

# Trained Particle Swarm Optimization for Ad-Hoc Collaborative Computing Networks

Shahin Gheitanchi, Falah Ali, Elias Stipidis

**Abstract.** Distributed processing is an essential part of collaborative computing techniques over ad-hoc networks. In this paper, a generalized particle swarm optimization (PSO) model for communication networks is introduced. A modified version of PSO, called trained PSO (TPSO), consisting of distributed particles that are adapted to reduce traffic and computational overhead of the optimization process is proposed. The TPSO technique is used to find the node with the highest processing load in an ad-hoc collaborative computing system. The simulation results show that the TPSO algorithm significantly reduces the traffic overhead, computation complexity and convergence time of particles, in comparison to the PSO.

## 1 INTRODUCTION

Deployment of wireless communication services based on collaborative computing has recently been considered by researchers in different areas [1-3]. Collaborative computing requires ad-hoc networking and efficient distributed processing techniques to perform complex tasks in a reasonable time.

Particle Swarm Optimization (PSO) [4] is a well known pervasive optimization technique in artificial intelligence that is based on behavior of social animals such as bees and birds. In the PSO technique each individual member of social colony, like a bird, is called a particle. For example, we observe PSO in a swarm of birds in a field. Their goal is to find the location with the highest density of food. Without any prior knowledge of the field, the birds begin their search in random locations with random velocities. Each bird can remember the locations that it has found more food, and somehow knows the locations where the other birds found large amount of food. Each bird has the choice to decide between returning to the location where it had found the most food itself, or exploring the location reported by others to have the most food, the bird accelerates in both directions somewhere between the two points depending on whether nostalgia or social influence dominates its decision. Along the way, a bird might find a place with more food than it had found previously. It would then head to this new location as well as the location of the most food found by the whole swarm. Occasionally, one bird may fly over a place with more food than have been found by any other bird in the swarm. The whole swarm would then head toward that location in addition to their discovery. Soon, all or most of the birds gather around the location where the highest density of food is there.

To increase the efficiency and performance of different OSI layers [5], the PSO technique has been used in the literature for

various scenarios [6–10]. Most of PSO applications in communications have been focused on clustering in ad-hoc networks aiming to minimize energy consumption [6-8]. In [6] the authors have applied PSO to cluster head selection and in [7] it has been used for distance based clustering of wireless sensor networks. Also in [8] the algorithm was used to optimize the cost and coverage of clustering in mobile ad-hoc networks. Many other applications for PSO in communications such as IP multicasting and channel assignment have been mentioned in [9]. Utilizing the PSO in ad-hoc networks increases flexibility, adaptation and robustness. While being simple, it can also introduce enormous traffic and computation overhead to the network and may lead to long convergence time. The traffic overhead is the number of extra packets needed to be transmitted over the network to accomplish the optimization task. The computation complexity overhead is the time (number of iterations) needed by a node to process the particles gathered over it.

In this paper, we introduce a generalized PSO model for communication networks. Then, based on the PSO model, we propose trained PSO (TPSO) for ad-hoc networks as a new technique to support collaborative computing networks. Using the proposed models, we simulate PSO and TPSO techniques in an ad-hoc network to find the node with the highest processing load. Finally, we compare the traffic overhead, computation complexity overhead and convergence time of the techniques.

## 2 GENERALIZED PSO MODEL

PSO system model consists of  $P$  number of particles and unique particle IDs (PIDs) which are randomly distributed over the problem solution space. The solution space,  $S$ , is the set of all possible solutions for the optimization problem. Depending on the problem, the solution space can have  $N$  number of dimensions,  $S^N$ , where each dimension contains different number of elements. Each particle is capable of measuring the suitability of solutions by using the fitness function  $f(s^1, s^2, \dots, s^n)$ , where  $0 < n \leq N$  and  $s^n \in S^N$ . All particles use unique and identical fitness function to be able to assess the suitability of a solution. The objective of the optimization is to find a set of  $\hat{S} \subset S$  to maximize the fitness function

$$\hat{S} = \text{Argmax } f(s^1, s^2, \dots, s^n) \quad (1)$$

Each particle stores the value and location of the best solution found so far, called the local best (LB). Also all particles are aware of the value and location of the best solution found by all other particles, called the global best (GB). Assuming synchronized timing and unique speed among the particles, the optimization is performed during  $\Gamma$  iterations. At each iteration the particles compare the LB and the GB to choose a direction independently based on the distance differences from current location to the GB and to the LB locations. The distance between two locations,  $(s_1^1, s_2^1)$  and  $(s_1^2, s_2^2)$ , for  $N = 2$  is given by

$$d = \sqrt{(s_1^2 - s_1^1)^2 + (s_2^2 - s_2^1)^2} \quad (2)$$

Particles consider nostalgia,  $w_n$ , and social influence,  $w_s$ , for deciding their directions. The weights,  $w_n$  and  $w_s$ , describe the importance of nostalgia and social influence for particles, where  $w_n + w_s = 1$ . We define the following expression for deciding the direction

$$\text{direction is } \begin{cases} \text{LB} & \text{if } (w_n d_{LB} - w_s d_{GB}) \leq 0 \\ \text{GB} & \text{if } (w_n d_{LB} - w_s d_{GB}) > 0 \end{cases} \quad (3)$$

Where, for each particle,  $d_{LB}$  is the distance from the current location to the LB and  $d_{GB}$  is the distance from the current location to the GB. After specifying the direction the particle moves toward the decided destination which is the location of the LB or the GB. During the optimization process, the GB is updated when a solution with higher fitness value is found by a particle. After  $\Gamma$  iterations the particles gather (or converge) on the location with the highest fitness value and the algorithm terminates which is referred to as the termination criteria 1. When the particles converge, the value of the GB is considered as the solution of the optimization problem. To avoid infinite loop in cases of having more than one GB value and also managing the execution time of the PSO algorithm, we set a relatively large number as the maximum iteration number,  $\Gamma_{\max}$  in compare with elements of solution space.

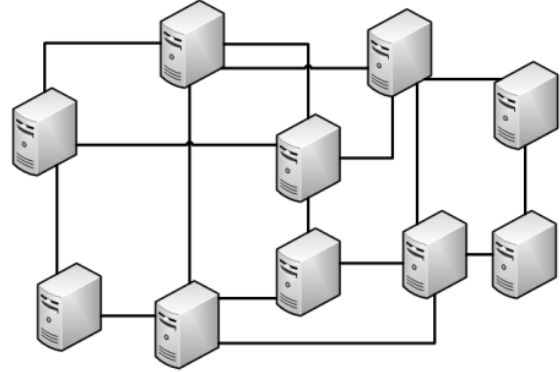
When the process is stopped by reaching  $\Gamma_{\max}$ , called termination criteria 2, the GB with highest population of particles over it is chosen as the solution for the optimization problem. Table 1 shows the general PSO algorithm.

**Table 1.** The generalized PSO algorithm

1: Initialize and distribute particles
2: Loop while not (termination criteria 1 and 2)
3: For each particle:
4:     If (LB > GB)
5:         Replace GB and LB
6:     Calculate LB
7:     Decide the direction towards LB or GB
8:     Move to new position
9: End of for
10: End of loop

### 3 TPSO FOR AD-HOC COLLABORATIVE COMPUTING

Because of distributed nature of particles, the proposed PSO model is suitable for efficient distributed processing with different objectives. Figure 1 shows the distributed nodes of an ad-hoc collaborative computing system. The movement of particles in an ad-hoc network introduces high traffic and computation complexity overhead. Also it may take a long time to converge. The traffic overhead is caused by movement of particles and their related information such as history of the LB. To reduce the overheads, we introduce the TPSO technique. TPSO is an improved PSO algorithm which the values of  $w_n$ ,  $w_s$  and  $P$  are defined based on the requirements of the system using a training system. The idea is to adapt the particles behavior in order to benefit from the system limitations such as limited number of routes that a particle can take or the number of particles that can be over a single solution (node). We assume a perfect training system is responsible for training the particles of the PSO algorithm. As it will be shown in the simulation results, training the particles reduces the traffic and computational complexity overheads and also significant reduces the convergence time.



**Figure 1:** Ad-hoc collaborative computing network model.

In the collaborative computing system we consider an ad-hoc network consisting of randomly distributed nodes in a two-dimension ( $N = 2$ ),  $X$ - $Y$ , landscape. The nodes in the network support multi-connections to their neighboring nodes and are able to transmit the data to the destination using shortest path routing method. The solution space is equal to positions of nodes and is stored in a sparse matrix as following

$$S_{x,y} = \begin{cases} 1 & \text{if a node exist in } x \in X, y \in Y \\ 0 & \text{else} \end{cases} \quad (4)$$

The distance is measured using eq. 2 and the number of particles is less than the number of nodes. As it will be shown in the simulation results, the number of particles in PSO contributes to the convergence time of PSO algorithm but in TPSO the convergence time does not significantly change by the number of

particles. However, low number of particles in the system may lead to sub-optimal results.

At the beginning of the TPSO process, the particles are randomly distributed among the nodes. In the network, the packets move only through the single available route between each two neighboring nodes. Since the solution space is equal to the position of nodes, and is a sparse matrix, it is not expected to find any solution between two neighboring nodes. Therefore, the movement of particles between two neighboring nodes that is caused by uncertainty between nostalgia and social influence will not lead to finding of a new LB or GB value. By manual training, we force the particles to always follow the social influence (choosing the GB as the next destination) using the following configuration:  $w_s = 1$ ,  $w_n = 0$ . This configuration will avoid redundant movements of the particles between two neighboring nodes; hence it will reduce the traffic and computation complexity overhead. Furthermore, as it will be shown in the simulation results, the particles converge in a constant number of iterations. This is because of constant maximum distance between every two nodes, movement of all particles towards an identical node and hopping from one node to another in a single iteration.

Figure 2 shows the flowchart of the TPSO algorithm for an ad-hoc collaborative computing network. The particles are implemented on each node using an identical software agent, called the particle process (PP). It is responsible to calculate, compare and update the LB and the GB values as well as moving the particle towards GB. The GB updates are done using a broadcast algorithm in the network layer. Since the updating is performed occasionally, we neglect the caused overheads. The PP of a node runs only when at least one particle is over that node. Therefore, increasing number of particles over a node will increase the computational complexity overhead. Particles move between two nodes by using a flag, carried by the data packets circulating in the network. The flag indicates when to run a PP process in a node and also is used for counting the PIDs over a node. Since particles move among the nodes using the data packets, their movement and directions depend on the availability of connection among the nodes.

In TPSO, all particles on a node have similar destination which is the GB location or the next hop towards the GB location. To further reduce the traffic overhead and computation complexity on a node, the particles are batched in single super particle which is the aggregation of all the particles on the node but with a new PID that is known to the PP processes. Then the super particle acts similar to the normal particle in the network and at each node it is treated as sum of particles when calculating the number of PIDs in PP. Using super particles will gradually reduce the number of particles in the system,  $P$ , as the TPSO process continues. The TPSO terminates when one of the termination criteria, explained in section 2, is met. Reducing the network traffic will reduce the computation overhead on each node by requiring fewer packets to be processed. Using the big-oh notation, we assume the computation complexity of the PSO on a single node is in order of  $O(g(\Gamma))$  where  $g(\Gamma)$  is the complexity function for  $\Gamma$  iterations on each node. The complexity will increase to  $O(Qg(\Gamma))$  when  $Q$  number of particles ( $Q < P$ ) overlap on the node. In TPSO, since  $Q$  and  $\Gamma$  are reduced by using the super particles, the PP will run

fewer times and computational complexity over a node will decrease.

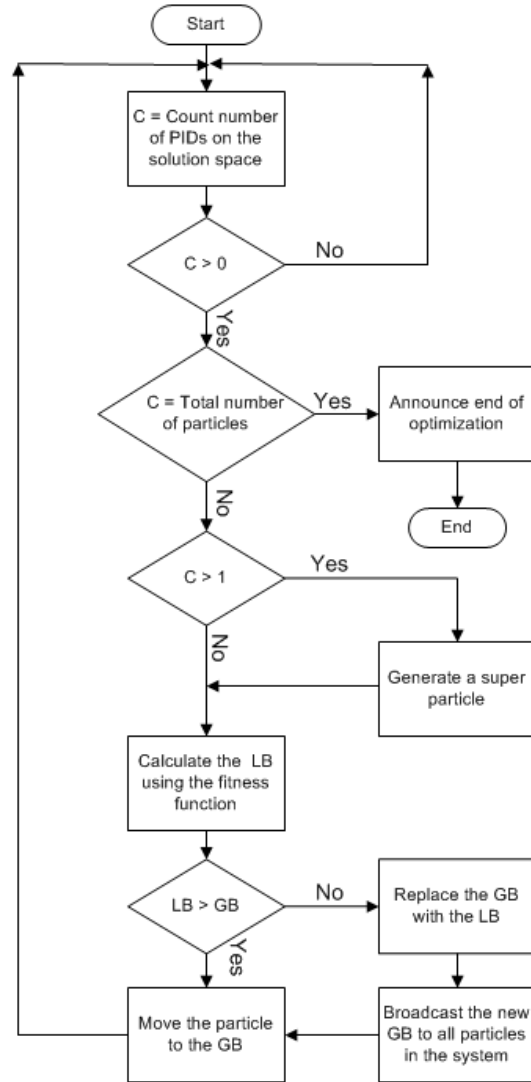


Figure 2: Flowchart of the TPSO algorithm for ad-hoc collaborative computing network.

## 4 SIMULATION RESULTS

In this section we consider an ad-hoc collaborative computing network with 50 nodes in a 100 by 100 landscape and 30 particles which are randomly distributed over the nodes. The nodes use shortest path routing algorithms for transmitting packets between the nodes. We simulate the proposed PSO and TPSO algorithms to find the node with the highest processing load. The results can be fed to different algorithms such as loading-based routing algorithm

[11]. The fitness function,  $f(x,y)$ , is equal to the load of a node in location of  $(x,y)$ . The load of a node is a measure of number of tasks (i.e. packets) that needs to be processed and we assume it remains constant during the optimization process. The processing load of a node in  $(x,y)$  with  $U$  number of task queues, is given by

$$f(x, y) = \sum_{u=1}^U L_{x,y,u} \quad (5)$$

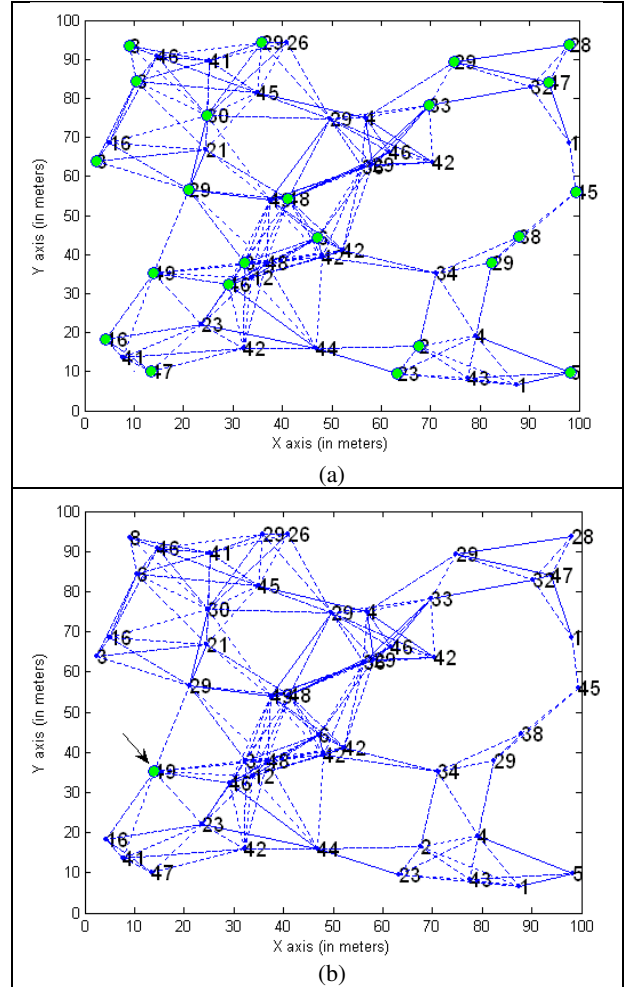
where  $L_{x,y,u}$  is the size of the  $u$ -th task queue. In Table 2 we introduce the packets used in the system Assuming that each data occupies only one byte, the size of packets for each packet type is calculated. We do not consider the overhead caused by the routing protocol of network layer.

**Table 2.**

PSO and TPSO packet sizes and description used in the simulation.

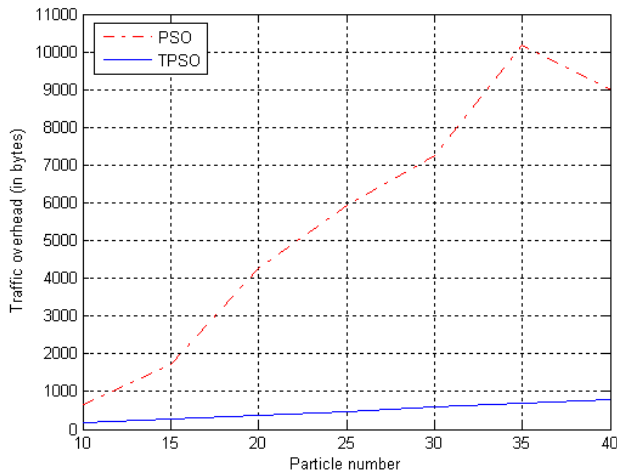
Packet type	Packet size in TPSO	Packet size in PSO	Description
GB_Broadcast	2	2	For broadcasting the value and location of the GB.
P_Move	1	4	For moving the particle from one node to another node. For PSO it contains PID, LB location and value and destination address. For the TPSO it only contains PID.
Terminate	1	1	A unique packet to indicate optimization termination

Figure 3 shows a snapshot of the proposed TPSO algorithm. The weights on each node represent the measure of processing load on that node and the distributed circles on the nodes show the particles. As the process progresses, the particles converge over the node with the highest load. Based on the termination criteria explained before, the algorithm broadcasts the found solution to the other nodes when all particles have converged over a node or maximum number of iterations has been reached.



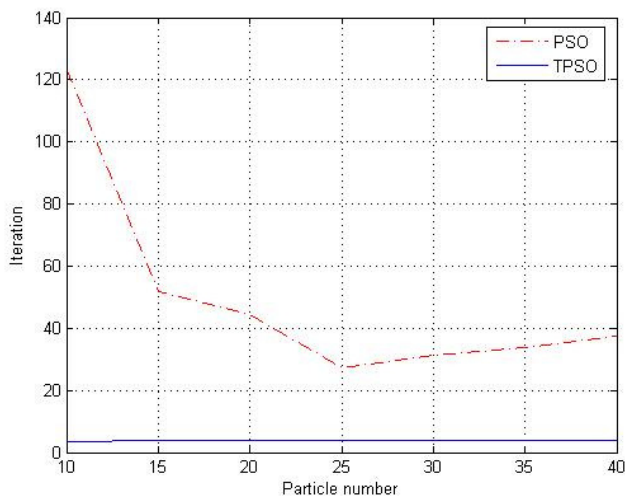
**Figure 3:** Snap shot of TPSO in ad-hoc collaborative computing network with 50 nodes and 30 particles showing particles a) before convergence, b) after convergence.

Figure 4 shows the difference of the average traffic overhead caused by P\_Move packets of PSO and TPSO in the mentioned ad-hoc network. The reduction of the traffic overhead is due to carriage of less data from node to node. As a result of training all the particles to always move towards the node with the GB value and do not return to LB location. However the GB location may change during the optimization process but at each interval all the particles in the system are heading towards the same destination which the current GB. As mentioned before, in TPSO, when there is more than one particle over a node, they are considered as one super particle. Since each super particle is treated similar to a particle, using super particles will reduce number of packets needed to be transmitted.



**Figure 4:** Comparison of PSO and TPSO traffic overheads for different number of particles over 50 distributed nodes.

In figure 5 we compare the average convergence time for PSO and TPSO based on our simulation results for different number of particles. As explained in section 3, when using TPSO the particles converge over the optimization solution with near constant number of iteration in comparison to PSO.



**Figure 5:** Comparison of convergence speed for TPSO and PSO techniques for different number of particles over 50 distributed nodes.

## 5 CONCLUSION

In this paper we introduced a generalized PSO algorithm for communication networks. Using the introduced PSO model, we proposed TPSO as an efficient optimization algorithm for ad-hoc collaborative computing networks. The PSO and TPSO techniques were simulated over distributed nodes to find the node with the

highest processing load. The simulation results show the convergence time of TPSO is almost constant while the traffic and computational complexity over a node is reduced in comparison to PSO.

## REFERENCES

- [1] S. Konomi, S. Inoue, T. Kobayashi, M. Tsuchida, M. Kitsuregawa, "Supporting Colocated Interactions using RFID and Social Network Display," IEEE Pervasive Computing, Jul.- Sep. 2006.
- [2] P. K. McKinley, C. Tang and A. P. Mani, "A Study of Adaptive Forward Error Correction for Wireless Collaborative Computing," IEEE Transaction on parallel and distributed systems, Vol. 13, No. 9, Sep. 2002.
- [3] C. Lai, W. Polak, "A Collaborative Approach to Stochastic Load Balancing with Networked Queues of Autonomous Service Clusters," MobCops Workshop of the 2nd IEEE Collaborative Computing Conference (CollaborateCom 2006), Atlanta, GA, USA, Nov. 2006.
- [4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc. IEEE Conf. Neural Networks IV, Piscataway, NJ, 1995.
- [5] A. S. Tanenbaum, "Computer Networks," Prentice Hall Ptr; 4 ed, 2002.
- [6] Tillett, J. Rao, R. Sahin, F., "Cluster-head identification in ad hoc sensor networks using particle swarm optimization," IEEE int. conf. on Personal Wireless Communications Proc., 2002, pp. 201- 205.
- [7] S.M. Guru, S.K. Halgamuge, S. Fernando, "Particle Swarm Optimisers for Cluster formation in Wireless Sensor Networks," International conference on intelligent sensors, sensor networks and information processing proc., 2005, pp. 219- 224.
- [8] X. Wu, S. Lei, J. Yang, X. Hui, J. Cho, S. Lee, "Swarm Based Sensor Deployment Optimization in Ad Hoc Sensor Networks," Int. conf. on embedded software and systems (ICESSE), 2005.
- [9] W. Ping, W. Guang-Zing, Z. Yang-Yang, "Particle Swarm Optimization Approach of Solving Communication Optimization Problems," Journal of Northeastern University, Natural Science (China). Vol. 25, no. 10, Oct. 2004, pp. 934-937.
- [10] S. Gheitanchi, F. H. Ali, E. Stipidis, "Particle Swarm Optimization for Resource Allocation in OFDMA," IEEE DSP07 conference proceeding, Jul. 2007, pp. 383-386.
- [11] H. Hassanein, A. Zhou, "Routing With Load Balancing in Wireless Ad hoc Networks", Proc. of 4th ACM international workshop on modeling, analysis and simulation of wireless and mobile systems, 2001.