Compact Genetic Algorithm for Active Interval Scheduling in Hierarchical Sensor Networks

Ming-Hui Jin[1], Cheng-Yan Kao[4], Yu-Cheng Huang[5]
Dep. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
[1]mhjin@mems.iam.ntu.edu.tw
[4]cykao@csie.ntu.edu.tw
[5]r91021@csie.ntu.edu.tw

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

Ren-Guey Lee
Dep. of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan
evans@ntut.edu.tw

Chih-Kung Lee
Inst. of Applied Mechanics, National Taiwan University, Taipei, Taiwan
cklee@mems.iam.ntu.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. 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.

Categories & Subject Descriptors: Computer Application → MISCELLANEOUS

General Terms: Algorithm

Keywords: Sensor Network, Active Interval Scheduling Problem

1. NETWORK ARCHITECTURE

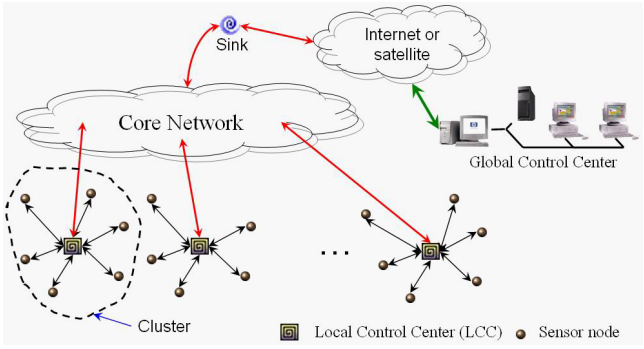

Figure 1. The network architecture for sensor networks with immobile sensors.

Figure 1 shows the network architecture proposed in [1]. 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 the Core Network constructed by only the LCCs. The sink may communicate with the *global control center* (GCC) via Internet or satellite. The sensor nodes are immobile in this study. In each cluster, the LCC applies the polling protocol to communicate with all its sensor nodes.

GECCO'05, June 25-29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

In [1], each cluster is assumed to be active periodically and the entire clusters apply the same period. The period for all the clusters is said to be the detection cycle and the length of detection cycle is denoted as *ldc*. Each cluster is allowed to be active in the same period in all the detection cycle, and the period is called the active interval of the cluster.

2. THE ACTIVE INTERVAL SCHEDULING PROBLEM

We adopt the following definitions proposed in [1] as following.

D 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.

D 2: For each $1 \le i \le n$, cluster C_i contains s_i sensors.

D 3: The active interval of C_i is denoted as $(t_s(i), t_e(i))$.

D 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.

D 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.

D 6: The minimal feasible value of *ldc* is denoted as ldc_{min}.

Since $ldc_{min} = min\{\, t_e(i) \mid C_i \in CL\}$, and the network requires adjacent clusters should not be active simultaneously. Therefore, in [1], 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)

3. THE PROPOSED COMPACT GENETIC ALGORITHM

The methodology of the algorithm design is to classified all the clusters into m sets $V_1, \ldots, V_m$. The set V_i is called the i^{th} selection of the solution. The clusters in the same selection can be active simultaneously. The active interval of each cluster in the i^{th} selection is a sub-interval of (t_{i-1}, t_i), and hence $ldc_{min} = t_m$.

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, we make the following definitions

D. 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 $u_j = 1$.

D. 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.

D. 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 $(u_x, u_y) \neq (1, 1)$.

D. 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).

3.1 The Genetic Operators

The ***competition operator*** generates two chromosomes called *winner* and *loser*, where the *winner* is the one with higher fitness. 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 in [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.

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.

3.2 The Proposed Algorithm

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 [1] 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.

4. 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, 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. Five benchmark problems proposed in [1] are examined in the experiments.

4.1 Experimental Results and Comparisons

Table 1. The experimental results

Problem	Algorithm 1 in [1]	Algorithm 2 in [1]	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 [1]) 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.

5. REFERENCES

[1] 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.