Relay Placement Based on Divide-and-Conquer

Ravanbakhsh Akhlaghinia, Azadeh Kavianfar, and Mohamad Javad Rostami, *Member, IACSIT*

***Abstract*—In this paper, we define a relay placement problem to cover a large number of sensors according to multiple purposes using a minimal number of relays. Finding the best solution requires exponential run time that takes years in large networks. Therefore, we divide the main problem into sub-problems and design a polynomial-time algorithm for finding an approximate solution. We developed a software tool for running the algorithm and graphical representation of placement. Using this tool, our evaluation experiments show the performance of the polynomial-time algorithm in comparison with the best solution.**

***Index Terms*—Wireless sensor network, placement, coverage, clusterhead, relay.**

I. Introduction

Wireless sensor networks are suitable for many applications, including national security, military operations, and environment monitoring. Node placement is an important area in these networks that studies where to place sensor or relay nodes to improve network performance and reduce energy consumption.

Node placement may be either random or deterministic [1]. In this paper, we consider deterministic placement in which nodes are placed at exact desired locations. We consider the network model (Fig. 1) in which sensor monitors the environment and sends the sensed information to the base station through relay nodes.

In a wireless sensor network (WSN), if a small set of special nodes, whose main function is packet forwarding, are deployed, the management and network operations in the network is simplified. These nodes are called relay nodes or clusterheads [2]. As shown in Fig. 1, each clusterhead covers a number of sensors determined by a virtual circle around it. A fundamental problem [1] which arises when establishing such a network is where to deploy those relay nodes to achieve the required grade of service, while meeting the system constraints.

Fig. 2 demonstrates an example of clustering affect on communication energy consumption. In this figure, two sample clustering options, namely A-B and C-D have been shown. The total square of distances between clusterheads and sensors in clustering A-B is 65 units, while it is 117 units for clustering C-D. Considering the relation of energy dissipation and this simplified metric (total square of distances), Fig. 2 highlights the effect of clustering quality on network's total energy dissipation.

The placement algorithm computes number of required relays considering the fact that a relay may simultaneously cover two or more neighbor sensors and less number of relays is used in this way. Optimal placement is achieved when all the sensors are covered with the minimum number of relays.

The Divide-and-Conquer approach is a practical way to divide the problem space and search in it for a good fast solution. Based on this approach, we design an algorithm that divides the physical area into small sub areas, places clusterheads in each sub area, and then combines the solutions of the sub areas. The algorithm tries to use the lowest number of clusterheads in each sub area. When two sub areas are combined, the result should also contain the minimum number of clusterheads.

The rest of this paper is organized as follows. Section II reviews the existing work on clusterhead placement. In Section III, we define a novel placement problem. In Section IV, we design an approximate algorithm to solve the problem by breaking the problem into sub-problems. Section V introduces an algorithm to solve a sub-problem. Section VI contains the numerical experiments and results. Section VII finally concludes the paper.

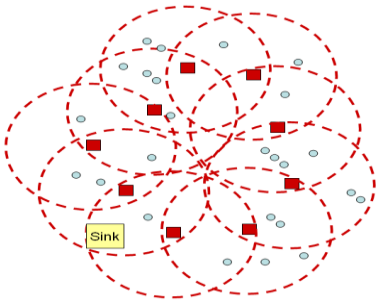


Fig. 1. A clustered wireless sensor network

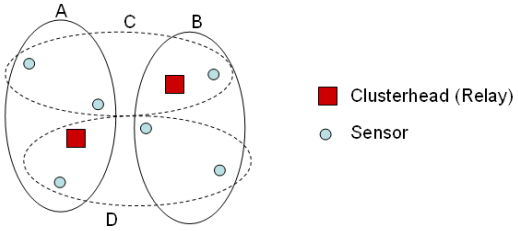


Fig. 2. Two examples of clustering

II. Related Work

We review the prior work related to relay placement. The transmission ranges for relays and ordinary sensors are denoted *R* and *r*, respectively.

Manuscript received October 1, 2011; revised October 16, 2011.

Ravanbakhsh Akhlaghinia is with the Department of Engineering, Azad University of Gachsaran, Gachsaran, Iran (e-mail: akhlaghinia.r@gmail.com).

Azadeh Kavianfar is with Guilan University, Rasht, Iran (e-mail: azadeh_kaviyanfar@yahoo.com).

Mohamad Javad Rostami is with the Department of Computer Engineering, Bahonar University, Kerman, Iran (e-mail: mjrostamy@yahoo.com).

The setting that both relay nodes and sensors can perform the packet forwarding is known as the single-tiered infrastructure. Cheng et al. [3] developed algorithms to place the minimum number of relay nodes and maintain the connectivity of a single-tiered WSN, under the assumption that $R = r$. The problem was modeled by the Steiner Minimum Tree with Minimum Number of Steiner Points and Bounded Edge Length (SMT-MSP) problem, which arose in the study of amplifier deployment in optical networks, and was proved to be NP-hard [4]. Based on a minimum spanning tree, Lin and Xue [4] developed an algorithm to solve the SMT-MST problem. They proved it to have an approximation ratio of 5, which Chen et al. [5] tightened to 4. In the same paper, a 3-approximation algorithm was also proposed. Based on Lin and Xue's algorithm, Cheng et al. [3] proposed a different 3-approximation algorithm and a randomized 2.5-approximation algorithm.

In order to provide fault-tolerance, Kashyap et al. [6] studied how to place minimum number of relays such that the resulted WSN is 2-connected, when $R = r$. Zhang et al. [7] improved the results of Kashyap et al. by developing algorithms to compute the optimal node placement for networks to achieve 2-connectivity, under the more general condition that $R \geq r$. These algorithms aimed to minimize the number of relay nodes while providing fault-tolerance.

The setting that only relay nodes can perform the packet forwarding is known as the two-tiered infrastructure. Pan et al. [8] first investigated the two-tiered infrastructure for optimal node placement. Further studies considering an i.i.d. uniformly distributed sensor location with $R \geq 4r$ were given in [9] and [10]. Lloyd and Xue [11] developed algorithms to find optimal placement of relay nodes for the more general relationship $R \geq r$, under single-tiered and two-tiered infrastructures.

III. The MPCHP Placement Problem

First, we define an optimization problem called MinCHP (Minimum ClusterHead Placement) that determines the minimum number of clusterheads which are required to satisfy a number of purposes. Then, we define a relay placement problem called MPCHP (Multi-Purpose ClusterHead Placement) in wireless sensor networks that places the minimum number of clusterheads determined by MinCHP in the network while considering more purposes. We design a polynomial-time algorithm called solveMPCHP to find an approximate solution to MPCHP. solveMPCHP is flexible in a way that it can consider more purposes.

A. Problem Definition

In this section, we define and formulate the placement problem. In the next sections, we refer to the conditions defined in this problem.

Problem MinCHP: Given a number of sensors randomly located in two dimensional area *A*. Find the minimum number $n_{\min}$ of cluster heads which are required to be placed in *A* such that the following conditions are satisfied:

Condition 1: For each sensor *s*, there is at least one clusterhead *c* such that $dis\tan ce(s,c) \leq r$.

Condition 2: No convex sub-area A' exists in *A* such that there are more than *M* sensors per clusterhead in A'.

Condition 3: Each clusterhead *c* has two disjoint paths to the base.

Two cluster heads are able to directly communicate if their distance is not more than R. In other words, two clusterheads are able to directly communicate if they are located on the border or inside of the communication circle of each other. A clusterhead and a sensor are able to directly communicate if their distance is not more than r.

Clusterhead c has two disjoint paths to the base, if at least two clusterheads are placed on the border or inside of the communication circle of c. If the number of sensors around c is high, then more than two clusterheads may be required within the communication range of c to cover the sensors and c may get more than two disjoint paths to the base.

Now, we want to extend MinCHP to a placement problem that also considers the following two purposes.

- Purpose 1: Clusterheads are placed as close as possible to sensors.
- Purpose 2: There exist similar numbers of sensors around clusterheads.

We define two parameters called *t* and *u* for a deployed wireless sensor network containing *n* clusterheads $c_1, c_2, \ldots, c_n$. Let us assume that the network is clustered in a way that n_i sensors $s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}$ are members of clusterhead c_i, for $i = 1,2,\ldots,n$. d_i equals the summation of distances between c_i and its sensors for $i = 1,2,\ldots,n$. We define

$$e = \frac{1}{n}\sum_{i=1}^{n} n_i \qquad (1)$$

Then, we define

$$u = \frac{1}{n}\sum_{i=1}^{n} d_i \quad \text{and} \quad t = \frac{1}{n}\sum_{i=1}^{n} (n_i - e)^2 \qquad (2)$$

Parameter *u* quantifies Purpose 1 and parameter *t* quantifies Purpose 2 in a case of node placement. Now, we are able to define the MPCHP problem.

Problem MPCHP: Given a number of sensors randomly located in two dimensional area *A*. Place the minimum number $n_{\min}$ of clusterheads determined by MinCHP in *A* in a way to maximize

$$\frac{\min(t)}{t} + \frac{\min(u)}{u} \qquad (3)$$

such that the following conditions are satisfied:

Condition 1: For each sensor *s*, there is at least one clusterhead *c* such that $dis\tan ce(s,c) \leq r$.

Condition 2: No convex sub-area A' exists in *A* such that there are more than *M* sensors per clusterhead in A'.

Condition 3: Each clusterhead *c* has two disjoint paths to the base.

where *t* and *u* are the parameters of the placement defined in (2), and min(*t*) and min(*u*) are the minimum values for *t* and *u* among possible cases of placement.

IV. An Approximated Solution to MPCHP

In this section, we break MPCHP into smaller

sub-problems and design an algorithm to find an approximated solution to it.

A. Dividing the Problem

Finding the best solution requires exponential execution time and is impossible if number of sensors is large and the network area is huge. To solve the problem, we use a divide-and-conquer approach to design an algorithm with polynomial time complexity. To do this, the main problem is divided into smaller sub-problems and each sub-problem is solved separately. The combination of the solutions of the sub-problems is an approximate solution to the main problem.

We define the sub-problem of MPCHP called SubMPCHP to be something like MPCHP but try to optimize sensor coverage in the area limited by the communication circle of a single clusterhead. Then, SubMPCHP will be much simpler than MPCHP to solve.

First we define a problem called SubMinCHP to determine the minimum number of clusterheads in a sub-problem as follows.

Problem SubMinCHP: Given a number of sensors randomly located in two dimensional area *A*. A number of clusterheads are placed in *A*. Consider the placed clusterhead C. Find the minimum number $n_{sub\min}$ of clusterheads which are required to be placed on the border or inside the communication circle of C such that the following conditions are satisfied:

Condition 4: For each sensor *s* located on the border or inside the communication circle of C, there is at least one clusterhead *c* such that $dis\tan ce(s,c) \le r$.

Condition 5: No convex sub-area *A*' exists in the communication circle of C such that there are more than *M* sensors per clusterhead in *A*'.

Condition 6: Clusterhead *C* finds two disjoint paths to the base.

Let us consider the sub-problem of clusterhead C in the sensor network depicted in Fig. 3. We can achieve the followings by placing two clusterheads at the two locations shown in Fig. 3.

- C finds two paths.
- All the sensors on the border or inside the communication circle of C are covered.
- Some neighbor sensors outside the communication circle of C are covered.

Problem SubMPCHP: Given a number of sensors randomly located in two dimensional area *A*. A number of clusterheads are placed in *A*. Consider the placed clusterhead C. Place the minimum number $n_{sub\min}$ of clusterheads determined by SubMinCHP on the border or inside the communication circle of C in a way to maximize

$$\frac{\min(t)}{t}+\frac{\min(u)}{u} \qquad (4)$$

Such that the following conditions are satisfied:

Condition 4: For each sensor *s* located on the border or inside the communication circle of C, there is at least one clusterhead *c* such that $dis\tan ce(s,c) \le r$.

Condition 5: No convex sub-area *A*' exists in the communication circle of C such that there are more than *M* sensors per clusterhead in *A*'.

Condition 6: Clusterhead *C* finds two disjoint paths to the base.

where *t* and *u* are the parameters of the placement defined in (2), and min(*t*) and min(*u*) are the minimum values for *t* and *u* among possible cases of placement.

B. Solution

Based on the definition of SubMPCHP, we propose algorithm *solveMPCHP* to solve the MPCHP problem.

Algorithm. solveMPCHP

While there are still uncovered sensors in the network do

1) Place a clusterhead at a point in the network where it covers the highest number of uncovered sensors.
2) While there is unvisited clusterhead C in the network do
   - Make a SubMPCHP problem for C.
   - Solve the SubMPCHP problem.
   - Mark C as visited.

End.

V. Solving The SubMPCHP Problem

In this section, we propose an algorithm to solve the SubMPCHP problem.

According to the solveMPCHP algorithm, we place clusterheads one by one to reduce algorithmic complexity. Until all the clusterheads are not placed, we can not guarantee that every clusterhead finds two disjoint paths to the base. We should place the minimum number of clusterheads to achieve this property.

Condition 7: Each clusterhead contains at least two clusterheads within its communication range which are not within the communication ranges of each other.

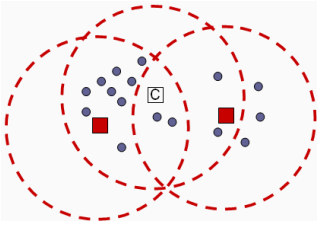
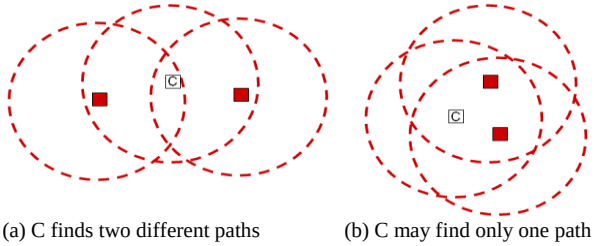
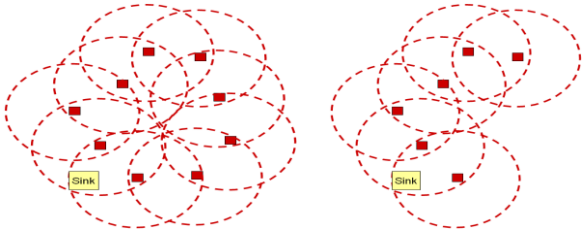


Fig. 3. Placing two clusterheads inside the communication circle of C



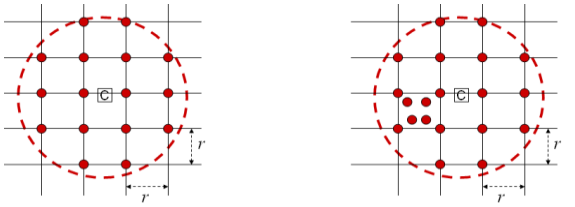
(a) C finds two different paths (b) C may find only one path

Fig. 4. Making two paths for clusterhead C



(a) A ring of clusterheads (b) A number of clusterheads without ring

Fig. 5. A number of placed clusterheads



(a) The virtual grid (b) Four clusterheads are placed in a sub-area

Fig. 6. The virtual grid on the placement area of a sub-problem

If we follow condition 7 while solving SubMPCHP, then it guarantees that C finds two different paths (Fig. 4(a)). Otherwise, the case of Fig. 4(b) may happen. In this case, C has two neighbor clusterheads, but they both lead to one single clusterhead at the right. In other words, C finds only one path through two different neighbors.

If we follow condition 7 while solving the sub-problems, then the resulted network contains rings of clusterheads (such as the one depicted in Fig. 5(a)). Every clusterhead will be part of a ring such that we can not have a broken line of clusterheads (such as the one depicted in Fig. 5(b)).

It is not efficient to check all possible points within the communication range of C for clusterhead placement. To reduce number of candidate points, we consider a virtual grid (Fig. 6(a)) on the communication circle of C. Each grid point is a candidate for clusterhead placement. The grid makes it possible to distribute the clusterheads as uniform as possible. We set the distance between grid points at minimum to keep number of points small. Considering too few grid points leads to missing efficient candidate locations. Since the distance between a clusterhead and a sensor must be no more than r to communicate, we set the distance on the grid to r. We may sometimes need to place more clusterheads in a sub-area than what the grid offers. This case happens only because of high density of sensors to satisfy Condition 5. In this case, we place additional clusterheads in sub-areas of the grid (such as Fig. 6(b)).

Algorithm solveSubMPCHP()

Place a virtual grid GR on the network with a size and location that satisfies the following two conditions.

1) Grid points are only located within the distance of R from clusterhead C.
2) The distance between every two neighbor points of GR is r.

For every rxr square sub-area SA on GR, do

1) While there are more than M sensors per clusterhead in SA, do
   - Place a new clusterhead at a random location in SA.
   - Place new clusterheads at all the points of GR.
   - Mark all the points of GR as non-visited.
   - While there is non-visited points on GR do
2) Select non-visited point gp which its clusterhead covers the least number of sensors. In the case of a tie, select the point which is closer to C.
3) Remove the clusterhead at gp if both Condition 4 and Condition 7 remain satisfied after removing this clusterhead.
4) Mark gp as visited.
   - Set n_c equal to the number of clusterheads on GR.
   - Remove all the clusterheads on GR.
   - Among all the cases of placing n_c clusterheads on GR, select the case that
5) satisfies both Condition 4 and Condition 7
6) and maximizes (4).
   - Return the locations of the currently-placed clusterheads in the network as the solution.

VI. Numerical Evaluation

In this section, we evaluate the solveMPCHP algorithm comparing with two other relay placement algorithm and we review the results. We define the scenario defined in Table I. In this scenario, number of sensors is changed while network dimension is fixed.

We evaluate the following three relay placement algorithms in our experiments.

1) solveMPCHP: presented in this paper
2) findBestMPCHP: the best solution of MPCHP found by testing all the cases of placement leading to exponential run time
3) Algorithm2.2: proposed in [10]

TABLE I: Evaluation Parameters

Parameter	Value
Network Dimension	500m x 500m
Clusterhead's Communication Range	50 m
Sensor's Communication Range	30 m
Number of Sensors	variable from 100 to 800
Sensor Distribution in Network	Non-uniform

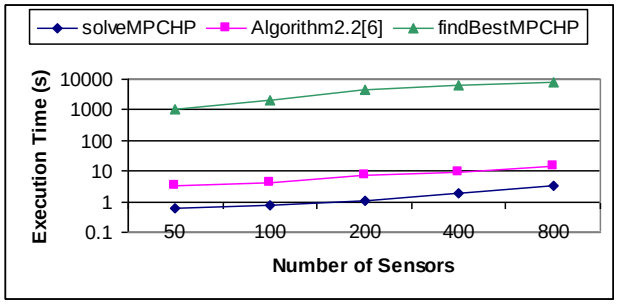


Fig. 7. Execution time versus number of sensors

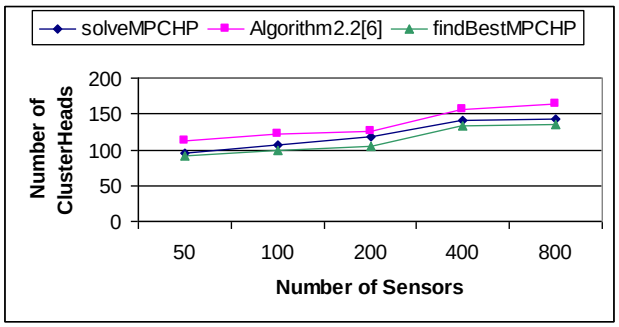


Fig. 8. Number of clusterheads versus number of sensors

A. Numerical Results

Execution of solveMPCHP takes a number of seconds whereas evolutionary placement algorithms such as [12] typically need tens of minutes to complete. Fig. 7 compares solveMPCHP in execution time to findBestMPCHP and Algorithm2.2. According to this chart, solveMPCHP is averagely 5 times faster than Algorithm2.2 and 3000 times

faster than findBestMPCHP.

Fig. 8 shows number of required clusterheads to solve MPCHP versus number of sensors. We expect that by increasing number of sensors, number of clusterheads goes up, because an additional sensor may require a new clusterhead for coverage. But not every sensor requires a dedicated clusterhead since a clusterhead can simultaneously cover multiple sensors. So, doubling number in the horizontal axis, number in the vertical axis is averagely magnified by 1.1 in the chart. That is, number of sensors has a slight effect on number of required clusterheads. The solution of solveMPCHP is close to the best solution. The performance ratio of solveMPCHP to findBestMPCHP is averagely 1.06.

B. *Implemetation*

We developed a graphical environment written in Java to run and display clusterhead placement and implemented the proposed algorithm in it. Fig. 9 shows a view of the software. In the network area, sensors are displayed as blue points. Clusterhead is displayed as a red point with the communication circle around it. The software is able to initially generate sensors at either deterministic or random locations in the network. After generating sensors, the placement algorithm is executed.

Part of the graphical environment of the software is developed in [12]. Now, we look at the internal structure of the software which is developed as a Java project in Eclipse [13]. Fig. 10 shows the components of the software and their relation. Each component is defined as a Java class. The software is composed of the following classes:

1) Parameter: This class keeps the global parameters of the evaluation which are configured at the beginning.
2) Sensor: This class defines a sensor node and stores its location.
3) ClusterHead: This class defines a clusterhead and stores its location.
4) NetInfo: This class stores network information including sensors and clusterheads.
5) AreaPanel: This is a class that is displayed as a white graphical area within the main frame in which sensors and clusterheads are drawn. Size of AreaPanel is configured at the beginning as a parameter.
6) ParameterPanel: This is a class that is displayed as a frame in the left corner of the main frame in which the Run button and a text box are placed.
7) Placement: The placement algorithm is implemented in this class that reads network information from the NetInfo class. It determines locations of clusterheads and then uses the DrawUtility class for drawing them.
8) DrawUtility: This class contains routines for drawing sensors and clusterheads on AreaPanel.
9) GenarateSensors: A routine in this class is called at the beginning of the evaluation to generate a number of sensors in random/deterministic locations.
10) CoverageFrame: This class is the main frame that contains an object of the AreaPanel class and an object of the ParameterPanel class. This class calls routine runPlacement() from the Placement class to run the placement algorithm.

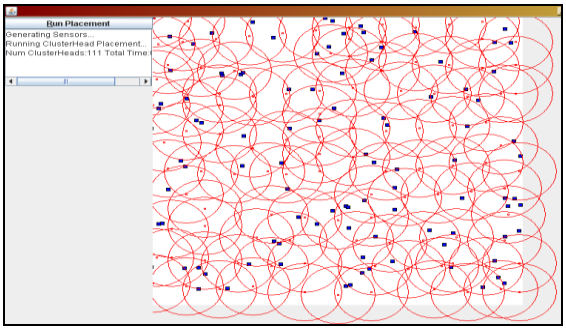


Fig. 9. A view of the clusterhead placement software

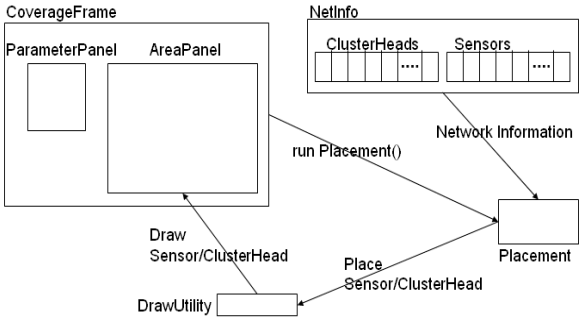


Fig. 10. Components of the clusterhead placement software

VII. Conclusion

In this paper, we define a clusterhead placement problem called MPCHP to cover a large number of sensors according to multiple purposes using a minimal number of clusterheads.

Since finding the best solution requires exponential run time, we divide MPCHP into sub-problems and design a polynomial-time algorithm called solveMPCHP for finding an approximate solution to MPCHP. We developed a software tool for running the algorithm and graphical representation of placement. Our evaluation experiments show the performance of the solveMPCHP algorithm in comparison with the best solution.

References

[1] J. Li, L. L. H. Andrew, C. H. Foh, M. Zukerman, and H.-H. Chen, "Connectivity, Coverage and Placement in Wireless Sensor Networks", *Sensors*, vol. 9, no. 10, pp. 7664-7693, 2009.

[2] H. Karl, and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, Chichester, U.K: Wiley, 2005, ch. 10, pp. 274-285.

[3] X. Cheng, D. Du, L. Wang, and B. Xu, "Relay Sensor Placement in Wireless Sensor Networks", *Wirel. Netw.*, vol. 14, pp. 347–355, 2008.

[4] G. Lin, and G. Xue, "Steiner Tree Problem with Minimum Number of Steiner Points and Bounded Edge-Length", *Inf. Proc. Lett.*, vol. 69, pp. 53–57, 1999.

[5] D. Chen, D. Du, X. Hu, G. Lin, L. Wang, and G. Xue, "Approximations for Steiner Trees with Minimum Number of Steiner Points", *J. Glob. Opt.*, vol.18, pp.17–33, 2000.

[6] A. Kashyap, S. Khuller, and M. Shayman, "Relay Placement for Higher Order Connectivity in Wireless Sensor Networks", in *Proc. 25th IEEE International Conference on Computer Communications*, Barcelona, 2006, pp. 1-12.

[7] W. Zhang, G. Xue, and S. Misra, "Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Problems and Algorithms", in *Proc. 26th IEEE International Conference on Computer Communications*, Anchorage, AK, USA, 2007, pp. 1649–1657.

[8] J. Pan, Y. Hou, L. Cai, Y. Shi, and S. Shen, "Topology control for Wireless Sensor Networks", in *Proc. ACM Mobicom'03*, San Diego, CA, USA, 2003, pp. 286–299.

[9] B. Hao, J. Tang, and G. Xue, "Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Formulation and Approximation", in *Proc.*

IEEE Workshop High performance switching and routing, U.S.A., 2004, pp. 246–250.
[10] J. Tang, B. Hao, A. Sen, "Relay Node Placement in Large Scale Wireless Sensor Networks", *Comput. Comm.s*, vol. 29, pp. 490–501, 2006.
[11] E. Lloyd, and G. Xue, "Relay Node Placement in Wireless Sensor Networks", *IEEE Trans. Comoput.*, vol. 56, pp. 134–138, 2007.
[12] K. Yıldırım, T. Kalaycı, A. Uğur, "Optimizing Coverage in a K-Covered and Connected Sensor Network Using Genetic Algorithms", in *Proc. 9th WSEAS International Conference on Evolutionary Computing*, Sofia, Bulgaria, 2008.
[13] Eclipse Java Software, Available: http://www.eclipse.org.

Ravanbakhsh Akhlaghinia was born in 1978 in Iran. Mr. Akhlaghinia received his B.Engr. degree from Isfahan University Of Technology (IUT) in 2000, and his M.S. degree from Azad University, Dezfol branch (Dezfol , Iran) in computer engineering in 2010.
From 2004 to 2011, he is working as a network professional at National Iranian South Oil Company (NISOC). He is interested in the area of wireless ad hoc networks.

Azadeh Kavianfar was born in 1981 in Iran. Ms. Kavianfar received her B.Engr. degree from Azad University, Lahijan branch (Lahijan, Iran) in 2004 and her M.S. degree from Sharif University, Kish branch (Kish, Iran) in IT engineering in 2009 Since 2006, she is working as a lecturer at Azad University, and as an administrator of educational system at Guilan University.She is interested in the area of wireless networks and network engineering.

Mohammad Javad Rostami was born in 1978 in Iran. He received his B.Sc in computer engineering from Bahonar University (Kerman, Iran) in 2001 and M.Sc in computer engineering from Amirkabir University of Technology (Tehran, Iran) in 2005. He has been a faculty member of Bahonar University since 2006. His main research interests include diverse routing and QoS routing algorithms, wireless sensor networks, and heuristic network algorithms Mr. Rostami is a member of International Association of Computer Science and Information Technology.