WIRELESS SENSOR NETWORKS COVERAGE-ENERGY ALGORITHMS BASED ON PARTICLE SWARM OPTIMIZATION

Nor Azlina Ab Aziz

Faculty of Engineering & Technology – Multimedia University
Melaka 75450 - Malaysia
azlina.aziz@mmu.edu.my

(Received December 2012 and Accepted May 2013)

In wireless sensor networks (WSN) coverage is an important aspect in measuring the quality of service of the network. Network with mobile WSN is able to improve its coverage by moving the sensor nodes so that a better arrangement is achieved. However sensors movement consumes energy. In this paper three algorithms are presented to optimize coverage of mobile WSN while at the same time taking energy usage into consideration. The algorithms proposed are based on particle swarm optimization (PSO). PSO is chosen due to its good performance record. Based on the tests conducted, the algorithms successfully achieve their objective.

1. INTRODUCTION

One of the most active research areas in wireless communication is the wireless sensor networks (WSN) field. WSN allow continuous and autonomous supply of information on a particular object or area of interest, therefore it is able to provide a more reliable, up to date and advanced information communication system. This is among the reasons why WSN has gained a lot of attention among researchers.

WSN consists of a group of sensor nodes working together to sense their environment, communicate wirelessly over a short distance and perform simple data processing [1]. These sensor nodes are typically small in size, battery powered, low cost, and deployed randomly and abundantly. Research had been conducted to enable the nodes to have their own locomotion capability [2]. The mobility can be obtained by mounting the nodes on mobile platform. Mobility is a very useful feature in WSN, where it can be used to harvest energy [6], to collect information using mobile based stations [7], to improve coverage [3,4,5] and to compensate for lack of sensors in providing enough coverage by constantly moving the sensors so that the chance of target detection is improved [8]. Coverage is an important issue in WSN as it is used to measure the quality of service of a network. A network unable to cover the environment efficiently is a network of poor quality. On the other hand, movement is the highest energy consumption task of sensor nodes, where it consumes more energy than communication, sensing and computation (the energy consumption reduces in this order) [2].

In this paper, algorithms focusing in improving mobile WSN coverage by repositioning the sensors after initial random placement are presented. The first algorithm focuses only in improving the coverage without taking into account the mobility cost, whereas the second algorithm maximizes coverage while at the same time ensure the energy usage is as small as possible. The third algorithm is for limited mobility WSN coverage optimization. The mobility is limited so that the amount of energy consumed for coverage improvement is below a threshold value.

All problems considered here are expressed as optimization problems and solved using particle swarm optimization (PSO) based algorithms. PSO is suitable for WSN due to its simplicity, low computational cost, fast convergence and high quality solution [9].

In section 2 PSO is introduced followed by a review on related works in section 3. The algorithms are presented in section 4, 5 and 6. Results and discussion are presented in section 7. This paper is concluded in the last section.

2. PARTICLE SWARM OPTIMIZATION

PSO was introduced in 1995 by James Kennedy and Russell Eberhart. It is a swarm based algorithm that
mimics the social behaviour of organisms like birds and fishes. The success of an individual in these communities is affected not only by its own effort but also by the information shared by its surrounding neighbours. This social behaviour is imitated by PSO using swarm of agents called particles [10]. Particles neighbourhood in PSO has 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.

A particle in PSO has a position ($X_i$) and velocity ($V_i$). The position represents a solution suggested by the particle while velocity is the rate of changes from current position to the next position. Initially 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:

$$V_i = w \times V_i + c_1 \times \text{rand}() \times (P_i - X) + c_2 \times \text{rand}() \times (P_g - X)$$  \hspace{1cm} (1)

$$X_i = X_i + V_i$$  \hspace{1cm} (2)

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

As what can be observed in Eq.1 velocity is influenced by; $P_i$ – the best position found so far by the particle and $P_g$ – the best position found by the neighbouring particles. $P_g$ could be either the position of gBest or lBest. The $V_i$ value is clamped to $\pm V_{\text{max}}$ to prevent explosion. If the value of $V_{\text{max}}$ is too large the exploration range is too wide however if it is too small particles will favour local search [11]. $c_1$ and $c_2$ are the learning factors to control the effect of the “best” factors of particles; $P_i$ and $P_g$ typically both are set to $2$ [12]. $\text{rand}()$ and $\text{rand}()$ are two independent random numbers in the range of $[0,0.1,0]$.

The randomness 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 avoid continuing to explore the wide search space and switch to fine tuning when a good area is found [13].

The particle position is updated using Eq.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 [14].

The quality of the solution is evaluated by a fitness function, which is a problem-dependent function. If the current solution is better than the fitness of $P_i$ or $P_g$ or both, the best value will be replaced by current solution 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). The PSO algorithm is presented in Fig. 1.

```
Initialize particles population:
Do{
    Calculate fitness values of each particles using fitness function;
    Update $P_{\text{id}}$ if the current fitness value is better than $P_{\text{id}}$;
    Determine $P_{\text{gd}}$: choose the particle position with the best fitness value of all the neighbors as the $P_{\text{gd}}$;
    For each particle {
        Calculate particle velocity according to (1);
        Update particle position according to (2);
    }
} While maximum iteration or ideal fitness is not attained;
```

Figure 1. PSO Algorithm

3. RELATED WORKS

The wireless sensors have restricted sensing range as well as limited battery capacity [15]. The limitations cause issues such as coverage and network lifetime. According to [16], coverage can be classified into three classes namely area coverage, point coverage and barrier coverage.

- **Area coverage** is on how to cover an area with the sensors. The objective here is to maximize the coverage percentage. Where the coverage percentage is defined as the ratio of area covered by at least one sensor to the total area of ROI. Coverage problem can also be seen as a minimization problem [17]. From the minimization point of view, the objective is to make sure the total area of the coverage holes in the network is as small as possible. Area coverage is the focus of this work.

- **Point coverage** is the coverage for a set of points of interest. This type of coverage concentrates on how to cover a set of targets or hotspots in an area, instead of the whole area as in area coverage.

- **Barrier coverage** is about covering the barrier of an area. The barrier coverage focuses on decreasing the probability of undetected penetration to a protected area. Therefore, the sensors need to be deployed along the area’s border.

In mobile WSN, initial coverage after random deployment can be enhanced by using incremental deployment or repositioning the sensors. In the incremental deployment, coverage holes are identified after initial deployment, using this information, new sensors are deployed to cover the holes. Shen et al.
[17] suggested a multi step, greedy exploration based algorithm to increase coverage using an incremental redeployment method. The algorithm divides the ROI into small grids and the coverage rate is measured as ratio of the number of grid points covered to the total number of the grid points in the ROI. If the coverage rate does not satisfy the expected condition, a new sensor will be deployed. After the deployment of the new sensor, the new coverage rate is measured. As long as the expected coverage value is not satisfied the incremental deployment process will continue.

As mention above, sensors repositioning can improved the coverage. This concept is used in [18]. The sensors are initially deployed randomly. Based on the initial placement, the sensors broadcast their locations and construct a Voronoi diagram. The sensors decide whether to reposition and eliminate (reduce) the coverage holes, or to stay at their current location, using this diagram. Three protocols were suggested:

1. **VECtor based algorithm (VEC)** – to push sensors out from a densely covered area,
2. **VORonoi based algorithm (VOR)** – a sensor moves to its farthest Voronoi vertex when it detects a coverage hole and
3. **Minimax** – to cover holes by moving closer to the farthest Voronoi vertex but not as far as VOR.

A work proposed in [5] also used Voronoi diagram to optimize the coverage of WSN. The WSN considered consist of static and mobile sensors. The static sensors construct a Voronoi diagram which is used to detect coverage holes, while the mobile sensors are moved to close these holes.

Howard and Poduri [3] proposed the virtual field concept to WSN. They assumed that the sensor nodes and obstacles have potential fields which exert virtual forces. These forces cause the nodes to repel each other, thus resulting in physical movement by the sensors. The movement continues until either the sensors sensing fields no longer overlap or they cannot detect each other. Based on the distance, the force is determined and used to initiate sensors movement. Although this method ensures full coverage and full connectivity, it relies highly on sensor mobility.

The works discussed above manipulate the actuation capability of the sensors. Even though mobility is an advantage to WSN, it is a high energy consuming task [2]. The wireless sensor nodes are normally battery powered, therefore due to the limited battery capacity energy conservation measures are important to prolong the WSN lifetime. Hence, movement planning is an important issue in WSN. It is desired to achieve optimal coverage and at the same time not to relax the mobility energy consumption issue due to the fact that sensors have a limited energy supply. Among the works done on coverage and sensors mobility is [4], other than improving the coverage, the algorithm proposed here takes into account the minimization of the travelled distance. Both objectives are achieved by trying to match the sensors to the best grid points which are evenly distributed in the ROI. An assumption where the sensors can only move as far as the allocated fuel resource allowed is also imposed.

A similar concept as used in [3] is proposed in [19]. However in contrast to [3], the force directed algorithm proposed by Zou and Chakrabarty reduced the physical movement to a single movement, which is done at the end of the algorithm execution. During the execution of the algorithm the sensors are virtually moved. The single movement reduces the energy consumed in repositioning of the sensors, in addition to this, another energy conserving method is also included in the algorithm which is limiting the distance of initial and final position.

The limited maximum distance moved by a sensor is considered in [20]. There are two objectives to be accomplished. The first objective is to evenly deploy the sensors throughout the ROI, and the second objective is to minimize the total number of movement. The problem is tackled by dividing the ROI into smaller regions, and the sensors are then divided evenly to each region by moving them from a dense region to a sparse region. The movement of sensors from one region to another is carefully planned so that the maximum distance travelled is minimized and does not exceed the maximum value.

As what can be observed from the works presented above, limiting the maximum distance moved is the common method chosen. This kind of WSN is known as limited mobility WSN [21].

The mobile WSN problem has also been approached using the load balancing method [22]. The load balancing method is used to improve WSN coverage and minimized the actuation energy consumption during sensors repositioning. The proposed method starts with deploying the sensors randomly in a square area. After the initial deployment the area is divided into small regions and each of the regions is assigned a load value based on the number of sensors in the region. Using the load value the proposed algorithm distributes the sensors evenly among the regions. The algorithm looks for an arrangement that minimize total moving distance and number of movement.

In [23], velocity schedule approach is proposed. The schedule aims to minimize the entire actuation energy consumed by considering all aspects involve in movement such as acceleration, heating, viscous damping, friction and also road condition.

**PSO and WSN Coverage**

Many researches have been conducted in the usage of PSO for WSN coverage improvement. In [24], PSO is used to optimize coverage and communication energy consumption in mobile WSN. The algorithm
proposed is to be executed at the base station and it is divided into two phases where the first phase is for coverage optimization while the second phase is for the minimization of energy consumption. In the first phase of the algorithm, a particle’s position represents a possible sensors arrangement and it is improved based on the coverage calculated in the fitness function. The fitness function calculates the coverage by dividing the ROI into uniform grid, where the coverage is then estimated using the ratio of the total number of grid points covered by at least one sensor to the total number of grid points. The particle with the best coverage value will be selected as the solution for the first phase. Based on this solution, the sensors are moved to their optimal position. In the second phase, the energy consumption in sensor communication is reduced by electing the best set of cluster heads.

Wang et. al. [25] proposed an algorithm based on the virtual force and co-evolutionary PSO. The algorithm is termed as virtual force directed co-evolutionary PSO (VFPCPSO). VFPCPSO is designed to be implemented at a super node. The mobile sensors move to their respective final positions after the position is determined by the super node using VFPCPSO. Based on the concept of virtual force, the sensors are subjected to repulsion and attraction forces. The repulsion force is exerted by the sensors and obstacles on the ROI. The sensors repel each other in order to improve the coverage. The repulsion process needs to take into account that the sensors are still within each other communication range so that their connectivity is maintained. The attraction force comes from preferred areas in the ROI and from sensors that are too far apart (i.e., the sensors are outside of each other’s communication range). In VFPCPSO the virtual force is incorporated to the PSO by introducing a virtual force term to the velocity of PSO. Hence, in addition to the momentum, cognitive and social term, the velocity of VFPCPSO now has an additional term which is to represent the virtual force. The basic PSO search dimension increases exponentially with the increase in the problem dimension. Therefore, in order to overcome this weakness, instead of the basic PSO the authors proposed the use of co-evolutionary PSO (CPSO). In CPSO multiple swarms are co-evolving to optimize the coverage of WSN. The multi dimensional optimization problem is divided into single dimensional problems and each of the swarms is in charge to optimize one of the subdivisions of the problem. CPSO is proven to perform better as compared to basic PSO in terms of coverage and execution time. In [25], the fitness of the solution is evaluated by dividing the ROI into grids similar to [24].

Another work that used PSO for WSN coverage optimization is presented in [26]. In their work, a multiobjective WSN problem is considered. The objectives are to maximize coverage and minimize communication energy consumption. The objectives are tackled using PSO by aggregating the two objectives into one function. The coverage is evaluated using the grid method. The authors suggested combining PSO with simulated annealing (SA) in order to overcome the weakness of PSO in finding local optima. The PSO algorithm is run in a sink node before the SA algorithm is executed. The SA is used to optimize a number of the best solutions found by PSO particles, with the best among the best solutions is optimized by the sink node, while the remaining best solutions are optimized by randomly selected wireless sensor nodes. The proposed algorithm improves the performance of WSN in term of coverage and energy efficiency.

Ngatchou et. al. [27] in their work proposed the usage of sequential PSO (S-PSO) to deploy the wireless sensors. The S-PSO is used to position the sensors sequentially – one at a time – until the coverage requirement is satisfied. The S-PSO reduces the complexity of the problem by tackling a subspace of the problem dimension one at a time.

Individual particle optimization (IPO) is used in [28]. IPO is similar to PSO but without the social term in its velocity, the social term is replaced by a chaotic term. The chaotic term improves exploration ability and makes the algorithm faster. The quality of the solution is measured using the same method as [24].

4. ALGORITHM 1: WSNPSO\textsubscript{VOR}

The problem to be addressed by WSNPSO\textsubscript{VOR} is “How to enhance mobile WSN coverage after random deployment?” There are few assumptions made:

1. The ROI is a two-dimensional square area.
2. The WSN is homogeneous (i.e., all the sensors have similar sensing radius).
3. The sensors know their positions.

The algorithm designed is to be executed at a base station after an initial random deployment. The final optimal position found will be transmitted by the base station to the sensors. Based on this information the sensors will move to their optimal position.

4.1. Particle Encoding

The coding of the particle is straightforward. A particle encodes the positions of the sensors. The position of a sensor $j$ is described by a coordinate $(y_j,z_j)$. Considering $M$ number of sensor nodes, the particle encoding is depicted in Fig. 2. Thus, the dimension of the particle is two times the number of sensors. The final best particle represents the optimum positions of the sensors that lead to the minimum coverage holes.

![Figure 2. Particle encoding](image-url)
4.2. Fitness Function

WSNPSO\textsubscript{vor} utilises Voronoi diagram in the fitness function to measure the coverage. Voronoi diagram is a partition of sites (shown as ◊’s in Fig. 3) in such a way that points inside a polygon are closer to the site of the polygon than any other sites, thus the farthest point of the polygon to its site lies on one of the vertices of the polygon. In other PSO based algorithms grid is used in their fitness function to evaluate the fitness of WSN coverage. Although grid is a good choice for the fitness function but its computational cost is expensive and it relies not only on the number of sensors but also the size of the ROI. The lower bound for the Voronoi diagram computational complexity is \( \Omega \left( N \log N \right) \), where \( N \) is the number of sites – in this case; number of sensors [29].

Voronoi diagram is used to measure the coverage. If all Voronoi polygons vertices are covered, then the ROI is fully covered otherwise coverage holes exist [30]. This can be determined by measuring the Euclidean distance of the vertices to their nearest sensors (\( d_\text{p} \)) [31]. However in WSNPSO\textsubscript{vor} the algorithm aims to minimize the area of the coverage holes (i.e., areas not covered by the sensing field of any sensor) where the Euclidean distance is used as the radius of the holes. This gives a better coverage compared to [31]. The objective of the algorithm is expressed as:

\[
\text{minimize} \quad f_\text{coverage} = \sum_{\text{points of interest}} \text{coverage hole}(\text{point}) \tag{3}
\]

The set of interest points contains:

1. Vertices of the Voronoi polygons:
2. A number of points distributed evenly on the boundary of the polygons.

These additional interest points on the boundary act as pulling forces that prevent the sensors from congregating around a particular point in the ROI. There are two types of boundary points:

a. Corner points: Compulsory boundary interest points, used to provide equal forces from every corner direction thus avoiding congregation.

b. Equally spaced points along the boundary: The points are located in between two corner points. They provide additional pulling force.

The number of points at the boundary has to be carefully chosen because too many points will pull the sensors strongly towards the boundary, wasting and reducing the coverage while too few points are not sufficient to prevent the sensors from congregating.

![Figure 3. Voronoi diagram with 9 sites](image)

![Figure 4. Effect of boundary points](image)

![Figure 5. Hole area estimation](image)
interest_points, the performance improves drastically as shown in Fig. 4(c). When the boundary points are too many and the pulling force is too strong, as in Fig. 4(d), the sensors are pulled too much towards the boundary thus wasting the network’s sensing ability.

The hole area estimation is shown in Fig. 5 and the pseudocode for fitness function computation is shown in Fig. 6. This fitness function is used to determine the quality of solution proposed by PSO. The PSO particles are initialized using the initial sensors position.

\[
f_{\text{coverage}} = 0
\]

Compute Voronoi diagram based on \( X_i \)

\[
\text{interest_points} = \{ \text{Voronoi vertices, boundary points} \}
\]

\[
curr_ip = 0  \quad // \text{current interest point}
\]

\[
\max_ip = \text{maximum number of interest points}
\]

While (curr_ip < max_ip)

Find \( d_p \)

if \( d_p > r, \text{then} \)

\[
\text{coverage}_\text{hole} = \pi \times (d_p - r)^2
\]

// initially hole is computed as a circle

// check which type or interest point is the curr_ip

if curr_ip on ROI’s boundary then

if curr_ip on ROI’s corner then

\[
\text{f}_{\text{coverage}} = \text{f}_{\text{coverage}} + \text{coverage}_\text{hole} / 4
\]

else\[
\text{f}_{\text{coverage}} = \text{f}_{\text{coverage}} + \text{coverage}_\text{hole} / 2
\]

else\[
\text{f}_{\text{coverage}} = \text{f}_{\text{coverage}} + \text{coverage}_\text{hole}
\]

curr_ip = curr_ip + 1;
end while

return \( f_{\text{coverage}} \)

Figure 6. Coverage fitness function

5. ALGORITHM 2: WSNPSOPER

The first algorithm, WSNPSOvor focuses on maximizing the sensor coverage. However, it does not consider the energy consumed in repositioning the sensors. Therefore, here we proposed a second phase algorithm to take the cost of moving the sensors into consideration. The algorithm aims to reduce the distance moved so that the limited energy is saved.

Fig. 7 demonstrates possible initial and the final positions of two sensors (in black). Fig. 7(a) shows the initial positions of these two sensors and their coverage after random deployment. Clearly, this arrangement poorly utilises the sensing capability of the sensors. Fig. 7(b) shows the final positions generated by WSNPSOvor where a better coverage is achieved. WSNPSOvor assigns the sensors to the final positions based on the position sequence in the particle. Based on this procedure, \( s_1 \) needs to move to position 1 (\( y_1, z_1 \)) while \( s_2 \) to position 2 (\( y_2, z_2 \)) (Fig.7(c)). Since the distance of \( s_1 \) to position 1 is further than \( s_2 \) to position 2 the energy consumed by \( s_1 \) during the repositioning is higher than \( s_2 \). This might cause \( s_1 \) to run out of energy before \( s_2 \) which will widens the coverage hole. However, as what can be observed it is better if \( s_1 \) moves to position 2 and \( s_2 \) to position 1, because the maximum distance moved will be shorter and the energy consumed is smaller and fairly distributed (Fig.7(d)).

Therefore a PSO based algorithm; WSNPSOPER is proposed. The purpose of WSNPSOPER is to minimize the maximum distance moved; \( d_{\text{max mov}} \) by any sensor through better assignment of the sensors to their final positions. Hence, WSNPSOPER has the following objective function;

minimize \( f_{\text{energy}} = d_{\text{max mov}} \) \( (4) \)

where \( d_{\text{max mov}} = \max\{d_j\}, j = 1,2,\ldots, M \)

\( d_j \) is the distance moved by sensor \( j \) and \( M \) is the number of sensors in the network.

This is a permutation task, where WSNPSOvor need to achieve its objective without reducing the coverage found by WSNPSOvor.

Table 1. Initial content of WSNPSOPER repository

<table>
<thead>
<tr>
<th>Sensor ID no.</th>
<th>Initial coordinate</th>
<th>Position ID no.</th>
<th>Coordinate</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_{\text{initial}}, z_{\text{initial}} )</td>
<td>( y_{\text{pos}}, z_{\text{pos}} )</td>
<td>( y', z' )</td>
<td>( d_{\text{max mov}} )</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( y_1, z_1 )</td>
<td>( y_1', z_1' )</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( y_8, z_8 )</td>
<td>( y_8', z_8' )</td>
<td>( z_{\text{max}} )</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>( y_{13}, z_{13} )</td>
<td>( y_{13}', z_{13}' )</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( M )</td>
<td>( y_M, z_M )</td>
<td>( y_M', z_M' )</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

The flow of WSNPSOPER is as follows; WSNPSOPER takes as input the following information about the sensors: the sensor IDs, their initial positions and the final positions of the sensors as found by WSNPSOvor. WSNPSOPER stores this information in a repository as depicted in Table 1. WSNPSOPER assigns
the sensors to the final positions such that the maximum distance moved \((d_{\text{max} \text{ mov}})\), is kept at a minimum.

A simple encoding that makes it independent of the problem size (number of sensors) is proposed here. In this implementation a particle encodes a single parameter which is an ID of a sensor.

The global solution \((P_g)\) represents the sensor with the maximum distance to move, which is determined based on the arrangement that gives the best fitness – the smallest \(d_{\text{max} \text{ mov}}\). Particle fitness evaluation involves swapping the final position of the sensor encoded by particle with the sensor encoded by \(P_g\) (Fig. 8). During the swap process the repository is updated if the particle leads to a reduction in \(d_{\text{max} \text{ mov}}\). The fitness of the particle is measured by the value of \(d_{\text{max} \text{ mov}}\). A good particle contributes to smaller \(d_{\text{max} \text{ mov}}\) and in contrary a bad particle increases \(d_{\text{max} \text{ mov}}\).

The swarm update process continues until the maximum number of iterations is reached. The final optimum solution that assigns the sensors to the positions is represented in the final content of the repository. Fig. 9 represents the relation of WSNPSO\text{per} with WSNPSO\text{vor}.

---

**Table:**

<table>
<thead>
<tr>
<th>Sensor ID no.</th>
<th>Initial coordinate</th>
<th>Final position</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(y_{\text{init}})</td>
<td>(z_{\text{init}})</td>
<td>(y_{\text{pos}})</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(y_8)</td>
<td>(z_8)</td>
<td>(P_{\text{pos}_8})</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>(y_f)</td>
<td>(z_f)</td>
<td>(P_{\text{pos}_f})</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 8. Sensors swapping conducted in fitness function*

*Figure 9. Flow diagram of WSNPSO\text{per} and WSNPSO\text{vor}***
6. ALGORITHM 3: WSNPSOCON

WSNPSO_{con} minimizes the actuation energy consumption as small as possible. However, WSNPSO_{con} is not suitable for limited mobility WSN, therefore the third algorithm is aiming for this type of WSN where the maximum distance travelled \((d_{max\text{mov}})\) is limited to a threshold value; \(D_{max}\). Due to this limitation, the objective function is now expressed as a constrained optimization problem;

\[
\text{minimize } f_{\text{coverage}} = \sum_{\text{points interest}} \text{coverage}_h \text{hole}(\text{point})
\]

subject to \(d_{max\text{mov}} \leq D_{max}\)

where \(d_{max\text{mov}}\) is the actual length of the maximum distance moved among the sensors. \(f_{\text{coverage}}\) value is calculated using the same procedure as WSNPSO_{vor}.

Using the penalty method, Eq. 5 is redefined as;

\[
\text{minimize } f_{\text{coverage}} + \gamma P(d_{\text{max\text{mov}}})
\]

where \(\gamma\) is a positive value penalty parameter and \(P(d_{\text{max\text{mov}}})\) is the penalty function. The penalty function penalizes any solution outside the feasible region and it is normally defined as a function of constraints. An absolute value penalty function \(P(d_{\text{max\text{mov}}})\) is used here;

\[
P(d_{\text{max\text{mov}}}) = \max(0, d_{\text{max\text{mov}}} - D_{\text{max}}) \tag{7}
\]

The accuracy of the approximation of the optimal solution found by a penalty method is controlled by \(\gamma\). A large value of \(\gamma\) results in a heavier penalty to any breach of constraint, a very severe penalty makes it hard to find an optimal solution [32]. On the contrary, a small penalty might be too lenient causing an infeasible solution.

In [33], fuzzy penalty approach is proposed to provide a suitable value of penalty to the objective function based on the situation of the solution. The solution is evaluated with respect to its original objective and constraint function before the penalty value is decided.

A similar concept is used here where \(d_{max\text{mov}}\) is associated with fuzzy values such as good and bad, whereas \(\gamma\) is:

\[
\gamma = \exp^\alpha
\]

The value of \(\alpha\) is decided based on the condition of the solution with respect to its fuzzy value associated constraint function, where:

- if \(d_{max\text{mov}} \text{ is good and } d_{max\text{mov}} \text{ is not bad}\) then \(\alpha = \alpha_1\)
- if \(d_{max\text{mov}} \text{ is good and } d_{max\text{mov}} \text{ is bad}\) then \(\alpha = \alpha_2\) /*not good, not bad*/
- if \(d_{max\text{mov}} \text{ is not good and } d_{max\text{mov}} \text{ is bad}\) then \(\alpha = \alpha_3\)

\(\alpha_1, \alpha_2\) and \(\alpha_3\) are as follows:

\[
\alpha_1 = 0
\]

\[
\alpha_2 = \frac{d_{max\text{mov}} - D_{\text{max}}}{\Delta}
\]

\[
\alpha_3 = 2\left(\frac{d_{max\text{mov}} - (D_{\text{max}} + \Delta)}{D_{\text{max}} - (D_{\text{max}} + \Delta)}\right) + 1
\]

Fig. 10 represents the Fuzzy set used to determine the penalty parameter; \(\gamma\) and \(\alpha\) based on the value of \(d_{max\text{mov}}\).

\[\text{Figure 10. Fuzzy set for } \gamma\]

\[\text{Figure 11. Change of } \gamma \text{ with respect to } d_{max\text{mov}}\]

The operation flow diagram of WSNPSO_{con} is shown in Fig. 12. WSNPSO_{con} uses the same particle encoding as WSNPSO_{vor}. In every iterations, the maximum distance moved by any particle \(d_{max\text{mov}}\) is passed to the fuzzy system to compute a new value of the penalty parameter \(\gamma\). This value is passed to PSO to be used in the fitness function for the particles fitness evaluation. The stopping criteria are either; the
maximum number of iterations is reached or 100% coverage with \( d_{\text{max}} \leq D_{\text{max}} \) – is met.

![Image](image.png)

**Figure 12. WSNPSO\text{con}**

### 6. RESULTS & DISCUSSION

The performances of the algorithms discussed in this paper are studied and compared with PSOGrid algorithm [24]. Two size of grid are used for the PSOGrid algorithm: 1×1 and 5×5. The parameters of the tests conducted are shown in Table II. Each of the tests is run 20 times. For WSNPSO\text{con} the maximum distance moved; \( D_{\text{max}} \), is set to 20. The algorithms are implemented using MATLAB.

<table>
<thead>
<tr>
<th>Size of ROI</th>
<th>Sensing Radius</th>
<th>No. of Sensors</th>
<th>Coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test I 50x50</td>
<td>5</td>
<td>20</td>
<td>( \approx 62 ) 43.21</td>
</tr>
<tr>
<td>Test II 50x50</td>
<td>5</td>
<td>40</td>
<td>100 67.84</td>
</tr>
<tr>
<td>Test III 50x50</td>
<td>5</td>
<td>60</td>
<td>100 79.73</td>
</tr>
<tr>
<td>Test IV 100x100</td>
<td>7</td>
<td>100</td>
<td>100 75.23</td>
</tr>
</tbody>
</table>

The results of the tests show that all algorithms tested successfully improve the coverage (Fig. 13). PSOGrid with grid size 1×1 gives a better coverage compared to other algorithms however due to the smaller grid size the cost of the fitness function is increased, Fig. 14. Increasing the grid size to 5×5 reduces execution time but compromised the quality of coverage. WSNPSO\text{vor}, WSNPSO\text{per} and WSNPSO\text{con} provide coverage quality close to PSOGrid 1×1 but with a better time efficiency especially with larger network (test IV, Fig. 14).

The contribution of WSNPSO\text{per} and WSNPSO\text{con} in reducing the actuation energy usage through minimization of the maximum distance moved is presented in Fig. 15. In comparison to WSNPSO\text{vor} it could be seen that WSNPSO\text{con} is able to minimize the energy usage, while WSNPSO\text{per} maximizes the coverage without violating the constraint (i.e., the maximum distance moved is always lower than 20 in all the tests. In test I-III, the contribution of WSNPSO\text{con} might seem irrelevant compared to WSNPSO\text{per}, as the maximum distance moved by sensors of WSNPSO\text{per} is smaller than WSNPSO\text{con}. However, as the area of ROI grows (test IV) the contribution is clearer, where in a larger ROI, WSNPSO\text{per} is not able to minimize the maximum distance moved to be less than 20 but WSNPSO\text{con} manages to ensure the maximum distance moved is below 20. Hence, when a sensor is given limited mobility, WSNPSO\text{con} is a good choice as it ensures the constraint is obeyed regardless of the size of the ROI and density of the network.

### 7. CONCLUSION

Coverage is an important issue in WSN where it is used to measure the quality of service of the network. Mobile WSN can improve its coverage by moving the sensors, however movement consumes significant amount of energy. In this paper three PSO based algorithms are presented WSNPSO\text{vor}, WSNPSO\text{per} and WSNPSO\text{con}. The algorithms aim to maximize coverage (WSNPSO\text{vor}), maximize coverage while minimizing energy usage (WSNPSO\text{per}) and maximize the coverage of limited mobility WSN (WSNPSO\text{con}). Tests conducted show that the algorithms are able to achieve their objective.

![Graph](graph.png)

**Figure 13. Coverage percentage of the algorithms tested**
Figure 14. Execution time (s) of the algorithms tested

Figure 15. Sensors’ maximum distance moved of the algorithms proposed

ACKNOWLEDGEMENT
I wish to acknowledge the anonymous reviewers for their valuable comments and insights. I also would like to thank Dr Ammar W. Mohamed, Assoc. Prof. Dr Kamarulzaman Ab. Aziz, Assoc. Prof. Dr Mohamad Yusoff Alias, Prof. Mengjie Zhang, Prof. B.S Daya Sagar, and Ms. Syabeela Syahali for their assistance throughout this research.

REFERENCES


Handbook of Computational Geometry, Elsevier Science Publishing


