Localized Movement-Assisted Sensor Deployment in Wireless Sensor Networks

Shuhui Yang and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Fei Dai
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND 58105

***Abstract*— In a wireless sensor network (WSN), the sensor distribution is vital to the quality of service (QoS) of the network, because the effectiveness of the network depends on the coverage of the monitoring area. In WSNs where sensors have locomotion facilities, after the initial deployment, the sensors can move around and self-redeploy to ensure global coverage and load balance. The movement-assisted sensor deployment aims at moving sensors to meet coverage and load balance requirements. In this paper, we focus on developing a distributed and localized solution, approximating the global optimal solution [1]. The proposed local solution has similar performance to the Hungarian method, without centralized control. Extensive theoretical analysis together with simulations has been done to verify the effectiveness of the proposed distributed solutions.**

***Index Terms*— Hungarian method, load balance, local solution, sensor deployment, wireless sensor networks.**

I. Introduction

A wireless sensor network (WSN) [2] is a distributed system for information collection combining sensing, processing, and communications. The effectiveness of a WSN depends on the coverage of the monitoring area by the deployed sensors. Generally, a sufficient number of sensors are used to ensure the coverage and even a certain degree of redundancy so that sensors can rotate between active and sleep modes. However, a good sensor distribution is still needed for coverage and to balance the workload of sensors. By load balance, we mean each unit of monitoring area is covered by the same number of sensors. Recently, equipped with locomotion facilities, sensors can move around after initial deployment. Thus, WSNs are now capable of self-deploying to further improve the coverage and load balance.

In existing works, two methods are used to enhance the sensor coverage: *incremental sensor deployment* and *movement-assisted sensor deployment*. Incremental self-deployment [3] incrementally deploys sensors, with each one using information gathered from previously deployed nodes to determine its optimal location. This method is developed for robot applications, and a centralized control is necessary. Movement-assisted sensor deployment [4] uses a potential-field-based approach to move existing sensors by treating sensors as virtual particles, subject to virtual forces. Note that here load balance implies coverage and hence it is a stronger requirement. To achieve coverage/load balance, various optimization problems can be defined to minimize different parameters, including total moving distance, total number of moves, communication/computation cost, and convergence rate.

In SMART [5], Wu and Yang related the sensor deployment in a flat 2-D grid-based mesh to the classic load balance problem in parallel processing. They proposed a *scan-based solution* that does not resort to global (load) information. One unique issue in WSNs called the communication hole problem was identified and addressed. In a recent paper [1], they further proposed an optimal solution for the sensor deployment issue in 2-D meshes. This solution is based on the classic Hungarian method, which requires global information, and achieves optimal total moving distance and number of moves.

In this paper, we focus on the distributed and localized load balance solutions in WSNs that minimize the total moving distance of sensors and the number of moves. The basic monitoring area is still a 2-D grid-based mesh (2-D mesh). We provide a local solution in 2-D meshes which has approximate performance compared with the optimal global solution. This local solution can be extended to solve load balance in any network topology. Specifically, in this paper, we: (1) propose a local load balance solution which has approximate performance compared with the Hungarian method based optimal solution, (2) conduct theoretical analysis to verify the effectiveness and limitations of the proposed solution, and (3) perform extensive simulations to verify its performance.

II. Preliminaries and Related Work

A. Movement-assisted sensor deployment overview

The sensor placement issue has been widely studied recently [6], [7]. In incremental sensor deployment [3], nodes are deployed one by one, using the location information of

This work was supported in part by NSF grants CCR 0329741, CNS 0422762, CNS 0434533, ANI 0073736, EIA 0130806, and CNS 0531410.
Emails: syang1@fau.edu, jie@cse.fau.edu, fdai@ece.ndsu.edu.

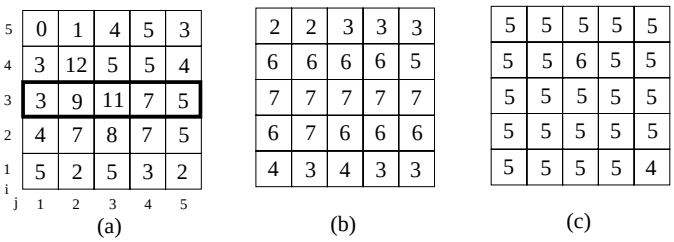


Fig. 1. An ideal case for SMART: (a) initial deployment, (b) after row scan, and (c) after column scan.

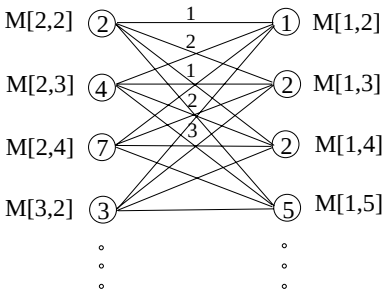


Fig. 2. The node and edge weighted bipartite graph of Figure 1 with "give" grids at the left-hand side and "take" grids at the right-hand side.

previously deployed nodes to deploy the current one. This centralized algorithm is not scalable and is computationally expensive. Most existing movement-assisted sensor deployment protocols rely on the notion of virtual force to move existing sensors from an initial unbalanced state to a balanced state.

In [8], Zou and Chakrabarty proposed a centralized virtual force based mobile sensor deployment algorithm (VFA), which combines the idea of potential field and disk packing [9].

In [10], Wang, Cao, and La Porta used Voronoi diagrams [11] to find coverage holes in the sensor network, and proposed algorithms to guide sensor movement toward the coverage hole. The termination condition of their algorithms is coverage instead of load balance. In a recent work [12], they proposed a grid-quorum solution to quickly locate the closest redundant sensors to the target area, where a sensor failure occurs.

B. SMART: a scan-based approach

The scan-based SMART [5] approach is a hybrid of local and global. In the $n \times n$ 2-D mesh of grids, two scans are used in sequence: one for all rows, followed by another for all columns. Within each array, the scan operation, where the prefix sum of the loads in all grids is passed on from one end of the array to the other, is used to calculate the average load and then to determine the amount of overload or underload in grids. Load is shifted from overloaded grids to underloaded grids in an optimal way to achieve a balanced state.

The 2-D scan process involves a row scan followed by a column scan as shown in Figures 1 (b) and 1 (c), respectively. The result of the 2-D scan process usually does not generate an ideal global balanced state (as in Figure 1, still one grid is underload and one is overload). However, the maximum load difference between any two grids is bounded by 2.

C. The Hungarian method based optimal solution

The Hungarian method is proposed to solve the edge weighted matching problem in a complete bipartite graph $K_{m,m}$ with number associated edges called weights. The objective is to find a perfect matching (of m pairs), such that the sum of the weights of edges in the matching is maximum (or minimum).

To use the Hungarian method for load balance in WSNs, we need to first convert the 2-D mesh to a complete bipartite graph. After the global average is achieved, the "give" and "take" grids appear at the left and right hand sides of the bipartite graph, respectively. The node weight corresponds to the amount of overload and underload, and the edge weight represents the distance between the "give" and "take" grids in a matching pair as shown in Figure 2 (based on the example in Figure 1). The edge weight is the distance between two end nodes $M[i,j]$ and $M[i^{'},j^{'}]$. Then an edge weighted perfect bipartite graph is derived by expanding each node with weight k to k "clone" nodes. The edge weight of clone nodes will be inherited from the original nodes. The Hungarian method is then applied to this graph and the optimal result is to find m lines from the m^2 dotted lines.

The cost of the Hungarian method for load balance in WSNs is $O(m^3)$ [13], where m is the amount of overloads (underloads) which is bounded by the number of sensors. Usually, the number of sensors is one or two magnitudes higher than the number of grids (n). The solution based on the Hungarian method is centralized.

III. A Local Solution for the Hungarian Algorithm

In this section, we propose a local solution for the Hungarian method. After the states of the grids are decided, the grids communicate among themselves to determine the load transferring without the centralized control.

A. Assumptions

The following assumptions are used: (1) The monitoring and deployment area is an $n \times n$ grid, with each grid of size $r \times r$. In the grid-based 2-D mesh, each grid point at position (i,j) has four neighbors at positions: $(i-1,j)$, $(i,j-1)$, $(i,j+1)$, and $(i+1,j)$. Among existing approaches, TTDD [14] and GAF [15] use geographic location to partition the network into a 2-D mesh. (2) Each sensor has position information and has uniform sensing range $\sqrt{2}r$ and two transmission

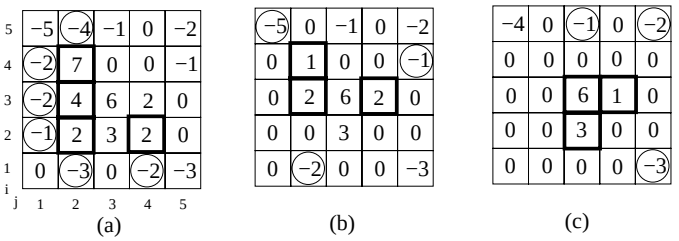


Fig. 3. Local Hungarian method: (a) first iteration, (b) second iteration, and (c) third iteration.

ranges $\sqrt{2}r$ (for intra-grid communication) and $\sqrt{5}r$ (for inter-grid communication). (3) The sensor network is sufficiently dense so that each grid (cluster) has at least one sensor. A leader (clusterhead) in each grid is selected to coordinate activities with leaders of its four neighbors, using inter-grid transmission range. We also assume that each grid knows the global average load already, and hence its take/give state and absent/extra loads. The global average could be a predefined system parameter or achieved by the scan-based approach as in SMART algorithm. The synchronization of the network can be achieved by techniques providing micro-second level synchronization [16].

B. The Local Hungarian Solution in 2-D Meshes

In the local algorithm, each grid in the give state initializes the matching procedure by sending out an "invite" message (IM) to its neighbors, including Δ, which is the difference between its loads and the global average, and a predefined time-to-live data (TTL) which indicates the transmission range of this IM. If the grid that gets the IM is not in the take state, it forwards the IM if the TTL is not expired. A grid in the take state sends a reply message (RM) back when it gets the IM as well as forwarding it, and indicates the load it lacks. When it receives the RM, the give grid sends out some of its loads to the take grid. When a give state grid sends out a IM, the TTL of the message is tunable. At first, a small range is used. If there is no response, the range is increased to search a larger area for take state grids. This procedure is similar to the expanding ring search in ad hoc networks. If a take state grid gets several IMs, it picks the nearest to respond. It may respond to several give grids in case the donation of any one of them is not enough. When a give state grid receives several RMs, it donates its loads according to the order of distance, the closer having priority. The entire procedure contains several synchronized iterations to terminate.

Figure 3 shows the first two processing procedures of the local Hungarian algorithm applied to the example in Figure 1. (a) shows the difference between the load of each grid and the global average, Δ_+ or Δ_- of each grid. The initial TTL is 1, then it is increased by 1 each iteration. The grids shown by the bold boxes donate loads in the first iteration, and the grids with circles are receiving loads. (b) is the resultant network after iteration 1, and (c) is after iteration 2. After 5 iterations, all the loads are balanced. For this single example, the total moving distance of SMART is 63, the Hungarian method is 54, and the local Hungarian method is 56. The number of moves of SMART is 63, the Hungarian method is 26, and the local Hungarian algorithm is also 26. Both Hungarian method and local Hungarian algorithm achieve a complete balanced state, while SMART does not guarantee it. In addition to its huge computation consumption, the Hungarian method needs large storage space, which is $O(N^2)$, where N is the number of nodes. In the other two method, storage space is $O(n^2)$.

Local Hungarian Algorithm

(Initialization) Each grid decides its state and the loads under (Δ_-) or over (Δ_+) the average.

- For a give state grid,
  1) sends out a IM with TTL=1, including its Δ;
     a) If gets RMs, sends out loads according to the distance order and updates loads state and Δ;
  2) If in give state and TTL is smaller than the diameter of the network, expands TTL and goes to (1);
- For a take state grid,
  1) If receives IMs, picks the closest ones to reply;
  2) If receives loads, updates its loads state and Δ.
- For a neutral grid,
  1) If there is a IM, decreases its TTL; If TTL is not 0, forwards the IM, drops it otherwise.

C. Discussions

Timer of the give/take state grids. Each give state grid keeps a timer for IM sending. After an IM is sent out, the timer is set to $2 \times TTL$. The take grid also set the timer of TTL, in which IMs of TTL hops away could arrive. When the timer expires, the take grid makes decision according to all received IMs and sends back RMs. When the timer of a give grid expires, it schedules the loads donation according to all received RMs and sends out loads.

Threshold for state decision. Since the exact global average load may not be an integer, there needs a threshold δ to determine states of grids. That is, if the difference between the loads of the grid and the average is larger than δ, then it is in the give/take state. When δ is set to 0.5, its smallest value, the resultant network will be absolutely balanced, where the difference between any two grids does not exceed 1. However, this may cause extra large communication overhead.

Expansion speed of search range. When the give state grid increases its TTL, it has two options, the linear or exponential expansion. Higher expansion speed needs fewer iterations but more overheads when take and give grids are close to each other, that is, the original sensor distribution is relatively balanced. If local matching is of large possibility in the network, lower expansion speed can achieve small overhead since most searches stop at relatively small ranges.

Termination condition. In the previous section, we use "not in give state or search range larger than the diameter" as the termination condition of the local algorithm. In fact, after the algorithm terminates, the system may not achieve the absolute balance. This is because confliction may occur. This case is common especially when the search expansion is fast. To achieve the absolute balance, we can change the termination condition to "not in give state" and search range stays constant after it reaches the diameter.

Network topology. The proposed Hungarian method and local solution can also be applied to topologies other than the mesh structure. Sensors can be grouped into clusters using any clustering algorithm. Each cluster is viewed as a grid in the mesh structure. The only difference is that each cluster may have up to 6 neighbor clusters in any direction.

IV. Properties

In this section, we discuss the performance metrics of the proposed local solution, the approximation ratio of the solution in terms of the total moving distance.

To calculate the bound of the total moving distance in the worst case, we have to find the worst case. Figure 4 is the case we provide for analysis, where circles with $+$ indicate give state grids and circles with $-$ are take grids. We set d to be the distance and e is a very small number. We also assume that Δ is 1 for every grid. (a) shows the most simple case. Since the middle two grids have a smaller distance, they may match to each other in the first iteration, and the other two grids match. This yields a total moving distance of $4d - 2e$. Obviously, in the optimal way, the first two grids match and the last two grids match, which leads to a total moving distance of $2d$. This lay out can be replicated as in (b). Again, the middle two grids match to force the first and the last grids match. Here, the moving distance is $14d-8e$ while the optimal result is $4d$. We can repeat the replication process as in (c).

If we denote the segment in Figure 4 (a) as the unit segment, $f(k)$ as the length of segment k, and $T(k)$ as the moving distance in segment k using the local solution, we have that:

$$f(k) = 3^{k+1}d - \frac{3^{k+1}-1}{2}e$$

The total moving distance in segment k is the moving distance in the two $k-1$ segments without the final move

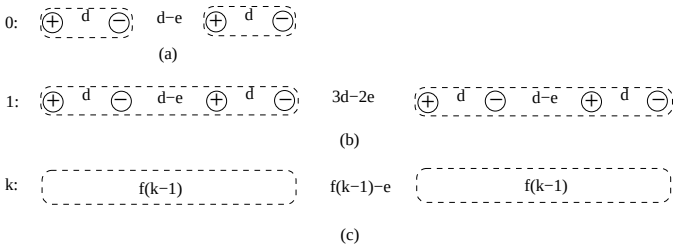


Fig. 4. Illustration of the worst case lay out. (a) The unit segment, (b) segment 1, and (c) segment k.

(the first and last grids match), the middle two grids match, and the first and last grids match. Thus we have that:

$$T(k) = (2 \times 3^{k+1} - 2^{k+1})d - (3^{k+1} - 1)e$$

The optimal result in segment k is $D_{opt}(k) = 2kd$. If we assume that the number of deployed sensors is N, and they are deployed according to the worst case scenario, we have that $k = \lg N - 2$. Thus, the approximation ratio in terms of the number of sensors is:

$$R = \frac{T(k)}{D_{opt}(k)} = \Theta(\frac{N^{\lg 3}}{\lg N})$$

.

This case, which may not be the worst case, shows that there is no constant bound. Therefore, there is no constant bound for the total moving distance in the worst case.

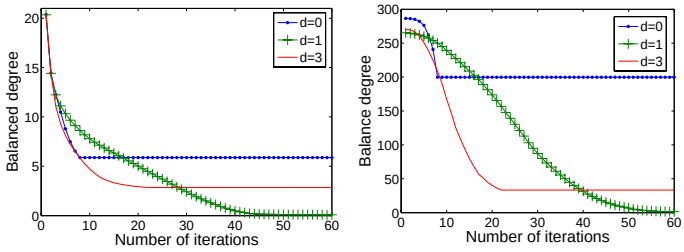
However, we conjecture that the local solution does have a constant probabilistic bound. That is, its expected total moving distance is a constant times the minimal moving distance in random sensor networks. A rigorous proof is lacking due to the complexity of an authentic probabilistic model. Instead we shed some insights with a simplified model.

In the simplified model, we consider give (take) state grids with $\Delta = 1$ and call them givers (takers). Assume N givers and N takers are evenly distributed, such that any square region of side $2d+1$ contains roughly $N_d = (2d+1)^2 N/A$ givers and N_d takers, where A is the area of the sensor network. In addition, givers (takers) are densely deployed such that $N_d \geq 1$ in all cases.

In each iteration of the local solution ($TTL = d$), the probability that each giver does not receive a reply from a nearby taker is $p_d = (1 - \frac{1}{N_d})^{N_d} < e^{-1}$. When the TTL value increases from 1 to D, the expected total moving distance is

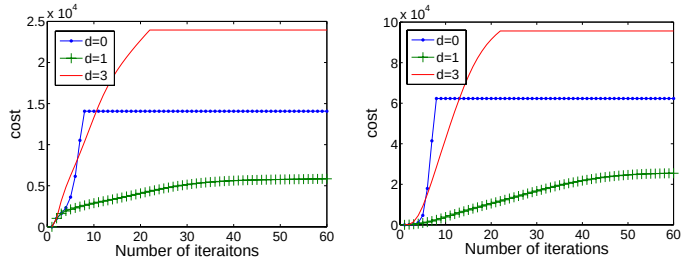
$$\sum_{d=1}^{D}[dN(1-p_d)\prod_{i=1}^{d-1} p_i] < N\sum_{d=1}^{D}\frac{d}{e^{d-1}} < cN$$

where $c = (\frac{1}{1-e^{-1}})^2$. As the minimal total moving distance is at least N, the average total moving distance of the local solution is bounded by c times that of the optimal solution.



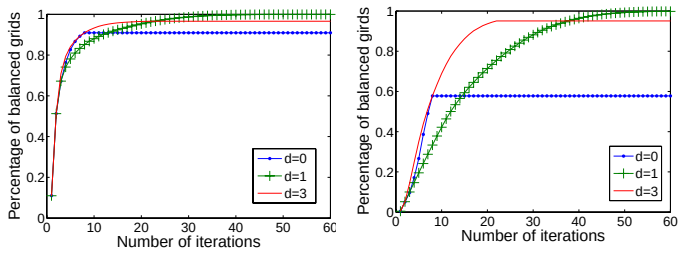
(a) Random distribution (b) Clustered distribution

Fig. 5. Balance degree with difference iteration number in Local Algorithm ($N = 9000, n = 30$).



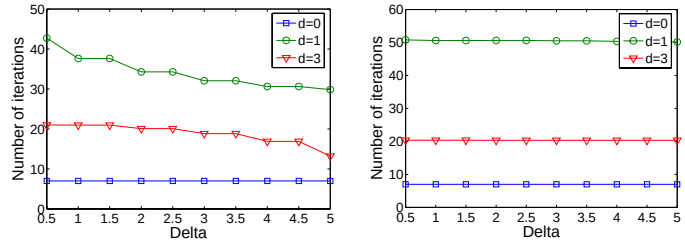
(a) Random distribution (b) Clustered distribution

Fig. 6. Message cost with difference iteration number in Local Algorithm ($N = 9000, n = 30$).



(a) Random distribution (b) Clustered distribution

Fig. 7. Percentage of balanced grids with difference iteration number in Local Algorithm ($N = 9000, n = 30$).



(a) Random distribution (b) Clustered distribution

Fig. 8. The number of iterations for different Δ in Local algorithm ($N = 9000, n = 30$).

V. Simulation

In this section, we present the results of our simulation of the proposed localized movement-assisted sensor deployment method (Local), and the comparison with the Hungarian based optimal method (Hungarian), and the SMART.

A. Simulation environment

All approaches are tested on a custom simulator. We set up the simulation in a $5,000 \times 5,000$ monitoring area. Sensors can be deployed in this area following a given distribution, random distribution (Random) where each sensor randomly selects a position in the area or clustered distribution (Clustered), where sensors are deployed to form one or several clustered areas of different sizes and different clustered degree. The tunable parameters are: (1) The number of grids $n \times n$. We use 10 and 30 as its values. (2) The number of sensors N. We vary N from 100 to 1000 and 900 to 9000 in small and large scale simulations, respectively. (3) The expansion speed of TTL. We increase TTL at a linear and an exponential speed, respectively. When using linear rate, we use 1 and 3 as the step value. (4) The state decision threshold Δ. We vary the value of Δ from 0.5 to 5 to check the performance of Local.

The performance metrics are (a) deployment quality and (b) cost. Deployment quality is shown by the balance degree measured by the standard deviation of sensor numbers in all the grids, and also the percentage of balanced grids. Deployment cost is measured in terms of overall moving distance and also, to a less extent, the number of total moves. The number of messages sent and the iteration number is also examined.

B. Simulation results

We first analyze the performance of Local in a large scaled network, where n is 30 and N is 9000. Figure 5 shows the balance degree of the network after each iteration, $d = 0$ represents exponential TTL expansion, and $d = 1, 3$ are linear expansion with step 1 and 3. (a) is Random, and (b) is Clustered distribution with 3 clustered area. We can see that when the increase of TTL is slower, the more balanced resultant network is achieved, but it takes more iterations to get the stable state. When TTL is 1, and the global average load is an integer, a balanced state can be achieved finally, where almost every grid has the same loads.

Figure 7 is the percentage of balanced grids after each iteration. (a) is Random, and (b) is Clustered distribution. The results are similar with those of Figure 5. The slower the increase of TTL, the more grids achieve balanced state finally, but the lower the speed to balance them, especially in Clustered distribution. We can also see that in most cases, the match procedure stops before the search range reaches the maximum value, the parameter of the network. We can see that in Random distribution, nearly 90% grids are balanced

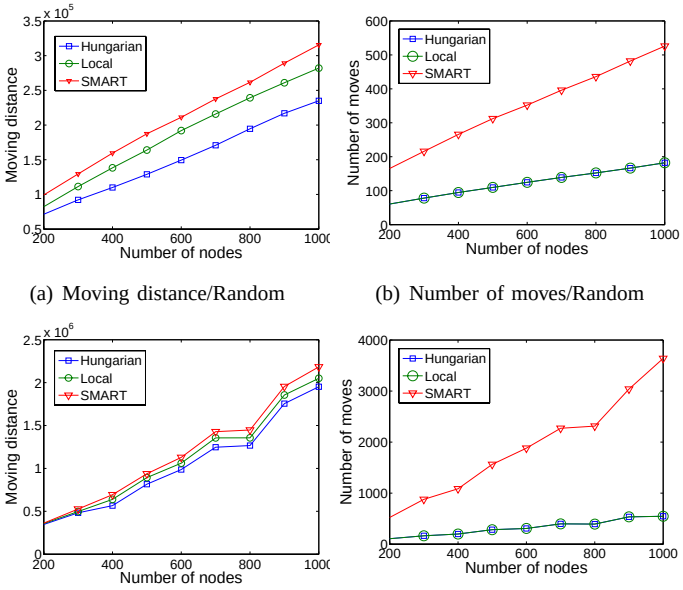
within 10 iterations.

Figure 6 shows the number of messages sent in the system after each iteration. We can see that the message cost is the lowest when $d = 1$, and the highest when $d = 3$. This is because although exponential expansion may cause lots of messages in each iteration, less iterations are needed to increase the TTL to cover the whole area. In Random distribution with TTL=1, each grid sends out less than 6 messages in average.

Figure 8 shows the effect of different Δ value. We can see that for linear expansion TTL, the larger the Δ, the less iterations needed in Random distribution. This decrease in the number of iterations is not significant in Clustered distribution. This is because more grids have large Δ_+ or Δ_-. Although Δ is large, still a certain number of iterations are necessary to achieve balance.

Then we compare the performance of Local with Hungarian and SMART, with $d = 1$ and $\Delta = 0.5$ in a small scale network ($n = 10$). Figure 9 shows the moving distance and the number of moves in Random and Clustered distributions. We can see that Hungarian has the smallest moving distance. SMART has the largest. Although the difference between them is not large, Local is still better than the other two, since Hungarian is global and SMART cannot achieve the complete balanced state. The simulation results of [5] and [1] show that the final balance degree of the SMART algorithm is around 1 in random distribution and larger in clustered distribution. The number of moves of Local is the same with Hungarian, which is smaller than that of SMART.

Simulation results can be summarized as follows: (1) The proposed Local algorithm has better performance than the SMART algorithm in terms of moving distance, the number of moves, and resultant balance degree. (2) The Local algorithm has approximate performance with the Hungarian method, which is a global algorithm and much more energy consuming. (3) In Local algorithm, different TTL expansion methods achieve different performance. The slowest linear expansion method achieves a complete balanced state, but needs more iterations. In most cases, the matching process stops before the search range reaches the whole network.



(a) Moving distance/Random (b) Number of moves/Random

(c) Moving distance/Clustered (d) Number of moves/Clustered

Fig. 9. Moving distance and the number of moves of the three algorithms in different distributions ($n = 10$).

VI. Conclusions

We present a local solution to the movement-assisted sensor deployment problem. This solution is implemented using local network information only. The simulation results show that the local solution has approximate performance with the optimal global solution, and is better than the SMART algorithm.

References

[1] J. Wu and S. Yang, "Optimal movement-assisted sensor deployment and its extensions in wireless sensor networks," in *submitted for publication*.

[2] I. F. Akyildiz, W. Su, Y. Sankrasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, pp. 102–114, August 2002.

[3] A. Howard, M. J. Mataric, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots, Special Issue on Intelligent Embedded Systems*, September 2002.

[4] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, August 1986.

[5] J. Wu and S. Yang, "SMART: A scan-based movement-assisted sensor deployment method in wireless sensor networks," in *Proceedings of INFOCOM*, 2005.

[6] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for target detection," in *Proceedings of WSNA*, 2002.

[7] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Proceedings of Mobicom*, 2001.

[8] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proceedings of INFOCOM*, March 2003.

[9] M. Locateli and U. Raber, "Packing equal circles in a square: a deterministic global optimization approach," *Discrete Applied Mathematics*, vol. 122, pp. 139–166, Octobor 2002.

[10] G. Wang, G. Cao, and T. La Porta, "Movement-assisted sensor deployment," in *Proceedings of INFOCOM*, March 2004.

[11] D. Du, F. Hwang, and S. Fortune, "Voronoi diagrams and delaunay triangulations," *Euclidean Geometry and Computers*, 1992.

[12] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proceedings of INFOCOM*, 2005.

[13] "Dictionary of algorithms and data structures," 2005, http://www.nist.gov/dads/HTML/munkresAssignment.html.

[14] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proceedings of MobiCOM*, 2002.

[15] Y. Xu, J. Heidemann, and D. Estrin, "Geography informed energy conservation for ad hoc routing," in *ACM/IEEE International conference on Mobile Computing and Networking*, 2001.

[16] J. Gruenen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proceedings of ACM WSNA*, 2003.