Wireless Sensor Network, 2010, 2, 784-792
doi:10.4236/wsn.2010.210094 Published Online October 2010 (http://www.SciRP.org/journal/wsn/).

Improved Dynamic K-Coverage Algorithms in Mobile Sensor Networks

Roghayeh Soleimanzadeh[1], Bahareh J. Farahani[2], Mahmood Fathy[3]
[1]*Technical and Engineering Department, Azad University of Maragheh, Iran*
[2]*School of Electrical and Computer Engineering, Tehran University, Tehran, Iran*
[3]*School of Computer Engineering, Iran University of Science & Technology, Tehran, Iran*
Email: {*soleimanzadeh.r, farahani.ict*}*@gmail.com, mahfathy@iust.ac.ir*
Received May 26, 2010; *revised July* 5, 2010; *accepted August* 23, 2010

Abstract

In this paper, four PSO based distributed algorithms are presented to attain k-coverage in the target filed. In the first algorithm named K-Coverage Particle Swarm Optimization (KPSO), each static sensor which discovers an event in its sensing range, implements Particle Swarm Optimization (PSO) algorithm in a distributed manner on its mobile sensors. The calculation time is considered as a big bottleneck in PSO, so a second algorithm named K-Coverage Virtual Force directed Particle Swarm Optimization (KVFPSO) is presented, comprised of Virtual Force and KPSO algorithms. In the first and second proposed algorithms, the best experiences of the particles were used to determine their speed. It is possible that these responses might not be the final result and cause extra movements. Another algorithm named KVFPSO-Learning Automata (KVFPSO-LA) is introduced based on which the speed of particles is corrected by using the existing knowledge and the feedback from the actual implementation of the algorithm. To improve performance of the algorithm, Improved KVFPSO-LA is introduced, in which static sensors are equipped with learning automata. Simulation results show that the proposed protocols perform well with respect to balanced energy consumption among nodes, thus maximizing network life-time.

Keywords: Wireless Sensor Networks, K-Coverage, Event Coverage, Particle Swarm Optimization, Learning Automata

1. Introduction

Coverage is one of the most important issues in wireless sensor networks, which is concerned with how well a specified area is monitored by sensors. Degree of coverage is often used as a measurement of the quality of service of WSNs [1]. In surveillance and monitoring applications, it is usually required to have at least k sensors cover each point in the surveillance zone (k-coverage) [2]. The coverage levels for different applications may vary. For example, for the application of home security, where the environment is friendly, the coverage level can be set to a low value. On the other hand, having a high coverage level is always a major concern if sensors work in a hostile environment such as battlefields or chemically polluted areas. Even for the same application, the coverage level requirements may vary. For example, for forest fire detections, the coverage level may be low in rainy seasons, but high in dry seasons. Therefore, having a user-defined parameter k to represent the coverage level is sometimes a mandatory requirement in designing a network [3].

Sensor nodes usually work with small low-power batteries as the source of energy [4]; therefore, power efficiency is the main challenge in sensor network applications. One of the solutions of power efficiency is scheduling sleep for extraneous nodes. Hence, use of MAC protocols such as S-MAC [5] or T-MAC [6] that supports energy-saving and sleep-scheduling mechanisms is necessary.

In [2] a dense sensor network is used to attain K-cover-age in fields where an event has occurred. In [7] a hybrid WSN is considered where static sensors are deployed to detect events and mobile sensors are dispatched to event locations to conduct more advanced analysis. In [8], boundary conditions to have k-coverage in a mostly sleepy network in three different distributions are presented.

This article presents four PSO based algorithms. In the first algorithm, named KPSO the calculation time is considered as a big bottleneck, so a second algorithm named

KVFPSO is presented comprised of Virtual Force and KPSO algorithms and as the result, less energy is consumed.

The other algorithm named KVFPSO-LA is introduced, based on which the speed of particles is corrected by using the existing knowledge and the feedback from the actual implementation of the algorithm, in addition to using the law of virtual force. In this method, there will be less movement and re-location to achieve the desired result. Finally a new algorithm named Improved KVFPSO-LA is introduced, in which static sensors are equipped with learning automata. As the result, only the required numbers of mobile nodes are re-located.

2. System Model

In this paper, a mobile sensor network is a collection of the mobile nodes, which have mobility and are able to change their position based on networks dynamics or requirements. The system model is as followings:

- Mobile sensors are aware of their location by using locating algorithms [9].
- 1-coverage is set up by using the method mentioned in [10] with minimum number of sensors and using the least amount of energy in the concerned field. All mobile sensors used for 1-coverage will be mentioned as static sensors.
- The communication range of sensors is greater or equal to their sensing range. As the result, static sensors are connected to each other. ($R_c >= 2R_s$)
- Covering detection model in these algorithms is Binary Detection Model [11].
- Static sensors use to detect an event that has occurred in their sensing range and determine a coverage degree, based on the size of the event and environmental situations [3]. Then, based on the methods, which will be discussed later, they would increase coverage degree for that event to a determined extent.
- These protocols consist of five phases:

1-*Initialization Phase,* 2-*Proxy Sensor Selection Phase,* 3-*Creating Mobile Sensor List Phase,* 4-*Creating Particle List Phase,* 5-*Running the Algorithm Phase*

Phases 1-4 are in common within all proposed algorithms and only the last phase varies, which will be discussed in herewith.

3. Algorithms Implementation Phases

3.1. Initialization Phase

In this phase, all mobile sensors would broadcast one hello message in the network. Static sensors in their communication range, which have received this message which would in response, create and send a feedback message consisting of two fields: identification number of static sensor and the location of the static sensor.

3.2. BProxy Sensor Selection Phase

In this phase, the mobile sensors should select one among many static sensors, which have sent feedback message to them. Therefore, a proxization probability is determined for each static sensor, which will be calculated using Equation (1). Then, the mobile sensor will choose as its proxy the static sensor, which has the highest probability. If more than one sensor has equal probability, the proxy selection will be done based on order of receiving messages, sending to it a delegate proxy request consisting of three fields. These would include the mobile identification number, level of remaining energy and the location where in these fields are initialized in sequence with mobile sensor Mac address, mobile energy level and its location. From there on, all relocations of mobile sensors shall be done under their proxy sensors.

$$P(a_i) = 1 - d / \sum_{i=1}^{m} d_i$$
$$d = \textit{The distance between proxy node } a_i \textit{ and the mobile node} \quad (1)$$

In this equation, m refers to the number of static sensors which have sent a feedback message for the mobile. The distance between the static sensor and the mobile is calculated by using the Euclidean distance.

After all mobile sensors have selected a proxy for themselves; they go to the sleep mode to cut down on energy consumption using the method mentioned in [5] and [6]. If an event occurs in their proxy's sensing area or if selected as appropriate for relocating, as per a request of other static sensors from its proxy, its proxy will wake it up by sending an awakening message to it.

3.3. Creating Mobile Sensor List Phase

This phase will be started by any static node, which detects an event in its sensing range. When the static sensor detects an intruder in its sensing range, it will determine how many mobile sensors are needed according to the size and bigness of the intruder and environmental situations; meaning that in this phase it calculates the size of K. For example, it can be said that K is a coefficient of the size of the intruder, for instance it needs five mobile sensors (6-coverage) for a panzer and two extraneous mobile sensors (3-coverage) for a soldier.

After determining the size of K, the proxy sensor which has detected an event in its sensing range refers to mobiles sensors which represent them and evaluates whether it has enough number of mobile sensors. In this phase, the following might happen for the static sensor:

If the number of its mobile sensors is greater or equal to *K*, those of its sensors with less distance, compared to the sensing range with the event, are awakened with a message. The other sensors are put in a list named *the list of mobile sensors* and advances to the next phase.

Otherwise, the following might occur:

If the number of mobile sensors is less than *K*, like the previous phase, those of its sensors which have less distance than the sensing range to the event are awakened and the other sensors are put in *the list of mobile sensors*. It would then assess the degree of coverage, deduct it from *K*, and the final amount will determine the number of mobiles which should be taken from the neighbors. Subsequently, it would send a message of request to its neighboring static sensors, which would include ID number of the static sensor (proxy) and location of the event. The adjacent static sensors which receive the message would evaluate it and might undertake either of the following actions:

- If, there is no event in its sensing range and has no need for its mobile sensors, in a message to static sensor, the mobile requester would send the list of mobiles. This message would include its own ID number as well as that of its other mobiles and the location of its mobiles and the level of energy.
- If the neighboring sensor has also detected an event and its mobile sensors are engaged, then even if it has extraneous mobiles (*i.e.* the number of its mobiles are more than the number of *K*), it would give a negative reply to the requesting proxy. This is because there is the possibility that it again detects an event in the near future in its sensing range and might need its mobiles.

The requesting static sensor, given the amount of *K*, gives a positive response to those static sensors, which fulfill its need. Therefore, it takes the specifications of its mobile sensors and puts them in *the list of mobile sensors*. Thus, a list, including mobile sensors on which algorithm will be implemented, is created.

3.4. Creating Particle List Phase

Each static sensor, which has discovered an event in its sensing range, starts this phase after creating a list of mobile sensors, creating its particle list. In this section, a record will be attributed to each sensor existing in the mobile sensors' list, which would include ID fields, location and energy level of the mobile sensor. At the same time, an initial speed is by coincidence attributed to it.

3.5. Running the Algorithm Phase

3.5.1. K-Coverage Particle Swarm Optimization (KPSO) Algorithm

This algorithm is implemented in distributed form by static sensors, which have detected an event in their sensing range. In this algorithm, each static sensor, which detects an event in its sensing range, implements 1-4 phases and thus, creates its particle list. Then, it implements the PSO algorithm on its particles [11]. In this algorithm, the function fitness will be the distance between the mobiles and the location of the event. Implementing this algorithm will continue until maximum number of repetition is achieved or the determined degree of coverage is attained by the proxy sensors for the event.

3.5.2. K-Coverage Virtual Force Directed Particle Swarm Optimization (KVFPSO) Algorithm

In KPSO algorithm, although the desired result is achieved, but the calculation time in this algorithm is a big bottleneck. Also, the speed of particles in this algorithm is determined only by using the local and global best experiences. This is while, there is the possibility that these might not be the best final results and cause extra and repeated movements.

To counter this problem, the law of virtual force has been used to determine the speed of particles in this algorithm. To do this, the locations of events are considered regions with preferred coverage area and in a circular shape. (Regions with preferred coverage area are regions where mobile sensors are attracted towards these regions by a force of attraction). The center of the circle is presumed as the location of the event and the radius of the circle is presumed as the sensing range of the sensors. Regions with preferred coverage are shown with A_m and the force of attraction applied to the sensor moving towards the location of the event is calculated as the following [11]:

$$F_{iAm} = \begin{cases} \left(W_{Apre} P_{Am}, \propto_{iAm}\right) if & r < d_{iAm} - r_{Am} < C \\ 0 & otherwise \end{cases} \quad (2)$$

where r and c are the sensing range and communication range of mobile sensors. r_{Am} and p_{Am} are the radius and importance level parameter of the area of preferential coverage A_m , w_{Apre} is a measure of the attractive forces exerted by obstacles , d_{iAm} is the distance between s_i and A_m, α_{iAm} is the orientation of a line segment from s_i to A_m.

KVFPSO algorithm also is a distributed algorithm which is implemented by any static sensor which has detected an event in its sensing range. In this algorithm, the following equation is used to determine the speed of particles:

$$\begin{aligned} v_{ij}(t+1) = {} & w(t) * v_{ij}(t) \\ & + c_1 * r_{1i}(t) * (pbest_{ij}(t) - position_{ij}(t)) \\ & + c_2 * r_{2i}(t) * (gbest_{ij}(t) - position_{ij}(t)) \\ & + c_3 * r_{3i}(t) * g_{ij}(t) \end{aligned} \quad (3)$$

where $v_{ij}(t)$ and $position_{ij}(t)$ represent the position and velocity of ith particle in the jth dimension at time t. $pbest_{ij}(t)$ is the local best position of ith particle in jth

dimension and $gbest_{ij}(t)$ is the global best position. $r_{1i}(t)$ and $r_{2i}(t)$ are two separate random function in the range [0,1]. c_1, c_2 are learning factors. The velocity of particle in each dimension is limited to the $Vmax[i]$ which i is the index of dimension. c_3 is acceleration constant, $r_{3i}(t)$ is also a random function in the range [0,1] which is independent to $r_{1i}(t)$ and $r_{2i}(t)$, $g_{ij}(t)$ is the prolepsis motion suggested by virtual force of ith particle in jth dimension [11] , which is computed by:

$$g_{ij}(t)=\begin{cases}\dfrac{F_x^{(i,(j+1)/2)}}{F_{xy}^{(i,(j+1)/2)}}\times MaxStep\times e^{\frac{-1}{F_{xy}^{(i,(j+1)/2)}}} & j=odd\\ \dfrac{F_y^{(i,j/2)}}{F_{xy}^{(i,j/2)}}\times MaxStep\times e^{\frac{-1}{F_{xy}^{(i,j/2)}}} & j=even\end{cases} \quad (4)$$

where the superscript of each parameter presents the index of particles and the index of wireless sensor nodes which the virtual force exerts on, the subscript presents the coordinate of the virtual force. The correlative virtual forces are carried out by Equation (2).

In this algorithm, in addition to using the local best experience and the global best experience, the law of virtual force is also used to determine the speed of the particles. Adding this force to the speed of particles causes them to move towards the location of the event at a faster pace. As the result, the algorithm achieves result with less number of repetitions and mobiles move more objectively. Therefore, in this method, there will be less re-location to achieve the desired result, causing less energy consumption.

3.5.3. K-Coverage Virtual Force Directed Particle Swarm Optimization - Learning Automata (KVFPSO-LA) Algorithm

In the standard PSO algorithm, the responsibility to strike a balance between global and local search is with the median coefficient. This coefficient determines how much of the particle's current speed was used in determining its speed in the next phase. However, in [12], a new algorithm has been recommended which uses learning automata to regulate the method of searching for particles. It is the learning automata which determine in each step that particles continue in their current route or follow the best particles found so far. Using the learning automata has two advantages: First, the existing knowledge can be used to determine the trend of changes in the medial weight, and second, the trend can be corrected in the actual environment by using the feedback from implementation of algorithm.

Here, a new algorithm is introduced by the name of KVFPSO-LA to achieve K degree of coverage in regions where an event has occurred. This algorithm, like the other two algorithms are distributed algorithms which are implemented on static sensors which have detected an event in their sensing range. The implementation phase of this algorithm is such that after implementing the first 4 phases and creating its particles list, the KVFPSO-LA algorithm is implemented as the following:

In this algorithm, it is assumed that the static sensors are equipped with learning automata. The used LA has two actions which are "Follow the best" and "Continue your way". Until the desired goal is met or maximum number of iterations is done, the following steps should be repeated.

- LA selects one of its actions based on its probability vectors.
- Based on the selected action, the method of velocity updating for particles is selected and then the particles update their velocity and position.
- According to particles' updating results, a reinforcement signal is generated which will be used to update the probability vectors of learning automaton.

The selected action of LA in any iteration specifies the velocity updating method of particles for that iteration. Selecting "Follow the best" action means that just following the best self experience and best team experience has effect on the velocity of the particle in the next iteration and the current particle's velocity is ignored. In this case, the velocity update of the particles is done according to Equation (5). If the "Continue your way" action is selected, the new velocity of the particle will be equal to its current velocity.

$$\begin{aligned}v_{ij}(t+1) = {} & c_1 * r_{1i}(t) * (pbest_{ij}(t) - position_{ij}(t))\\ & + c_2 * r_{2i}(t) * (gbest_{ij}(t) - position_{ij}(t))\\ & + c_3 * r_{3i}(t) * g_{ij}(t)\end{aligned} \quad (5)$$

As a matter of fact, the selection of "Follow the best" action causes a local search around the best particle and selection of "Continue your way" action has the effect of doing global search and discovering new unknown parts of the search space. The used LA is responsible for learning probability distribution and balancing between local and global search. Evaluating the selected action is done by comparing each particle's new position with its old position. If a specific percent of population (C_{imp}) are improved, the selected action will be evaluated as positive and in the other cases it will be evaluated as a negative action.

Given that in this algorithm, the speed of particles is corrected by using the existing knowledge with the less number of repetitions, in addition to using the law of virtual force, this algorithm achieves the result with less number of repetitions and mobiles move more objectively. As a result, in this method, there will be less

re-location for achieving the desired result and therefore, there will be less energy consumption compared to previous algorithms.

3.5.4. Improved KVFPSO-LA Algorithm

In previous algorithms, the algorithm is implemented on all particles existing in the particles list, even if the number of these particles (mobile nodes) is more that the required amount of *K*. As a result, with implementation of this algorithm, more mobiles than required would move towards the location of the event and cause excessive use of energy.

To overcome this problem, it is assumed that each static sensor which detects an event in its sensing range is equipped with the learning automata and uses this learning automata to determine the particles (mobile nodes) on which the algorithm is implemented. It then moves towards the location of the event. The implementation phases are as such that the static sensor, which detects an event in its sensing range, implements phases 1-4 and creates its particles list. The static sensor then, given its particles list, creates a learning automaton whose number of actions is equal to the number of particles existing in the particles list. In fact, there is a one-to-one correspondence between the learning automata's actions and the particles existing in the list. In this phase, the learning automata will attribute the probability of an initial selection equal to 1/*r* (*r* is the number of particles in the list). Then, using the following equations, it will give reward or penalty for possible selection of its particles and thus, the most appropriate particles are selected for re-location towards the event. The implementation results show that by applying these equations, those mobiles, which have higher energy level and lesser distance from the event have the higher probability of being selected.

When the energy of the mobile we want to re-locate:

- is less than 50 percent of the average energy of other nodes, the selected node is penalized based on the 3β parameter which is calculated using the following mentioned parameter:

$$\beta = \lambda_2 + \psi_2 * [(y(AveEnergy - Enregylevel(a_i)) + d(a_i, l_j)) / (y \times AveEnergy + dist\ between\ event\ and\ farthest\ node\ to\ event)] \quad (6)$$

- is more than 50 percent and less than 80 percent of the average energy of other nodes, the selected node is penalized according to the following parameter:

$$V_P = (1 + \frac{\frac{Energylevel(a_i)}{AveEnergy} - 50}{30})\beta \quad (7)$$

- is more than 80 percent and less than 100 percent of the average energy of other nodes and the distance between this node from the destination, among other nodes, is minimum. In such case the node is rewarded based on parameter α.

$$\alpha = \lambda_1 + \psi_1 [(y \times Enregylevel(a_i) + dist\ between\ event\ and\ farthest\ node\ to\ event - d(a_i, l_j)) / (y \times AveEnergy + dist\ between\ event\ and\ farthest\ node\ to\ event)] \quad (8)$$

- is more than the average energy of other mobile nodes. In such case, the mobile node is awarded based on α parameter.

λ_1 and λ_2 respectively determine the minimum amount acceptable for the reward and penalty. Taking into account the scale difference between distance and energy, the *y* parameter is regulated in way that these two terms are placed in an almost equal scale and ψ_1 and ψ_2 are regulated in such a way that the amount of parameters α and β do not exceed a specific limit.

After the probability of selecting the mobile nodes is determined by using the above-mentioned equations, *k* particles (mobile nodes) which have the highest selection probability are selected for covering the event and then one learning automata is allocated to each of the selected particles. Each learning automata is in fact the core of the particles, which guides its movement within the search space. Allocation of learning automata to each of the particles causes each particle to make decisions for determining the type of its movement without considering the movement of other particles and by using the result of its current movement in the environment. Each learning automata has two actions of "follow the best" and "continue your way". When an automata allocated to a particle chooses the action of "follow the best", it means that the particle, according the Equation (5), moves towards the best location met by the group (*gbest*) and best location which has met so far (*pbest*) to move in the search space with the zero weight motion inertia. If the learning automata of each particle select the action of "continue your way", it would mean that the particles is moving within the space at a current velocity and will continue the current way [13].

The algorithm in this method is as the following:

1) The phase of creating the particles list, for selected particles, is implemented.

2) The probability range of selection action of each particle's learning automata is measured.

3) As long as the maximum number of steps are taken or the desired objective is achieved, phases 4 to 9 are repeated:

4) The learning automata related to each particle selects one of its actions according to the probability factor of its actions.

5) If the learning automata allocated to *i*th particle selects "follow the best", the speed of the particles is updated according to Formula (5).

6) If the learning automata allocated to the particle selects "continue your way", the speed of the particle will be equal to its previous speed.

7) Considering the selection action, the method of updating the speed of the particles is determined and then, particles will update their speed and location.

8) If the new location of the particle improves compared to its previous location, the particle will be awarded for the action it selected. Otherwise, the action will be penalized.

9) The action selection probability range of the learning automata is corrected.

Thus, the selected particles will move towards location of the event and achieve the required degree of coverage. Given that in this algorithm, only the required number of particles will be re-located and each particle will determine the type of its action in the next phase according to the result of its implementation in the previous phase, thus less energy will be consumed.

4. Experimental Results

In this section, the performance of our algorithms will be evaluated by simulations. A 40 m × 40 m rectangle sensing field will be set up, in which there are several mobile sensors randomly deployed. Each mobile sensor has an initial energy reserved for movement and the moving energy cost per meter is set to 8.27 *J*.

A number of these mobile nodes were used for creating 1-coverage and the rest are used for increasing coverage in case of the occurrence of an event. The sensing range of these sensors is equal to 5 m and their communication range 20 m. It will be assumed that there are 5 events in specific locations in the space, and determined the results of implementation for these 5 events for different coverage degrees. The parameters for virtual force are set as $WApre = 1$, $Maxstep = 0.2$ $r = 1$ m according to the discussion in [14]. The acceleration constants of PSO are set as $c1 = c2 = c3 = 1$ and λ_1 and λ_2 parameters are measured equal to 0.1. The numbers of used particles in all PSO algorithms are 10.

For calculating the level of energy consumed by each mobile node by one unit, the following-mentioned equation can be used, using the reference [7]:

$$W = \Delta_{move} \times d(a_i, l_j) \quad (9)$$

Δ_{move} is the energy required to move a sensor one-unit distance ($\Delta_{move} = 8.27$), $d(a_i, l_j)$ is the Euclidean distance from a_i's current position and point l_j is the point which mobile node will move there.

Figure 1 shows that for each one of the mentioned algorithm, it will achieve the result with less number of repetitions with an increase in the number of mobile nodes. By comparing the number of repetitions in all four algorithms, it can be noted that the KPSO algorithm has the highest number of repetitions. This is because the speed of particles and their location are determined only through the previous phase on following up the local and global best experiences. Given that there is the possibility of the local and global best experiences not achieving the final result, this cause the algorithm to achieve the result through higher number of repetitions. Comparing with KPSO algorithm, the KVFPSO algorithm has better performance and achieves the final result with less number of repetitions. This is because, as it was mentioned, the law of virtual force has been used to determine the speed of particles and so more objective-oriented and more organized particles move towards the location of event. The KVFPSO-LA algorithm achieves the best result compared to the two previous algorithms because in addition to using the law of virtual force, the speed of the particles is determined by taking into considering the

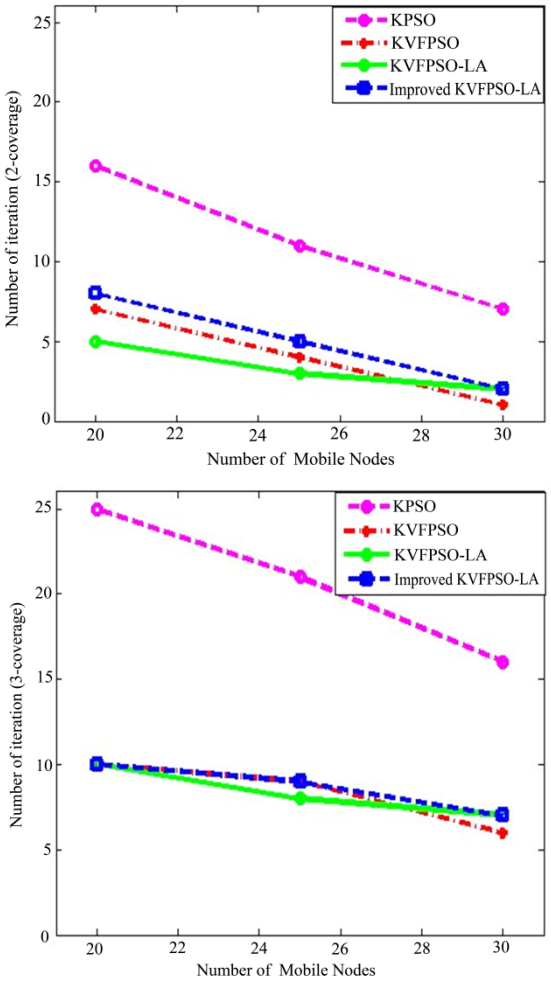

Figure 1. The number of algorithm iterations.

feedback from the actual implementation environment.

In the improved KVFPSO-LA algorithm, since algorithms acts on less number of particles, the number of repetitions is greater than or equal to KVFPSO and KVFPSO-LA algorithms.

Figure 2 shows a comparison of the number of mobile nodes that relocated to reach a certain degree of coverage. Number of these nodes will decrease when total number of mobile sensors increase; because the proxy node which detected the event in its sensing range has more mobile node to use and it has no need to make request to its neighbor proxy nodes. So the proxy node will send a message to the mobile nodes which have less distance with the event than the sensing range and wake up them to detect the event.

The KVFPSO algorithm, compared to KPSO, has less number of repetitions and as the result less number of re-located mobile nodes, due to the force of attraction that taken the mobiles towards the location of the event.

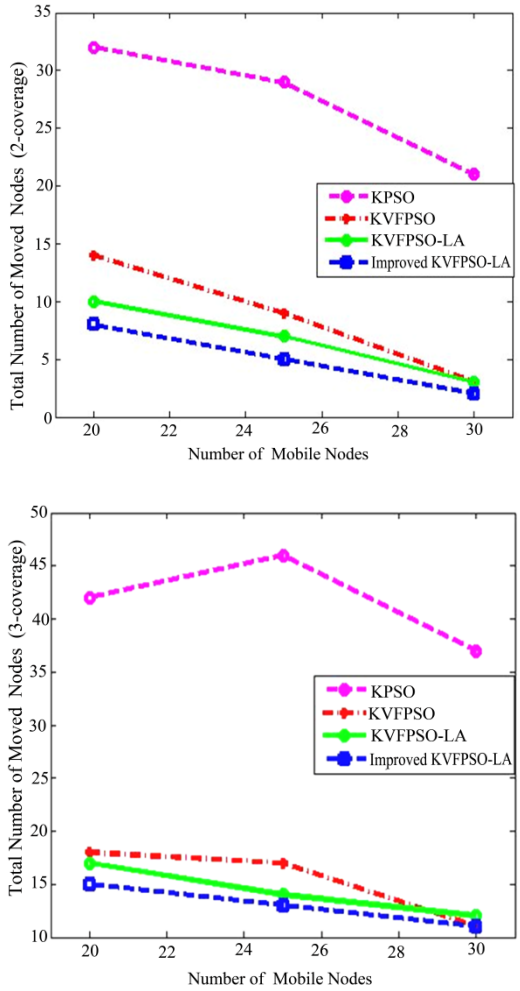

Figure 2. A comparison of the number of mobile nodes that relocated to reach a certain degree of coverage.

As it was mentioned before, the KVFPSO-LA algorithm, in comparison with the two previous algorithms, uses the result of the feedback from the movement of its particles in the actual environment, in addition to using the law of virtual force. Therefore, the result is achieving in less number of repetitions and less number of re-located nodes.

In the improved KVFPSO-LA algorithm, because it is implemented on the required number of particles, each particle determined the type of its movement in the next phase without considering of the movement of other particles, and only taking into account the result achieve from its own implementation in the previous phase. Therefore, the number of re-locations is less and particles move more objectively.

Figure 3 and **Figure 4** show the Average Moving Distance and Average Energy Consumption for each relocated mobile sensor, respectively. As the figures demonstrate, for a specific level of coverage, by increasing number of mobile nodes, Average Moving Distance and Aver-

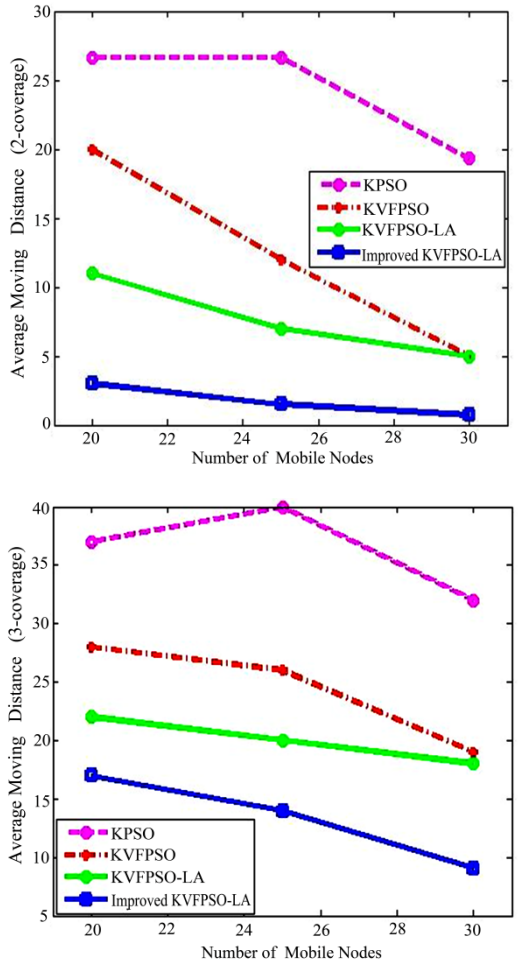

Figure 3. Average moving distance for each relocated mobile sensor.

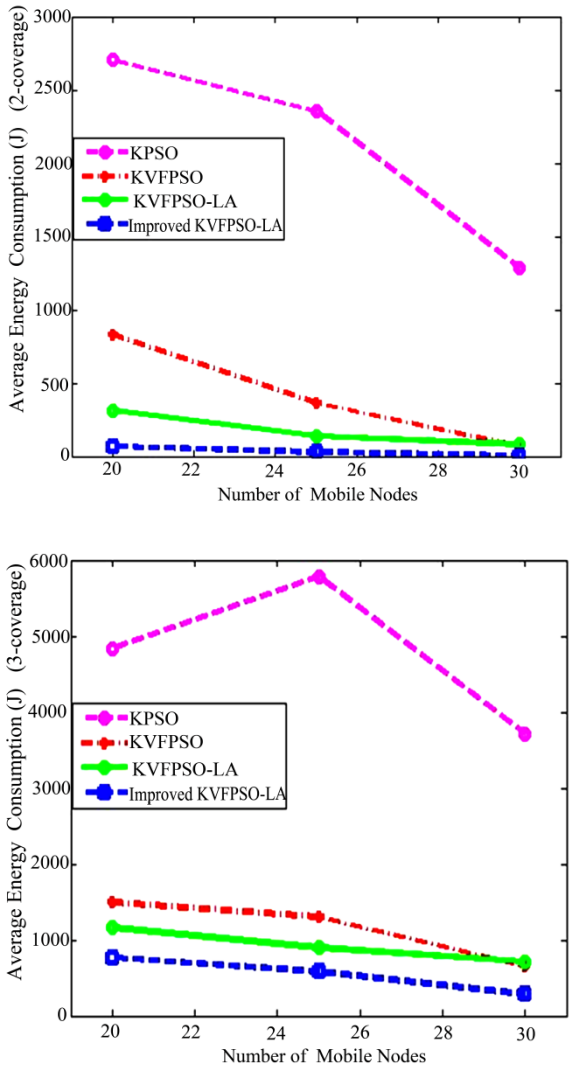

Figure 4. Average energy consumption for each relocated sensor.

age Energy Consumption decrease.

In the improved KVFPSO-LA algorithm, because it is implemented on the required number of particles, each particle will determine the type of its movement in the next phase according to the result of its own implementation in the previous phase. So, mobiles move within a shorter distance compared the other three algorithms.

By studying these four figures, it can be noted that by using the improved KVFPSO-LA and KVFPSO-LA algorithms, the mobiles have to travel within a shorter distance compared to other two algorithms and therefore, they consume less energy.

5. Conclusions

In this paper four PSO based algorithms to achieve *k*-coverage are proposed. In the KPSO, each static sensor which detects an event, implements PSO algorithm in a distributed manner on its mobile sensors. The calculation time is considered as a big bottleneck. So KVFPSO is proposed, comprised of Virtual Force and KPSO algorithms that resulted in less energy consumption. KVFPSO-LA is proposed based on which the speed of particles is corrected by using the existing knowledge and the feedback from the actual implementation of the algorithm, in addition to using the law of virtual force. As the result, this algorithm achieves the final result with less number of repetitions. Finally, Improved KVFPSO-LA is proposed, in which static sensors are equipped with learning automata. By using these automata only the required numbers of mobile nodes are re-located. The experiments demonstrated the performance of the four algorithms and the Improved KVFPSO-LA was best in most cases.

6. References

[1] J. P. Sheu, G. Y. Chang and Y. T. Chen, "A Novel Approach for K-Coverage Rate Evaluation and Re-Deployment in Wireless Sensor Networks," *IEEE Global Telecommunications Conference*, New Orleans, 2008, pp. 1-5.

[2] A. Yahyavi, L. Roostapour, R. Aslanzadeh and N. Yazdani, "DyKCo: Dynamic K-Coverage in Wireless Sensor Networks," *IEEE International Conference on Systems, Man and Cybernetics*, Singapore, 2008, pp. 2804-2809.

[3] Y. S. Li and S. Gao, "Designing K-Coverage Schedules in Wireless Sensor Networks," *Journal of Combinatorial Optimization*, Vol. 15, No. 2, 2008, pp. 127-146.

[4] F. Ye, G. Zhong, S. Lu and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks," *Proceeding of the* 10*th IEEE International Conference on Network Protocols*, Paris, 2002, pp. 200-201.

[5] W. Ye, J. Heidemann and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, Vol. 12, No. 3, 2004, pp. 493-506.

[6] T. V. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proceedings of the* 1*st International Conference on Embedded Networked Sensor Systems*, Los Angeles, 2003, pp. 171-180.

[7] Y. C. Wang, W. C. Peng, M. H. Chang and Y. C. Tseng, "Exploring Load-Balance to Dispatch Mobile Sensors in Wireless Sensor Networks," *Proceedings of* 16*th International Conference on Computer Communications and Networks*, Honolulu, 2007, pp. 669-674.

[8] S. Kumar, T. H. Lai and J. Balogh, "On K-Coverage in a Mostly Sleeping Sensor Network," *Wireless Networks*, Vol. 14, No. 3, 2006, pp. 277-294.

[9] N. Bulusu, J. Heidemann and D. Estrin, "GPS-Less Low-Cost Outdoor Localization for Very Small Devices," *IEEE*

Personal Communications, Vol. 7, No. 5, 2000, pp. 28-34.

[10] J. Xiao, L. L. Sun and S. Zhang, "Distance Optimization Based Coverage Control Algorithm in Mobile Sensor Network," *IEEE International Conference on Systems, Man and Cybernetics*, Singapore, 2008, pp. 3321-3325.

[11] X. Wang, S. Wang and J. J. Ma, "An Improved Co-Evolutionary Particle Swarm Optimization for Wireless Sensor Networks with Dynamic Deployment," *Sensors*, Vol. 7, No. 3, 2007, pp. 354-370.

[12] M. Sheybani and M. R. Meybodi, "PSO-LA: A New Model for Optimization," *Proceedings of* 12*th Annual CSI Computer Conference of Iran*, Shahid Beheshti University, Iran, 2007, pp. 1162-1169.

[13] M. Hamidi and M. R. Meybodi, "New Learning Automata Based Particle Swarm Optimization Algorithms," *Proceedings of the* 2*nd Iranian Data Mining Conference*, Amirkabir University of Technology, Iran, 2008.

[14] Y. Zou and K. Chakrabatry, "Sensor Deployment and Target Localization Based on Virtual Forces," 22*nd Annual Joint Conference of the IEEE Computer and Communications Societies*, San Franciso, Vol. 2, 2003, pp. 1293-1303.

