Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 41 (2012) 951 – 957

Engineering Procedia

www.elsevier.com/locate/procedia

International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)

# Asynchronous Particle Swarm Optimization for Swarm Robotics

Nor Azlina Ab Aziz[a]*, Zuwairie Ibrahim[b]

[a]Faculty of Engineering and Technology, Multimedia University, Malaysia
[b]Faculty of Electrical and Electronics Engineering, University Malaysia Pahang, Malaysia

**Abstract**

In the original particle swarm optimization algorithm, particles' update is done synchronously. The whole swarm fitness is evaluated first before particle update process is conducted. Whereas in asynchronous update a particle is able to update its velocity and position after its fitness is evaluated. This caused the particle's search to be conducted with imperfect information. However, asynchronous update is useful in field such as swarm robotics search problem, where the robots can move continuously based on the available information without waiting for the whole swarm. Hence this paper looks into the differences between synchronous and asynchronous PSO and its application in swarm robotics search.

*Keywords*: Asynchronous, particle swarm optimization, swarm robotics, synchronous.

**Nomenclature**

| | |
|---|---|
| $c_1$ | cognitive learning factor |
| $c_2$ | social learning factor |
| $d$ | problem dimension |
| $i$ | is particle's number |
| $N$ | number of particles in the swarm |
| $\mathbf{P_i}$ | the best position found so far by the particle |
| $\mathbf{P_g}$ | the best position found by the neighbouring particles |
| $rand_1()$ and $rand_2()$ | two independent random numbers in the range of [0.0,1.0] |
| $\mathbf{V_i}$ | velocity of particle $i$ $[-V_{max}, +V_{max}]$ |
| $w$ | inertia weight |
| $\mathbf{X_i}$ | position of particle $i$ |

## 1. Introduction

Swarm robotics refers to collective and distributed autonomous robotic systems [1]. Target search is one of the active research areas in swarm robotics. An example of target search problem is a search for chemical leakage source by a swarm of robots. A good search strategy allows the robots to perform the search collectively, yet independently of each other.

A swarm intelligence algorithm such as particle swarm optimization (PSO) is a good choice for swarm robotics search problem. In PSO the particles move within search area to find for optimal solution by updating their velocity and position.

* Corresponding author. Tel.: +6-06-2523954; fax: +6-06-2316552.
*E-mail address:* azlina.aziz@mmu.edu.my

PSO can be mapped to the swarm robotics search [2], where in swarm robotics instead of particles, robots with actual velocity and physical position move around a search space to look for the target. However in the original PSO the particles velocity and position update is done in synchronous fashion. This approach is not suitable for swarm robotics, as the robots need to be paused until complete information of the whole swarm is obtained. Another update method known as asynchronous update is believed to be a more suitable approach. Asynchronous update allows the update process to be done using incomplete information. The asynchronous version is able to avoid the robots from being halted during their search process.

In the following section the particle swarm optimization is studied and the difference between synchronous PSO (SPSO) and asynchronous PSO (APSO) is discussed. The performance of SPSO and APSO is studied using three test functions in section 3. Section 4 focuses on the works related to APSO for swarm robotics search. Finally is the conclusion.

## 2. Particle Swarm Optimization

Particle swarm optimization (PSO) was introduced by James Kennedy and Russell Eberhart in 1995. It is a swarm based optimization algorithm that mimics the social behaviour of organisms like birds and fishes. This social behaviour is imitated by PSO using swarm of agents called particles [3]. The interaction of the particles with their neighbours is the key to the effectiveness of PSO.

A particle in PSO has a position ($\mathbf{X_i}$) and velocity ($\mathbf{V_i}$). The position represents a solution suggested by the particle while velocity is the rate of changes of the next position with respect to current position. At the beginning of the algorithm these two values (position and velocity) are randomly initialised. In the subsequent iterations the search process is conducted by updating these values using the following equations:

$$\mathbf{V}_i = w \times \mathbf{V}_i + c_1 \times rand_1() \times (\mathbf{P}_i - \mathbf{X}_i) + c_2 \times rand_2() \times (\mathbf{P}_g - \mathbf{X}_i) \qquad (1)$$

$$\mathbf{X}_i = \mathbf{X}_i + \mathbf{V}_i \qquad (2)$$

where $i$ is particle's number ($i$ = 1,..,$N$; $N$: number of particles in the swarm).

The $\mathbf{V_i}$ value is clamped to $\pm V_{max}$ to prevent explosion. If the value of $V_{max}$ is too large, the exploration range is too wide, however if it is too small, particles will favour local search [4]. In equation (1), $c_1$ and $c_2$ are the learning factors that control the effect of cognitive and social influence to a particle, typically both $c_1$ and $c_2$ are set to 2 [5]. $rand_1()$ and $rand_2()$ are two independent random numbers in the range of [0.0,1.0]. The randomness terms provides energy to the particles. $w$ is known as inertia weight, a term added to improve PSO's performance. The inertia weight controls particles momentum so that, they can switch to fine tuning when a good area is found [6]. A time decreasing inertia weight is better than a fixed inertia weight [4]. This is because larger inertia weight at the beginning helps to find good area through exploration and small inertia weight towards the end – where typically a good area is already found – facilitates fine tuning. The small inertia weight at the end of the search, reduce the global search activity during this period [7].This ensure convergence.

The success of an individual in PSO is affected not only by particle's own effort and experience but also by the information shared by its surrounding neighbours. The particle's experience is represented in the equation by; $\mathbf{P_i}$ – the best position found so far by the particle, while the neighbours' influence is represented by; $\mathbf{P_g}$ – the best position found by the neighbouring particles. $\mathbf{P_g}$ value depends on the neighbourhood type. Particles neighbourhood in PSO had been studied from two perspectives; global neighbourhood (*gBest*) and local neighbourhood (*lBest*). In *gBest* the particles are fully connected therefore the particles search is directed by the best particle of the swarm. While in *lBest* the particles are connected to their neighbours only and their search is conducted by referring to the neighbourhood best.

The particle's position is updated using equation (2) where the velocity is added to the previous position. This shows how a particle next search is launched from its previous position and the new search is influenced by its pass search [8]. Typically $\mathbf{X_i}$ is bounded so that the particles are not wasting search time in infeasible region [9]. The quality of $\mathbf{X_i}$ is evaluated by a problem-dependent fitness function. If the current solution is better than the fitness of $\mathbf{P_g}$ or $\mathbf{P_i}$ or both, the new position value will replace $\mathbf{P_i}$ or $\mathbf{P_g}$ accordingly. This update process continues until stopping criterion is met, usually when either maximum iteration is achieved or target solution is attained. When the stopping criterion is satisfied, the best particle found so far is taken as the optimal solution (near optimal).

### *2.1. Neighbourhood*

PSO does not need a large population to look for the optimal solution successfully. Typical range of neighbourhood size is from 10 to 40 particles.

Originally PSO is a global neighbourhood; *gBest,* algorithm, where all particles in the swarm are fully connected. However, it is believed that this type of neighbourhood is often trapped in local optima and converged too fast [10]. Hence, *lBest* topology is adopted where the size of the neighbourhood is smaller. The information on the best solution found in *lBest* topology travels slower [11]. Examples of *lBest* topologies are; circle/ring, wheel and random neighbourhood [12]. In circle topology the particles are connected to their *K* immediate neighbours, with the smallest *K* is two. The wheel topology has a focal point where all particles are connected to it. Thus, the neighbour of non-focal point particles is just the focal point, while focal point particle has the entire swarm as its neighbours. As for random neighbourhood the neighbours of a particle are randomly selected.

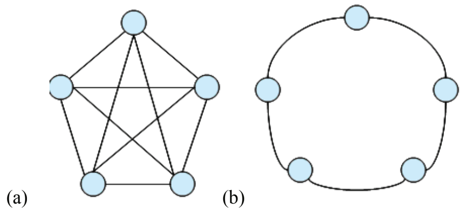


Fig. 1. (a)*gBest* topology (b)*lBest* topology; ring with *K*=2

## 2.2. Synchronous Update

In the original PSO, particles' $\mathbf{P}_i$ and $\mathbf{P}_g$ update are done after all particles' fitness has been evaluated, Fig. 2. This is known as synchronous update. The synchronous update ensures all particles, perfect and complete information of its neighborhood. According to study conducted by Vilela, et. al. [13], this is the best type of update process which gives good result and convergence speed. However, synchronous update is a costly choice [14], as a particle need to wait for the whole swarm to be updated before it can move to new position and continue its search. Hence the first particle evaluated is idle for the longest time, waiting for the whole swarm to be updated.

```
Initialize particles population;
Do{
  Calculate fitness values of each particles using fitness function;
  Update $p_{id}$ if the current fitness value is better than $p_{id}$;
  Determine $p_{gd}$ : choose the particle position with the best fitness value of all the neighbours as the $p_{gd}$;
  For each particle {
          Update velocity and position;
  }
} While maximum iteration or ideal fitness is not attained;
```

Fig. 2. Synchronous PSO (SPSO) Algorithm

## 2.3. Asynchronous Update

Asynchronous PSO (APSO) was first discussed in [14]. In APSO a particle's $\mathbf{P}_i$, $\mathbf{P}_g$, velocity and position are updated immediately after its fitness evaluation, Fig. 3. Therefore in this version a particle's search is guided by partial or imperfect information of its neighbourhood – some of the information is from current iteration while other from the previous iteration, this contributes to diversity in the swarm, which is a desirable trait [15]. Due to the asynchronous update, even in *gBest* topology, the value of $\mathbf{P}_g$ use by each particle in the same iteration can be varied. Other than diversity, lack of synchronicity in APSO solves the issue of idle particles faced in SPSO [16]. In several studies [14, 15, 17], APSO is claim to perform better than SPSO. Mussi, et. al. [15] studied the performance of both SPSO and APSO in global and local neighbourhood topology. It is shown in their work that in *gBest* topology, APSO outperform SPSO, while in *lBest,* the performance of both version of PSO algorithms closely match each other. This is because diversity is preserved in *lBest*. Xue, et. al., reported in their work that asynchronous update contributes to a lesser execution time [17]. The differences between APSO and SPSO are highlighted in Table 1. Asynchronous update enable the particles update sequence to change dynamically or a particle to be updated more that once [18, 16].

```
Initialize particles population;
Do{
  For each particle {
        Calculate particle's fitness values using fitness function;
        Update $p_{id}$ if the current fitness value is better than $p_{id}$;
        Update $p_{gd}$ : if the current fitness value is better than $p_{gd}$;
        Update velocity and position;
  }
} While maximum iteration or ideal fitness is not attained;
```

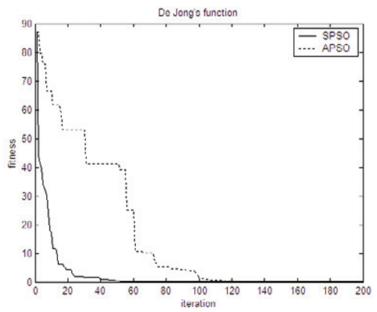
Fig. 3. Asynchronous PSO (APSO) Algorithm

Table 1. SPSO vs APSO

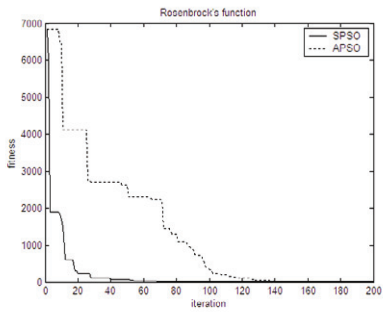
| | SPSO | APSO |
|---|---|---|
| Update | Synchronous – at the beginning of each iteration update all particle | Asynchronous – update a particle as soon as its fitness is evaluated |
| Information | Complete information – all particles use information from the same iteration | Partial – particles use combination of information from current and previous iteration |
| gBest value | uniform – all particles are updated based on the best of previous iteration | varying – particles use information available at the moment, better diversity |

Table 2. Test Functions

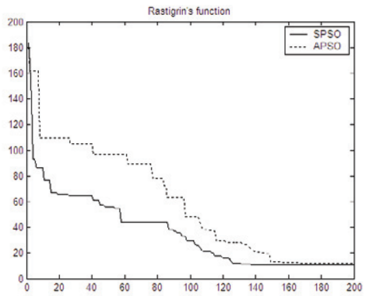
| | |
|---|---|
| De Jong's | $f(x)=\sum_{d=1}^{n} x_{id}^2$ |
| Rosenbrock's | $f(x)=\sum_{d=1}^{n}\left[100\left(x_{id+1}-x_{id}^2\right)^2+\left(1-x_{id}\right)^2\right]$ |
| Rastigrin's | $f(x)=10n+\sum_{d=1}^{n}\left[x_{id}^2-10\cos(2\pi\, x_{id})\right]$ |



(a)



(b)



(c)

Fig. 3. Result of (a) De Jong's function (b) Rosenbrock's function and (c) Rastigrin's function

## 3. Results

In this section the performance of APSO is compare with SPSO using three test functions. The test functions used are listed in Table 2. De Jong's and Rosenbrock's are unimodal functions whereas Rastigrin is a multimodal function. The functions dimension used is 10 (n=10). The algorithms are implemented using MATLAB. The number of particles in both SPSO and APSO is set to 10, while number of iteration is 200.

The results of the test are shown in Fig. 3. The results show that the performance of both version of particles update; SPSO and APSO, closely match each other. This proves that asynchronous update is an option of particles update process without compromising the performance of PSO. However the convergence rate of APSO is slower than SPSO. This is due to the higher diversity in APSO, which allow the particles to carefully explore the search space before convergence. This feature is not a disadvantage; rather it is a desirable trait as premature convergence is one of the problems faced by PSO [10].

## 4. Asynchronous PSO and Swarm Robotics Search Problem

A swarm robotics search problem's objective is to guide the swarm of robots to look for a target or multiple targets in a search area. According to Hereford and Siebold [19], swarm robotics search problem requires a search algorithm with the following characteristics;

- Distributed – each robots has its own processor to execute the algorithm on its own, thus a distributed algorithm ensures a single robot failure won't interrupt the whole swarm.
- Computationally simple – the robots are equipped with simple processor with limited memory and battery capacity, therefore the algorithm shouldn't be too complex
- Scalable – the swarm size varies from a small number of robots to a large swarm, the algorithm should be able to handle every cases.
- Contiguous movement – the robots have to move continuously until the target is reached.

Based on these characteristics, PSO is a suitable algorithm for the search problem. In [2], the authors had mapped the key elements of swarm robotics search to PSO, Table 3. According to this the robots act as particles and their fitness is evaluated based on the strength of the signal from the target to the robots. In original PSO, neighbourhood is usually predefined, but in swarm robotics the neighbourhood is based on the relative distance of the robots to each other.

Table 3. Mapping swarm robotics search to PSO

| Swarm Robotics Search | | PSO |
|---|---|---|
| Robots | ➡ | Particles |
| Signal detection | | Fitness evaluation |
| Relative localization | | Absolute localization |
| Path planning | | Particle update |
| Continuous control | | Iteration |
| Local communication – based on physical distance | | Global communication – based on fixed neighborhood |

In this study, it is believed that the PSO with asynchronous update is a better choice for swarm robotics in contrast to synchronous update. Asynchronous allows efficient usage of robots processor so that they won't be idle for a long time, therefore the robots can move continuously until target is found. It is also a good candidate for parallel implementation. Asynchronous parallel PSO (APPSO) [20], is chosen for swarm robotics search in [2&17]. APPSO allows the robots to efficiently utilize their individual processing power by searching for target asynchronously and simultaneously. APPSO allows the particles (i.e., robots) to evaluate their own fitness individually without the need to wait for others - parallelization. The velocity and position update in APPSO is a centralized process; the robots report their fitness to a master which performs the update process asynchronously. A centralized update does not use the robots processing capability to the fullest, hence a decentralized asynchronous PSO is proposed in [21]. This algorithm allows each particle to update their own velocity and position, making it more robust and scalable algorithm even in the event of failure among the swarm members. This algorithm also enables the swarm to have dynamic neighbourhood and variety of search tendency among the particles.

A physically embedded PSO (pePSO) is proposed in [19]. In pePSO, each robots acts as a particle where the robot movement is calculated based on PSO velocity and position equations. It is found that larger number of robots increases the number of successful search with lesser search time. In [22] a distributed PSO (dPSO) is introduced. The algorithm introduced uses asynchronous update, according to the author, this approach prevents the robots to stall during the search for target.

A work conducted by Doctor, et. al., [23], focuses on finding optimal PSO's parameters in collective search. Augmented Lagrangian PSO with velocity limits (VL-ALPSO) is discussed in [24], the algorithm is a decentralized and it takes the physical conditions of the search space into consideration

## 5. Conclusion

Target search is an important research problem in swarm robotics. A good search algorithm that is distributive, scalable, simple and allow robots' continuous movement is desired. PSO is suitable for this task due to its low computational cost and scalability. It is also easy to implement PSO in a distributed manner. However the synchronous nature of the original PSO does not ensure continuous and unhalted robots movement. Therefore this paper looks into the usage of asynchronous update in swarm robotics search. From the study conducted, it can be seen that asynchronous update is more suitable choice to ensure unhalted movement. In addition to that, asynchronous update is also suitable for parallel PSO implementation. Parallelization contributes to a more efficient usage of the robots processing capabilities.

## Acknowledgements

This project is funded by the Malaysian Ministry of Higher Education under its fundamental research grant scheme (FRGS). We would like to thank MOHE for their support. The authors also want to thank the reviewers for their invaluable reviews and comments.

## References

[1] Beni, G. 2004 "From Swarm Intelligence to Swarm Robotics, In Proceedings of International Conference on Swarm Robotics, pp.: 1-9

[2] Xue, S., Li, J.,Zeng, J., He, X., and Zhang, G. 2011. Synchronous and Asynchronous Communication Modes for Swarm Robotic Search, in *"Mobile Robots – Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training"*, J. Bedkowski, Editor, Intech (http://www.intechopen.com/)

[3] Kennedy, J., and Eberhart, R. C., 1995. "Particle swarm optimization", In Proceedings of IEEE International Conference on Neural Networks. pp.:1942-1948.

[4] Shi, Y., and Eberhart, R. C. 1998. "Parameter selection in particle swarm optimization", In Proceedings of the 7th International Conference on Evolutionary Programming pp.: 591 – 600

[5] Kennedy, J., Eberhart, R. and Shi, Y. 2001. Swarm Intelligence, Morgan Kaufmann, US.

[6] Shi, Y., and Eberhart, R. C., 1998. "A modified particle swarm optimizer", In Proceedings of IEEE International Conference on Evolutionary Computation. pp.: 69-73.

[7] Shi, Y., and Eberhart, R.C. 1999. "Empirical Study of Particle Swarm Optimization", In Proceedings of the IEEE Congress on Evolutionary Computation, pp.:1945-1950

[8] Kennedy, J. 2005 "Why Does It Need Velocity?", In Proceedings of Swarm Intelligence Symposium. pp.: 38 – 44

[9] Eberhart, R.C. and Shi, Y. 2000. "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", In Proceedings of the Congress on Evolutionary Computation Vol. 1. pp.: 84-88

[10] Shi, Y. 2004 Particle Swarm Optimization, IEEE Neural Networks Society, pp.8-13

[11] Vesterstrom, J. and Riget, J. 2002. Particle Swarms: Extension for improved local, multi-modal and dynamic search in numerical optimization, Master Thesis submitted at the Department of Computer Science, University of Aarhus

[12] Kennedy, J. 1999 "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance" In proceedings of the Conference on Evolutionary Computation. pp.:1931-1938

[13] Vilela, J.R., Zhang, M. and Seah, W. 2011 "A Performance Study on Synchronous and Asynchronous Updates in Particle Swarm Optimization" In GECCO'11, pp.:21-28

[14] Carlisle, A. and Dozier, G. 2001. "An off-the-shelf PSO", In Workshop on Particle Swarm Optimization, pp.:1-6

[15] Mussi, L., Cagnoni, S., and Daolio 2009. "Empirical Assessment of the Effects of update synchronization in Particle Swarm Optimization", In Proceeding of the AI*IA Workshop on Complexity, Evolution and Emergent Intelligence, pp.:1-10

[16] Vilela, J.R., Zhang, M. and Seah, W. 2011. "Random Asynchronous PSO", In Proceeding of 5th International Conference on Automation, Robotics and Applications, pp.: 220-225

[17] Xue, S., Zeng, J., and Zhang, J. 2009. Parallel Asynchronous Control Strategy for Target Search with Swarm Robots, International Journal of Bio-Inspired Computation, Vol. 1, No. 1, pp.:151-163

[18] Diosan, L. and Oltean, M. 2008. What Else is the Evolution of PSO Telling Us? Journal of Artificial Evolution and Application, Vol. 2008, 12 pages

[19] Hereford, J.M., and Siebold, M.A. 2010. Bio-Inspired Search Strategies for Robot Swarms, in *"Swarm Robotics from Biology to Robotics"*, E.M. Martin, Editor, Intech (http://www.intechopen.com/)

[20] Venter, G. and Sobieski, J.S. 2005 A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluation, Journal of Aerospace Computing, Information and Communication, no. 3, pp.:123-137

[21] Akat, S.B., and Gazi, V., 2008. "Decentralized Asynchronous Particle Swarm Optimization", In Proceedings of IEEE Swarm Intelligence Symposium, pp.:1-8

[22] Hereford, J.M 2006. "A Distributed Particle Swarm Optimization Algorithm for Swam Robotic Applications*",* In Proceedings of IEEE Congress on Evolutionary Computation, pp.:1678-1685

[23] Doctor, S., Venayagamoorthy, G.K, and Gudise, V.G, 2004. "Optimal PSO for Collective Robotic Search Applications", In Proceedings of IEEE Congress on Evolutionary Computation, pp.:1390-1395

[24] Tang, Q. and Eberhard, P. 2011 A PSO Based Algorithm Designed for a Swarm of Mobile Robots, Struct. Multidisc Optim. 44, pp.:483-498