Compact Genetic Algorithm for Performance Improvement in Hierarchical Sensor Networks Management

Ming-Hui Jin
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
mhjin@mems.iam.ntu.edu.tw

Win-Zing Liu
Tong An Element School, Taoyuan, Taiwan
liuwzz@mail.giga.net.tw

D. Frank Hsu
Dep. of Computer and Information Sciences, Fordham University, LL813, New York, NY 10023, U. S.
hsu@trill.cis.fordham.edu

Cheng-Yan Kao
Institute of Information Industry, Taipei, Taiwan
cykao@csie.ncu.edu.tw

Abstract

This paper introduces a novel scheduling problem called the active interval scheduling problem in hierarchical wireless sensor networks for long-term periodical monitoring applications. To improve the report sensitivity of the hierarchical wireless sensor networks, an efficient scheduling algorithm is desired. Therefore, in this paper, we propose a compact genetic algorithm (CGA) to optimize the solution quality for sensor network maintenance. The experimental result shows that the proposed CGA brings better solutions in acceptable calculation time.

1. Introduction

Sensors have been widely used in various applications (e.g., health, military, home). Traditionally, people apply sensors to detect the status of certain objects. Whenever the detection results diverge from normality significantly, alerts occur and corresponding procedures may be carried out. This workflow at current juncture no longer satisfies many applications because they do require the function of early warning. To reach the goal of early warning, long-term and periodical monitoring becomes necessary. Since data collection is a costly task, it is desirable to have an automatic detection results collecting mechanism in most long-term periodical maintaining. This motivates the design and implementation of novel and advanced sensor network technologies.

Most wireless sensor networks such as the WINS[1], the PicoRadio[2] and the AMPS[3] base their design on an ad hoc (multi-hop) network technology [4, 5, 6] that focus on organizing and maintaining a network formed by a group of moving objects with a communication device in an area with no fixed base stations or access points. Although ad hoc network technologies are capable of constructing a sensor network, the design and implementation of sensor networks for monitoring stationary nodes such as construction sites, historic buildings and bridge areas can be furthered simplified to reduce power consumption and overhead. Based on the nature of immobile nodes, previous studies such as SMAC [7], TMAC[8] and DMAC[9] proposed their new MAC designs for reducing the power consumption of wireless sensor nodes. Although their solutions do reduce the power consumption of sensor nodes, however, their studies do not improve other network issues such as construction cost minimization, routes maintenance, scalability, etc. To efficiently reduce the network construction cost and network maintenance cost, [10] proposed a novel hierarchical sensor network (HSN) architecture.

In the HSN architecture, all the sensor nodes in the sensor network are designed to be equipped with no functions for message forwarding and channel scheduling. To forward the detection results for the sensor nodes to the global control center (GCC), a special purpose device call the local control center (LCC) is introduced in the HSN. Each LCC adopts centralized communication protocols to communicate with the adjacent sensor nodes. A LCC and the adjacent sensor nodes form a cluster and hence the network is partitioned into several clusters. The LCC then forward the detection results reported from the sensor nodes in the same cluster to the GCC. To further reduce the power consumption of the sensor nodes, each LCC not only helps its sensor nodes to learn the correct communication time but also avoid the possibly interferences from other clusters to protect the inner-cluster communications. This approach brings numerous advantages in network construction cost minimization and network maintenance cost minimization. We state the obvious advantages as follow.

1. Since the sensor nodes in the HSN architecture can be designed to be equipped with no functions for message forwarding and channel scheduling, the design and implementation cost of the sensor nodes can be reduced significantly. This implies that the network construction cost can be further reduced if each cluster contains sufficient sensor nodes in average.

2. Since the sensor nodes need not forward messages for other sensor node and the LCC can help the sensor node to learn the correct communication time, the sensor node can correctly turn off its antenna. This significantly extends the lifetime of the sensor nodes and hence significantly reduces the network maintenance cost since the network maintainers can make less effort in maintaining the sensor nodes.

3. Because the number of LCC is much less than the number of sensor nodes, the complexity of the condensed network which contains only the LCCs is reduced significantly. This greatly simplifies the network maintenance cost in routing and multiple access problem issues, especially for large scale wireless sensor networks.

Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05)
1087-4089/05 $20.00 © 2005 IEEE

The advantages above motivates us to study the HSN. Although the centralized communication protocol proposed by [10] efficiently solves the multiple access problems [11] inside each cluster, the interferences between the communications in two adjacent clusters still occur. To avoid the interferences between adjacent clusters, adjacent LCCs should communicate with their sensor nodes in different time intervals. That is, the two adjacent clusters should not be active simultaneously. It implies that the GCC should provide a scheduling mechanism which can specify each LCC a periodical time interval to communicate with its sensor node.
The scheduling mechanism can be evaluated in several dimensions depending on the applications. For static sensor networks, the scheduling mechanisms are mainly measured by the flexibility of report sensitivity determination and rescheduling frequency. The report sensitivity is the frequency of reporting the detection results from the sensor nodes to the GCC. An ideal scheduling mechanism should allow users to flexibly specify the report sensitivity according to the application requirements and minimize the rescheduling activities.

This paper is organized as follows. The architecture and the protocols of the HSN are presented in Section 2. The active interval scheduling problem and its cost model are presented in Section 3. To provide better solutions for the network maintainer, a compact genetic algorithm [12] is proposed in Section 4. The experimental results and comparisons are presented in Section 5 and the conclusions and future works are drawn in Section 6.

2. Hierarchical sensor network architecture

2.1 The system architecture

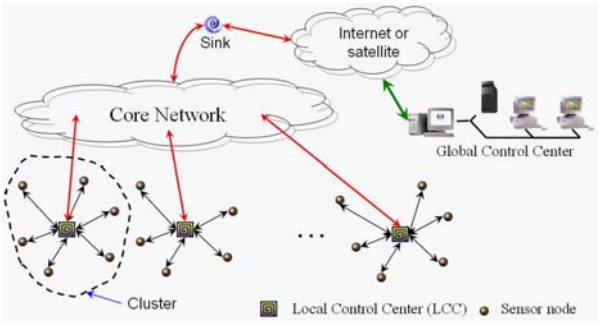

Figure 1. The network architecture for sensor networks with immobile sensors.

Figure 1 shows the proposed network architecture. In this architecture, the sensor network is partitioned into several *clusters*. Each cluster contains several sensor nodes and a *local control center* (LCC). A sensor node has capability to detect and then reports the detection results to its LCC. The detection results are then routed back to the sink through a wireless network called Core Network constructed by only the LCCs. The sink may communicate with the *global control center* (GCC) via Internet or satellite.
Because this architecture is designed for the applications in which all the sensor nodes are immobile, the corresponding protocols can apply centralized approaches to solve the multiple access problems inside each cluster. Specifically, inside each cluster, the LCC applies the polling protocol to communicate with all its sensor nodes. This approach allows each sensor node to equip with no functions for message forwarding or channel scheduling and hence significantly reduce the computation cost and design complexity. This also implies that this approach can largely reduce the implementation cost of each sensor node and hence reduce the cost of building and maintaining a sensor network.

2.2 The power saving mechanism for sensor nodes

Power resource is precious for sensor nodes in many applications [4, 5, 6]. To further reduce the power consumption for each sensor node, the concept of cluster node is introduced. That is, each cluster is a node in the core network. Figure 2 shows the life cycle of a cluster node.

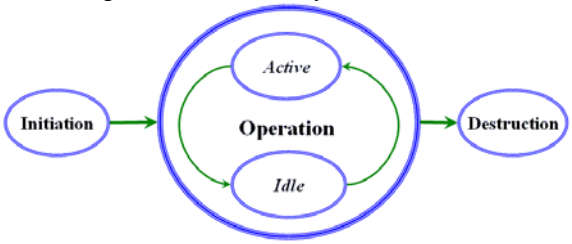

Figure 2. The life cycle of a cluster node.

Whenever a LCC joins this network and completes the initialization procedure, the corresponding cluster node is in the operation stage. In this stage, the cluster node may be in active mode or idle mode. A LCC is allowed to communicate with its sensor nodes if and only if the corresponding cluster node is in active mode. This implies that its sensor nodes can turn off their antenna and idle its CPU whenever its cluster node is in idle mode. This also implies that the sensor nodes can significantly reduce their power consumption in the idle mode.

Although this mechanism significantly reduces the power consumption of all the sensor nodes, however, this also brings several new problems. First, whenever a sensor node turns off its antenna, its LCC cannot communicate with it. In this situation, the sensor node has to correctly set up an alarm clock before its CPU become idle. Therefore, designing a protocol to correctly set the alarm clock for each sensor node is crucial. Second, if two adjacent cluster nodes are in the active mode simultaneously, the communications inside one cluster node may interfere with the communications inside the other cluster node. This implies that adjacent cluster node should not be active simultaneously.

In [10], the authors proposed several protocols to solve the first problem. The second problem is called the active interval scheduling problem in [10]. To facilitate the present of the active interval scheduling problem, we cite the protocols for solving the first problem in Section 2.3. We then state the problem in Section 3 and propose a compact genetic algorithm to solve the active interval scheduling problem in Section 4. Besides, several network topology maintenance

problems are also crucial for supporting this power saving mechanism. Since they do not affect the main issue of this study, therefore readers may reference [10] for more detail regarding the network topology maintenance mechanism of HSN.

2.3 The protocols inside each cluster

In [10], each cluster node is assumed to be active periodically and all the cluster nodes apply the same period. This assumption is held in most long-term periodical monitoring applications. According to this assumption, the concept of detection cycle is introduced and the length of detection cycle is denoted as *ldc*. Figure 3 shows an example of detection cycle of a sensor network. In figure 3, a cluster node becomes active at time t_s and become idle at time t_e in each detection cycle. Therefore, each sensor node of a cluster can set its alarm clock at time t_e and then wake up at time t_s in the next detection cycle. For convenient, the time interval (t_s, t_e) is said to be the active interval of the cluster node.

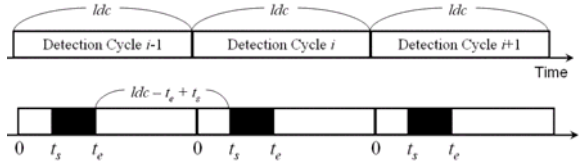

Figure 3. The detection cycle of a sensor network.

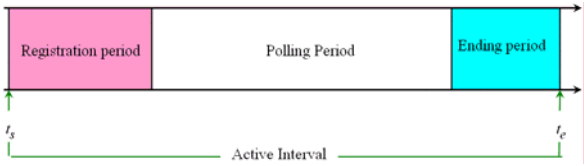

Figure 4. The compositions of an active interval.

In the proposed protocols, each active interval is partitioned into three periods as figure 4. The registration period is used for the sensor nodes which belongs to no clusters and desires to join this network. In the ending period the LCC broadcasts the time of the next active interval of this cluster to all its sensor nodes. The ending period is crucial since it allows the system to adjust the active interval of certain clusters. In the polling period, the LCC sequentially communicates with all its sensor nodes. The readers may reference [10] for more details.

3. The active interval scheduling problem

Scheduling the active interval of each cluster is crucial for the HSN maintenance. There are numerous requirements in the active interval scheduling problem. First, the active intervals of the adjacent clusters should not be overlapped to avoid the possibly interferences between adjacent clusters. Second, the system should allow the network maintainers to specify the length of the detection cycle. Since the network topology is not invariant forever, therefore, the existing schedule may become infeasible whenever a sensor node or LCC joins this network. In this situation, the network maintainer has to reschedule the active intervals of the clusters Since the rescheduling procedure is a costly activity, therefore, extending the lifetime of each active interval schedule is also a key issue in the active interval scheduling problems. In [10], the authors propose an active interval scheduling procedure and we present the procedure in Section 3.1. The procedure takes the above requirements into consideration, with associated to the corresponding optimization problems.

3.1 The active interval scheduling procedure

Figure 5 shows the active interval scheduling procedure. Whenever a new sensor node joins a cluster, the active interval of this cluster becomes longer. In this situation, the enlarged active interval may overlap the active interval of the adjacent clusters and hence make the existing schedule infeasible. Whenever the system perceives that the existing active schedule becomes infeasible, it starts the active interval scheduling procedure to reschedule the active intervals of all the clusters to avoid the interferences among the adjacent clusters. Besides, the system would request the corresponding LCC to block the newly joint sensor nodes until it generates a new feasible schedule.

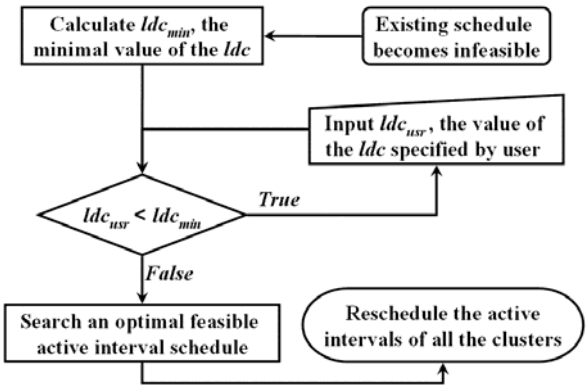

Figure 5. The active interval scheduling procedure.

Activate all the cluster nodes sequentially may be the simplest method in finding a non-interference schedule. Although this approach works, however, when the network contains numerous clusters, the value of the *ldc* may be too large to make the users unacceptable. Therefore, the system should allow the users to specify the value of the *ldc* according to the specification of the corresponding applications. Based on this requirement, we say that an active interval schedule is feasible if it is a non-interference active interval schedule and its $ldc = ldc_{usr}$.

Whenever the user specifies the value of the ldc_{usr}, the system starts to search a feasible active interval schedule. If the value of the ldc_{usr} is too small, there may be no feasible solution. In this situation, the user should try another value. To facilitate the determination of ldc_{usr}, the active interval scheduling procedure first calculates ldc_{min}, the minimal value of the feasible *ldc*. Whenever the system estimates the value of the ldc_{min}, it starts to search the optimal feasible active interval schedule as long as the relation $ldc_{min} \le ldc_{usr}$ is true. Otherwise, it requires the user to specify another value for ldc_{usr}. If the value of the ldc_{min} is small enough, then the users have higher freedom in determining the value of ldc_{usr}.

As long as $ldc_{min} \le ldc_{usr}$, the system starts to search an optimal feasible active interval schedule. The method of searching an optimal feasible active interval schedule given $ldc_{min} \le ldc_{usr}$ has been satisfied is very simple. The readers may reference [10] for more detailed. Therefore, minimizing the value ldc_{min} is the main challenge in the active interval scheduling procedure and hence this paper focuses on solving this problem.

3.2 The definitions and cost model

To facilitate the presentation of the cost model and algorithms for minimizing the value of ldc_{min}, we adopt the following definitions proposed in [10].

Def. 1. $CL = \{C_1, C_2, \ldots, C_n\}$ be the set of all clusters, where n is the number of cluster nodes in the core network.
Def. 2. For each $1 \le i \le n$, cluster C_i contains s_i sensors.
Def. 3. The active interval of C_i is denoted as $(t_s(i), t_e(i))$.
Def. 4. Two clusters C_i and C_j are adjacent if any sensor node in C_i can receive any broadcasted messages from the LCC of C_j to the sensor nodes of C_j.
Def. 5. For two different clusters C_i and C_j, $R_{ij} = 1$ if C_i and C_j are adjacent and $R_{ij} = 0$ otherwise.

Since $ldc_{min} = min\{\ t_e(i) \mid C_i \in CL\}$, and the power saving mechanism presented in Section 2.2 requires adjacent cluster nodes can not be active simultaneously. Therefore, the cost model for ldc_{min} minimization is stated as follow.

Minimize ldc_{min} (1)
Subject to
$\forall\ 1 \le i \ne j \le n, (t_s(i), t_e(i)) \cap (t_s(j), t_e(j)) = \phi$ if $R_{ij} = 1$ (2)
Where
$ldc_{min} = min\ \{\ t_s(i) + s_i \times t_r + t_c \mid C_i \in CL\}$ (3)

4. The proposed compact genetic algorithm

In [10], the authors proposed two greedy algorithms to minimize the value of ldc_{min}. Although the greedy algorithms can rapidly generate low cost solutions, however, the solution quality can be further improved by more sophisticated algorithms. In this section, a compact genetic algorithm (CGA) is proposed to generate each selection which brings lower-cost solutions.

The CGA represents the population as a probability distribution over the set of solutions, and is operationally equivalent to the order-one behavior of the simple GA with uniform crossover. Based on CGA and adaptive rules, the proposed approach consists of global and local strategies by probability-based mutations. CGA operates on a population of potential solutions, applying the principle of survival of the fittest [13] to produce successively better approximations to a solution

4.1 The Algorithm Design Principle

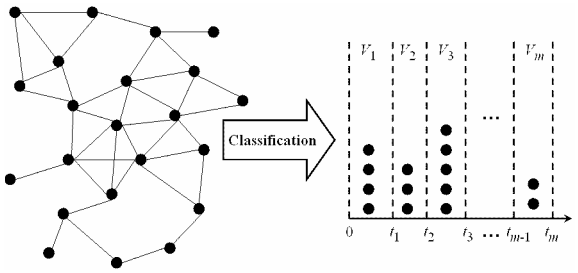

Figure 6. The algorithm design principle.

Figure 6 shows the methodology of the algorithm design for solving the active interval scheduling problem. In figure 6, the set of all clusters are classified into m sets $V_1, \ldots, V_m$. The set V_i is called the i^{th} selection of the solution. The classification only requires that, for each $1 \le i \le m$ and for each $\{X, Y\} \subseteq V_i$, X and Y are not adjacent. Because no cluster interfere any other clusters in the same selection, therefore, the clusters in the same selection can be active simultaneously. In figure 6, the active interval of each cluster in the i^{th} selection is a sub-interval of (t_{i-1}, t_i), where $t_i - t_{i-1} = max\{\ t_e(k) - t_s(k) \mid C_k$ is a cluster in the i^{th} selection$\}$, and $ldc_{min} = t_m$.

4.2 The chromosomes and initial population

Assume that $i - 1$ selections have been determined and there are $n(i)$ clusters $CL(i) = \{C_{\pi(i,\ 1)}, \ldots, C_{\pi\ (i,\ n(i))}\} \subseteq CL$ which are not classified to any selection. In this situation, for convenient, we make the following definitions

Def. 6. A chromosome is defined to be a vector of ordered pairs $<(u_1, p_1), \ldots, (u_{n(i)}, p_{n(i)})>$. For each $1 \le j \le n(i)$, $u_j = 1$ implies that the cluster $C_{\pi(i,\ j)}$ should be selected in the i^{th} selection and $u_j = 0$ otherwise. Besides, p_j is denoted as the possibility of selecting the cluster $C_{\pi\ (i,\ j)}$. And we say that the chromosome selects the cluster $C_{\pi(i,\ j)}$ iff $u_j = 1$.
Def. 7. For chromosome $X = <(u_1, p_1), \ldots, (u_{n(i)}, p_{n(i)})>$, the vector $<u_1, \ldots, u_{n(i)}>$ is called the selection vector of X, the vector $<p_1, \ldots, p_{n(i)}>$ is called the probability vector of X and the set $\{1 \le j \le n(i) |\ u_j = 1\}$ is called the selected index of X.
Def. 8. A chromosome is said to be feasible if, for each two different clusters $C_{\pi\ (i,\ x)}$ and $C_{\pi\ (i,\ y)}$ in $CL(i)$, $C_{\pi\ (i,\ x)}$ and $C_{\pi\ (i,\ y)}$ are adjacent implies $(u_x, u_y) \ne (1, 1)$.
Def. 9. A chromosome X is said to be a corrected chromosome of chromosome Y if X is a feasible and the selected index of X is a subset of the selected index of Y. Besides, we denote CC(Y) to be the set of all corrected chromosomes of Y. A correctness chromosome x of chromosome X is said to be a maximal correctness chromosome of X if, $\forall\ x \in$ CC(X), the number of elements of the selected index of y is greater than or equal to the number of elements of the selected index of x.

All the chromosomes in the each population are generated by two steps. First, determines the probability

vector. Second, applies the probability vector to randomly generate the selection vector. That is, the probability of $u_j = 1$ is p_j. All the chromosomes in the initial generation own the same probability vector $<p_1, \ldots, p_{n(i)}>$ with $p = p_j$ for all $1 \le j \le n(i)$. The parameter p is called the initial probability vector generator (IPVG).

4.3 The genetic operators

Given the set of all chromosomes of a certain population, the proposed CGA applies the crossover and competition operators to generate the probability vectors of all the chromosomes in the next generation.

Competition

The competition operator evaluates the fitness of two chromosomes. Given two chromosomes X and Y, the competition operator generates two chromosomes called *winner* and *loser*. If the fitness of X is higher than Y, then *winner* is set to be X and *loser* is set to be Y. The competition operator applies the following steps to determine the *winner* and *loser*.

Step 1. Randomly generate a maximal correctness chromosome of X.

Step 2. Given the condition that the previous i-1 selections have been determined and all the clusters which are selected by the maximal correctness chromosome of X have be classified into the i^{th} selection, apply the Algorithm 1 to generate an active interval schedule AIS_X.

Step 3. Apply Step 1 and Step 2 to generate an AIS_Y for Y

If the cost of AIS_X is lower than the cost of AIS_Y, then set *winner* to be X and then set *loser* to be Y. Otherwise, set *winner* to be Y and then set *loser* to be X.

Crossover

Given two chromosomes X and Y, the competition operator derives two chromosomes *winner* and *loser*. If the *winner* selects the cluster $C_{\pi(i,j)}$ but the *loser* does not, then the new chromosome should have higher probability to select this cluster. On the other hand, if the *loser* selects this cluster but the *winner* does not, then the new chromosome should apply lower probability to select this cluster. Based on this concept, the crossover operator applies the procedures below to generate a new chromosome from two given chromosomes X and Y.

Step 1. Apply the competition operator to derive the two chromosomes *winner* and *loser*.

Step 2. Let $j = 1$

Step 3. If $j > n(i)$, terminates this procedure

Step 4. Let $P_{win}(j)$ = the probability that the *winner* selects the cluster $C_{\pi(i,j)}$, $P_{lose}(j)$ = the probability that the *loser* selects the cluster $C_{\pi(i,j)}$, $P_{min}(j) = min\{ P_{win}(j), P_{lose}(j)\}$ and $P_{max}(j) = max\{ P_{win}(j), P_{lose}(j)\}$.

Step 5. If the *winner* selects the cluster $C_{\pi(i,j)}$, go to step 8.

Step 6. If the *loser* selects the cluster $C_{\pi(i,j)}$, then set the probability that the new chromosome will select $C_{\pi(i,j)}$ to be $P_{min}(j) - 1/S$, where S is the population size. Otherwise, set the probability that the new chromosome will select to be $P_{win}(j)$.

Step 7. $j = j+1$ and then go to step 3.

Step 8. If the *loser* selects the cluster $C_{\pi(i,j)}$, then set the probability that the new chromosome will select $C_{\pi(i,j)}$ to be $P_{win}(j)$. Otherwise, set the probability that the new chromosome will select to be $P_{max}(j) + 1/S$.

Step 9. $j = j+1$ and then go to step 3.

4.4 The proposed compact genetic algorithm

According to the definitions *Def. 6 – Def. 9*, the proposed compact genetic algorithm for generating the i^{th} selection is stated as follow, where G in Step 3 is the maximal generation of this algorithm.

Step 1. Let $g = 1$.

Step 2. Randomly generate S chromosomes each one of which has the same probability p to select all the remaining clusters.

Step 3. If $g > G$, go to step 8.

Step 4. Let $h = 1$

Step 5. Randomly selects two chromosomes X and Y generated in the g^{th} generation, and then applies the crossover operator to generate a new chromosome in the $(g+1)^{th}$ generation

Step 6. Let $h = h+1$. If $h \le S$, go to step 5.

Step 7. If there are any chromosomes in the $(g+1)^{th}$ generation which definitely select or definitely not select a certain cluster, then let $g = g + 1$ and then go to step 8. Otherwise, let $g = g + 1$ and then go to step 3.

Step 8. Apply the **Algorithm 1** of [10] to generate an active interval schedule for each chromosome in the g^{th} generation. Select the chromosome whose derived active interval schedule has lowest cost and then terminates this algorithm.

The selection vector of the chromosome which is selected in Step 8 indicates the i^{th} selection of the active interval schedule. Therefore, by applying algorithm 2 continuously until $n(i) = 0$, an active interval schedule is generated.

5. Experiments and comparisons

The experiments were performed on an Pentium IV 2.6 GHz PC with 1 GB memory running windows XP operation system. In all the experiments, t_c is 10 time units, t_r is 1 time unit, the population size S of solving each problem is 100, the maximal generation of each experiment G is 60000 and the IPVG is 0.5. Because different hardware brings different value for t_c and t_r due to the hardware response time, this study does not adopt any time unit.

Five benchmark problems proposed in [10] are examined in the experiments. Each benchmark problem contains an adjacent matrix which specifies the value of R_{ij} and a payload table which indicates the number of sensor nodes of each cluster. Readers may reference [10] for more details about the benchmark problems.

Table 1. The experimental results

Problem	Algorithm 1 in [10]	Algorithm 2 in [10]	CGA	
			Cost	CPU time
1	344	328	324	0.37
2	566	548	532	1.54
3	564	557	544	2.79
4	791	773	752	5.59
5	1396	1361	1138	10.29

Table 1 compares the cost of the best solution of problem 1 to problem 5 derived by the greedy algorithms (Algorithm 1 and Algorithm 2 in [10]) and the compact genetic algorithm. The CPU times for generating the best solutions by the greedy algorithms are not presented in Table 1 since they are too small and hence been ignored in this table. The unit of the CPU time is 1 second. The experimental result in Table 1 shows that the CGA can further improve the solution quality in acceptable calculation time.

6. Conclusions and future works

In this paper, we first introduce the proposed HSN architecture for sensor networks with immobile sensors and then introduce a new scheduling problem called the active interval scheduling problem in the HSN sensor networks. In order to find sensitive enough schedule, a compact genetic algorithm is designed and proposed. The experimental result shows that the CGA can further improve the solution quality in acceptable calculation time.

The HSN can also be applied in dynamic sensor network with immobile or group moving sensor nodes. In many construction sits, new sensor nodes and LCCs join and disjoint to the sensor network frequently. This implies that the length of the active interval of each cluster changes frequently. Therefore, designing an efficient mechanism for reducing the re-schedule activities is an important issue.

In [10], the author assumes that all the clusters actives periodically with the same period. However, in some applications, the GCC would like to learn the status of certain areas more frequently. In this situation, the assumption should be relaxed. Therefore, designing a scheduling mechanism for the sensor network with heterogeneous active periods is also an important issue for the sensor network managements. Besides, the proposed scheduling mechanism assumes that the sensor network is a homogeneous sensor network. Because the HSN architecture does not forbid heterogeneous sensor nodes, therefore, formulate a new scheduling problem for heterogeneous sensor networks and design a scheduling mechanism for solving the scheduling problem are also important future works.

7. References

[1] G. J. Pottie and W. J. Kaiser, “Wireless Integrated Network Sensors”, Communication ACM, vol. 43, no. 5, May 2000, pp. 51–58.

[2] J. M. Rabaey, et al., “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking”, IEEE Computer, vol. 33, no. 7, July 20000, pp. 42–48.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks”, Proc. 33rd Annu. Hawaii International Conference on System Sciences, 2000, pp. 3005 – 3014.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks”, IEEE Communication Magazine. Aug. 2002, pp. 102 – 114.

[5] T. J. Kwon, M. Gerla, V. K. Varma, M. Barton, and T. R. Hsing, “Efficient Flooding with Passive Clustering – An Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks”, IEEE Proceeding, vol. 91, no. 8, Aug. 2003, pp. 1210–1220.

[6] T. Y. Lin, and Y. C. Tseng, “An Adaptive Sniff Scheduling Scheme for Power Saving in BlueTooth”, IEEE Wireless Communication Magazine. Dec. 2002, pp. 92–103.

[7] G. Lu, B. Krishnamachari and C. S. Raghavendra, “An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks”, in Proceeding of the 18th International Symposium on Parallel and Distributed Processing. 2004, pp. 224 - 231.

[8] W. Ye, J. Heidemann, and D. Estrin, “Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks”, in IEEE/ACM Transaction on Networking, vol. 12, issue 3, June 2004, pp. 493-506.

[9] Tijs van Dam, Koen Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks”, in ACM Sensys. Nov. 2003.

[10] Ming-Hui Jin, Yu-Cheng Huang, D. Frank Hsu, Cheng-Yan Kao, You-Rui Wu, and Chih-Kung Lee, “On Active Interval Scheduling in Static Sensor Networks”, Proceeding of IASTED International Conference on Communication System and Applications, July 2004. pp. 126 – 131.

[11] Andrew S. Tanenbaum, Computer Networks, Third edition, Prentice-Hall International, INC.

[12] G. R. Harik, F. G.. Lobo, and D. E. Goldberg, “The Compact Genetic Algorithm”, IlliGAL Report No. 97006, Aug. 1997.

[13] M. Stinivas, and L. M. Patnaik, “Genetic algorithms: A survey”, Computer Journal 27(2), pp. 17-26, 1994.