

# An Improved Particle Swarm Optimization Algorithm for Power-Efficient Wireless Sensor Networks\*

Erfu Yang<sup>1</sup>, Ahmet T. Erdogan<sup>1</sup>, Tughrul Arslan<sup>1</sup>, and Nick Barton<sup>2</sup>

1. School of Engineering and Electronics 2. School of Biological Sciences  
The University of Edinburgh

King's Buildings, Edinburgh EH9 3JL, United Kingdom

{E.Yang, Ahmet.Erdogan, T.Arslan, N.Barton}@ed.ac.uk

## Abstract

*This paper presents an improved particle swarm optimization (PSO) algorithm for onboard embedded applications in power-efficient wireless sensor networks (WSNs) and WSN-based security systems. The objective is to keep the main advantages of the standard PSO algorithm, such as simple form, easy implementation, low algorithmic complexity, and low computational burden while the performance and efficiency can be significantly improved. Numerical experiments are performed on a very difficult benchmark function to validate the performance of the improved PSO algorithm. The results show that the improved PSO algorithm outperforms the standard PSO algorithm.*

## 1 Introduction

Particle swarm optimization (PSO) is one of biologically-inspired optimization techniques. It was first proposed by Eberhart and Kennedy in 1995 [1–3]. Over the last decade PSO has received extensive attention [4–8]. It has been successfully applied in many applications, such as function optimization, artificial neural network, fuzzy control, robotics, mechanical design, radio and antenna design, computer games, and other optimization problems. It has been recognized as one of competitive biologically-inspired algorithms to traditional evolutionary algorithms, such as genetic algorithms (GAs).

Wireless sensor networks (WSNs) represent a new data collection approach and will play an important role in future's high security systems. The objective of this study is to develop an efficient optimization algorithm for onboard embedded applications in power-efficient WSNs. An emergent implementation of such an algorithm is targeted at the

ESPACENET<sup>1</sup> project which aims at the design of evolvable and reconfigurable sensor networks and distributed reconfigurable System-On-Chip (SoC) sensor nodes for aerospace-based monitoring and diagnostics [9]. In this project, low-power is one of main design objectives. Thus, low complexity, high efficiency, predictable performance, and flexibility are the key requirements to optimization algorithm which runs onboard in an embedded manner.

For onboard embedded applications in power-efficient systems, PSO is becoming a promising optimization algorithm due to its many advantages. First, PSO has a fast convergence rate. Compared with traditional evolutionary algorithms (e.g., GAs), PSO tends to converge to the best solution quickly. Second, PSO is very simple and easy to implement. There are few parameters to adjust in PSO. This is particularly attractive for onboard embedded applications in power-efficient WSNs since the computational resources such as memory and energy are extremely limited. Third, PSO is also computation-efficient. In PSO the number of particles (i.e., population size) is very small. In fact, for most of the problems 10 particles will be large enough to get satisfactory results. For traditional evolutionary algorithms the population size is often required to be set as larger as possible to get an acceptable solution.

The intent of PSO is to solve the global optimization problem. However, the standard PSO algorithm is easily trapped in local optima. The second problem which PSO is often criticized is that some particles may fly far away from the search region. To solve the second problem the particles are often forced to be zero velocity at the boundary. However, this rough method may decrease the efficiency of PSO algorithm. It has also shown that in some cases the quality of solution does not improve as the number of iterations increases, see [10] and the reference therein.

The objective of this paper is to present an improved PSO

\*This research is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/C546318/1.

<sup>1</sup>Evolvable Networks of Intelligent and Secure Integrated and Distributed Reconfigurable System-On-Chip Sensor Nodes for Aerospace Based Monitoring and Diagnostics.

algorithm for onboard embedded applications in power-efficient WSNs. This improved PSO algorithm can keep all the advantages of the standard PSO, such as implementation simplicity, low computational burden, and few control parameters, etc. However, its performance can be much better than the standard PSO algorithm. This is particularly attractive to the applications in power-efficient WSNs.

## 2 Particle Swarm Optimization

PSO is inspired by social behavior of bird flocking or fish schooling. In these biological systems the collective behaviors of simple individuals interacting with their environment and each other play a very important role. PSO just simulates the behaviors of biological swarms by randomly searching food in an area. There is only one piece of food in the area being searched. All the swarms (such as birds) do not know where the food is. But they know how far the food is in each trial by observing their fitness. The question is how the particle swarm can find the food in an efficient way? The basic idea of solving such a swarm optimization problem is to follow the leader particle which is nearest to the food. Since the leader particles represent the current optimum solutions, in PSO the particles in a population always move through the problem space by following the current optimal particles.

Obviously, PSO is a population based stochastic optimization technique. It is first initialized with a group of random particles (solutions). In every iteration each particle is updated by following two “best” values, i.e., the personal best  $\mathbf{x}_{p,i}$ , and the global best  $\mathbf{x}_g$ .  $\mathbf{x}_{p,i}$  is the best solution (fitness) one particle has achieved so far.  $\mathbf{x}_g$  is the best value obtained so far by any particle in the population.

In the original PSO algorithm, the particle updates its velocity and position as follows:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1r_1[\mathbf{x}_{p,i}(t) - \mathbf{x}_i(t)] + c_2r_2[\mathbf{x}_g(t) - \mathbf{x}_i(t)] \quad (1)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2)$$

where  $\mathbf{v}_i$  and  $\mathbf{x}_i$  are the particle velocity and position at the  $t$ th iteration, respectively.  $c_1$  and  $c_2$  are the scaling constants, usually taken as  $c_1 = c_2 = 2.0$ .  $w$  is the inertia weight and used to control the trade-off between the global and the local exploration ability of the swarm. Random numbers  $r_1$  and  $r_2$  are uniformly distributed in  $[0, 1]$ .

As pointed out previously in this paper, the standard PSO algorithm has several drawbacks when dealing with some difficult optimization problems. To improve the performance of PSO algorithms there have been many variants. For instance, an improved particle PSO algorithm was proposed in [4] by adding an extra loop at the beginning

of the algorithm to keep randomly re-initializing infeasible particles for ensuring that they stay inside the feasible search space. In [10] a dissipative PSO algorithm was developed by using the self-organization mechanism of dissipative structure. Three versions of PSO algorithms with novel learning strategies were presented in [11].

Though these variants may improve the performance of the standard PSO in some sense, the main advantages of the standard PSO such as implementation simplicity, low computational burden, and few control parameters may be lost. Obviously, losing these merits of PSO algorithms are not desirable to onboard embedded applications in power-efficient WSNs, particularly for the ESPACENET project.

## 3 Improved PSO Algorithm

To improve the performance of the standard PSO algorithm, the key thing in this study is to keep an efficient balance between exploration and exploitation of particles in swarm. A basic idea is to keep the diversity of both local and global optima. To realize this point, a very natural way is to randomly “perturb” these optima at current iteration. In the standard PSO algorithm, these perturbations are only applied to the difference between the local best solution and the particle position or between the current global best solution and the particle current position. A drawback of this scheme lies in its restricting the exploration ability of particle swarm.

To widen the exploration ability of particle swarm, we propose that the perturbations should be directly applied to the local best solution and global best solution achieved so far. Such a kind of perturbations can be generated by using Metropolis algorithm(MA). However, a probability density function (PDF) may greatly increase the computational complexity of PSO algorithm. Therefore, in this paper we adopt a simple way to generate random perturbations as in the standard PSO algorithm. The second objective in developing our improved PSO algorithm is to improve the convergence process of the particles in swarm such that all the particles are prevented from flying far away from the feasible search space.

Keeping the above points in mind, we propose an improved PSO algorithm, formulated in **Algorithm 1**.

In (3)  $r_3$  and  $r_4$  are uniformly distributed random numbers in  $[-\varepsilon, \varepsilon]$ , where  $\varepsilon$  is a small positive constant. It can also be adaptively tuned online to speedup the convergence.

## 4 Numerical Example

In this section numerical experiments are performed to validate the effectiveness of our improved PSO algorithm and compare its performance with the standard PSO

---

**Algorithm 1** Improved PSO Algorithm
 

---

Set constant  $w$  and other control parameters.  
 For each particle  $i$  in the population  
   Randomly initialize particle velocity and position  
 End  
 Do  
   For each particle  $i$  in the population  
     Calculate fitness value and evaluate the move  
     If the current fitness value is better than  $\mathbf{x}_{p,i}$  in history  
     Set the current value as the new  $\mathbf{x}_{p,i}$

End  
 From the whole population or the prescribed topology of neighborhood, choose the particle with the best fitness value as the global best solution  $\mathbf{x}_g$

For each particle  $i$  in the population  
   Calculate particle velocity according to the following equation

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + [(1+r_3)\mathbf{x}_{p,i}(t) - \mathbf{x}_i(t)]/3 + [(1+r_4)\mathbf{x}_g(t) - \mathbf{x}_i(t)]/3 \quad (3)$$

Update particle position using the following relation:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (4)$$

End

While maximum iterations or minimum error criteria is not met.

---

algorithm. The latest version of the standard PSO algorithm was published in 2006 and can be obtained at <http://www.particleswarm.info> website. The idea of the standard PSO 2006 is to define a real standard at least for one year, validated by some researchers of the field, in particular James Kennedy and Maurice Clerc.

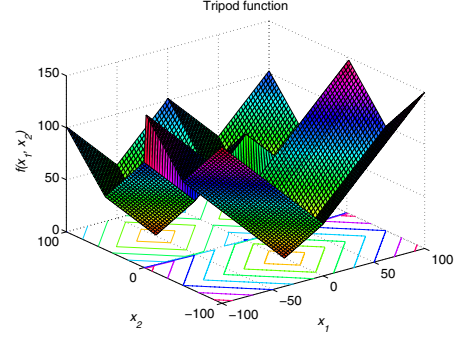
The version of the standard PSO 2006 is the nearest of the original version (1995) with just a few improvements based on some recent works. In the standard PSO 2006, information links topology is created by using a random approach. So the number of particles that informs a given one may be any value between 1 (for each particle informs itself) and the swarm (population) size. To keep the standard PSO 2006 as closer as the original version (1995), initial positions are chosen at random inside the search space according to an uniform distribution. Each initial velocity is simply defined as the difference of two random positions. It is simple, but needs no additional parameter.

To have a very fair comparison, we just modified the code of the standard PSO 2006 to represent our update rules (3) and (4). The other part of the code is kept the same as the standard PSO 2006. So the programming and running environment is exactly same for the two algorithms.

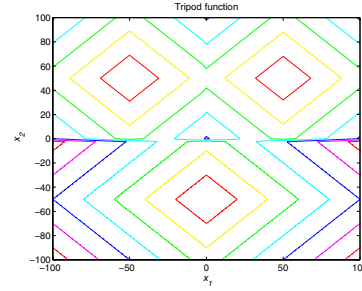
To test the ability of the algorithm not to be trapped in local optima, we chose Tripod function as the benchmark function, as shown in Fig. 1.

The 2-D Tripod function is defined as follows

$$f(x_1, x_2) = p(x_2)[1 + p(x_1)] + |x_1 + 50p(x_2)[1 - 2p(x_1)]| + |x_2 + 50[1 - 2p(x_2)]| \quad (5)$$



**Fig. 1.** 2-D Tripod function



**Fig. 2.** Contour of 2-D Tripod function

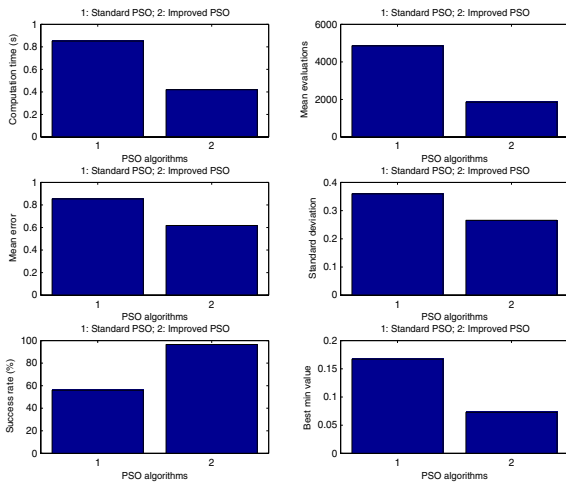
where

$$p(s) = \begin{cases} 1 & s \geq 0 \\ 0 & s < 0 \end{cases}$$

The contour of 2-D Tripod function is depicted in Fig. 2. As shown in Figs. 1 and 2, Tripod function has 2 local minima – the “valleys” in the figures. However, the function has just one global minimum, which occurs at the point (0, -50) in the  $x - y$  plane, where the value of the function is 0. At any local minimum other than (0, -50), the value of Tripod function is greater than 0. It should also be noted that Tripod function is not continuous in search space.

The difficulty of finding global optimum to Tripod function lies in that a lot of algorithms are easily trapped in the two local minima, as shown in Figs. 1 and 2. In this numerical experiment the search space is defined on  $[-100, 100] \times [-100, 100]$ . For the both PSO algorithms in the numerical experiments, the population size was taken as 12 only. The minimum error was set to be 0.9. The maximum number of executions or iterations was 200. The maximum evaluation number is 10,000. The inertia weight  $w = 0.721348$ .  $c_1$  and  $c_2$  were taken as  $0.5 + \log(2)$ .  $\varepsilon$  was picked as 2.0.

The best solutions found by the two algorithms are  $(-0.057177, -49.984349)$  and  $(0.147433, -49.980389)$  for the Improved PSO and standard PSO 2006, respectively. Fig. 3 shows the comparative results obtained from the



**Fig. 3. Performance comparisons of the improved and standard PSO 2006 algorithms**

both PSO algorithms, from which we can see that the improved PSO algorithm greatly outperformed the standard PSO 2006 in this testing case. In particular, the computation time is only about half of the standard PSO 2006. The success rate is nearly doubled against the standard PSO algorithm. Compared with other variants of the standard PSO algorithm, the computational complexity and burden have not been increased in our improved PSO algorithm. We have also conducted a series of numerical experiments on other benchmark functions, the improved performance can be achieved by using our improved PSO algorithm.

## 5 Conclusions

To improve the performance of the standard PSO algorithm, many variants have been proposed in recent years. However, most of them either improve the algorithmic complexity or increase the computational burden, though a performance gain may be achieved in some sense.

This paper has presented an improved PSO algorithm to enhance the performance of the standard PSO algorithm. Compared with other variants, this improved PSO algorithm can keep the main advantages of the standard PSO algorithm, such as simple form, easy implementation, low algorithmic complexity, and low computational burden while its performance and efficiency can be significantly improved. Numerical experiments have been performed on a very difficult benchmark function to validate the performance of the improved PSO algorithm. The results illustrated that the improved PSO algorithm outperformed the standard PSO algorithm. So, it is a very promising runtime optimizer for

onboard embedded applications in power-efficient WSNs and WSN-based high security systems in which the computation recourses and energy are very limited or expensive.

## Acknowledgment

The authors thank all of the team members of the ESPACENET project, which involves the Universities of Edinburgh, Surrey, Essex, and Kent, Surrey Satellite Technology (SSTL), NASA Jet Propulsion Laboratory (JPL), EPSON, and Spiral Gateway.

## References

- [1] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. 6th Int. Sym. on Micro-machine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.
- [2] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. 1995 IEEE Int. Conf. Neural Networks*, pp. 1942–1948, Perth, Australia, 1995.
- [3] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Proc. IEEE Int. Conf. Evol. Comput.*, pp. 69–73, Anchorage, Alaska, 1998.
- [4] S. He, E. Prempan, and Q.H. Wu. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5):585–605, 2004.
- [5] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming. Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Trans. Evol. Comput.*, 10(3):245–255, 2006.
- [6] M. Clerc and J. Kennedy. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.
- [7] S. M. Guru, S. K. Halgamuge, and S. Fernando. Particle swarm optimizers for cluster formation in wireless sensor networks. In *Proc. 2005 Int. Conf. Intel. Sensors, Sensor Networks and Info. Processing Conf.*, pp. 319–324, Melbourne, Australia, 2005.
- [8] M. Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proc. 1999 IEEE Congress on Evol. Comput.*, pp. 1951–1957, Washington, DC, 1999.
- [9] T. Arslan, N. Haridas, E. Yang, and et al. ESPACENET: a framework of evolvable and reconfigurable sensor networks for aerospace-based monitoring and diagnostics. In *Proc. 1st NASA/ESA Conf. Adapt. Hardware and Systems*, pp. 323–329, Istanbul, Turkey, 2006.
- [10] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang. A dissipative particle swarm optimization. In *Proc. 2002 IEEE Congress on Evol. Comput.*, pp. 1456–1461, Honolulu, Hawaii, 2002.
- [11] J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar. Particle swarm optimization algorithms with novel learning strategies. In *Proc. 2004 IEEE Int. Conf. SMC*, pp. 3659–3664, The Hague, the Netherlands, 2004.