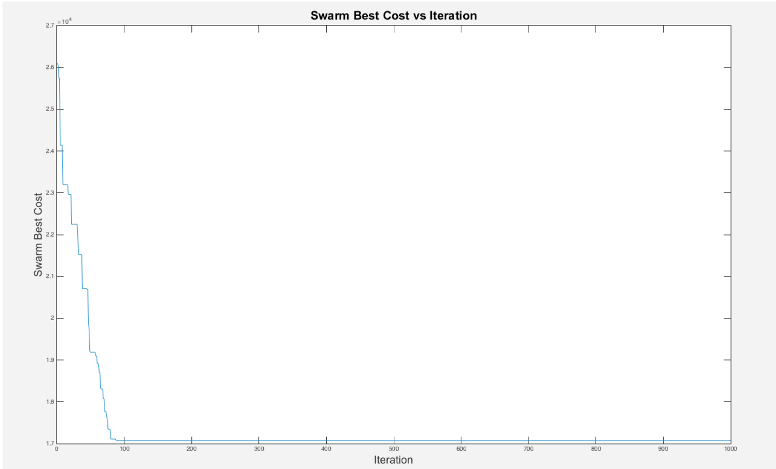


**Figure 1: Swarm Best Cost vs. Iteration (Travelling salesman problem solved by PSO)**

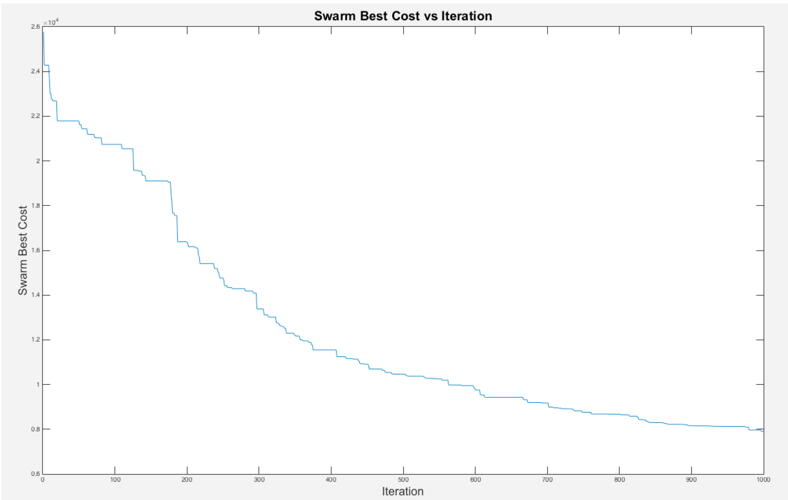


**Figure 2: Swarm Best Cost vs. Iteration (Travelling salesman problem solved by SA embedded within PSO)**

## 2. PSO and SA embedded within PSO (with benchmark functions):

To check the performance of the algorithm two benchmark functions were used for both PSO and SA embedded within PSO and the below results were found:

| Serial No. | Function | Range | Results | | |
|---|---|---|---|---|---|
| | | | Dimension n=2 | Dimension n=10 | Dimension n=30 |
| 1. | Basic PSO with Rastrigin | -5.12 to 5.12 | $2.6802\boldsymbol{e}^{-5}$ | $4.2633\boldsymbol{e}^{-14}$ | $7.2677\boldsymbol{e}^{-10}$ |
| 2. | PSO-SA with Rastrigin | -5.12 to 5.12 | 0 | 0 | 0 |
| 3. | Basic PSO with Griewank | -600 to 600 | $1.4163\boldsymbol{e}^{-5}$ | 2.7613 | 43.9708 |
| 4. | PSO-SA with Griewank | -600 to 600 | $3.3307\boldsymbol{e}^{-16}$ | 0.168 | 1.0269 |

**Table 2: PSO and SA embedded within PSO through Benchmark functions**

After getting all the results of the above experiments it can be declared that Simulated annealing (SA) embedded within particle swarm optimization (PSO) works better than the basic PSO algorithm as the this algorithm helps the particles not to stuck in the local minima and tend to obtain global optimal solution.

**Conclusion and Future work:**

Both basic PSO algorithm and SA embedded within PSO algorithm have been tested along with the benchmark functions in various dimensions and also a comparison is made through the travelling salesman problem. After comparing them we can conclude that the SA embedded within PSO algorithm provides better performance.

In future a dampening strategy can be applied on inertial coefficients to achieve better performance. More benchmark functions other than Rastrigin and Griewank can be used to check the performance of this algorithm.

**References:**

[1] M. Locatelli, "Simulated annealing algorithms for continuous global optimization," in *Handbook of Global Optimization*, P. M. Pardalos and H. E. Romeijn, Eds., vol. 2, pp. 179–230, Kluwer Academic, 2001.

[2] M. Pant, R. Thangaraj, and A. Abraham, "Particle swarm based metaheuristics for function optimization and engineering applications," in *Proceedings of the 7th Computer Information Systems and Industrial Management Applications (CISIM '08)*, vol. 7, pp. 84–90, IEEE Computer Society, Washington, DC, USA, 2008.

[3] J. Kennedy and C. E. Russell, "Swarm intelligence," in *Morgan Kaufmann, Academic Press,* 2001.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Oct. 1995, pp. 39–43.

[5] J. Q. Jiang, Y. C. Liang, X. H. Shi, and H. P. Lee, "A Hybrid Algorithm Based on PSO and SA and Its Application for Two-Dimensional Nonguillotine Cutting Stock Problem," in *Computational Science – ICCS 2004, ser. Lecture Notes in Computer Science*, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Springer Berlin Heidelberg, 2004, pp. 666–669.

[6] S. Kathpal, R. Vohra, J. Singh, and R. S. Sawhney, "Hybrid PSO – SA Algorithm for Achieving Partitioning Optimization in Various Network Applications," *Procedia Engineering*, vol. 38, pp. 1728–1734, Jan. 2012.

[7] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *Tech. Rep. 2005005*, Nanyang Technological University, Singapore; IIT Kanpur, India, 2005.

[8] W. Pan, K.-P. Wang, C.-G. Zhou, L.-J. Dong, and M. Liu, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 4, Aug. 2004, pp. 2342–2346 vol.4.