Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A New Genetic Algorithm for the SET *k*-cover Problem in Wireless Sensor Networks

Yun-long Li, Xiao-min Hu and Jun Zhang
Department of Computer Science
SUN Yat-sen University
Guangzhou, P.R. China, 510275
junzhang@ieee.org

***Abstract*—The SET *k*-cover problem is an NP-complete combinatorial optimization problem, which is derived from constructing energy efficient wireless sensor networks (WSNs). The goal of the problem is to find a way to divide sensors into disjoint cover sets, with every cover set being able to fully cover an area and the number of cover sets maximized. Instead of using deterministic algorithms or simple genetic algorithms (GAs), this paper presents a hybrid approach of a GA and a stochastic search. This approach comprises two core modules. The first is the interaction module, which is applied to improve the quality of the population through interaction of individuals. The second is the self construction module, which is a stochastic search procedure running without interaction of individuals. The interaction module is implemented as a combination of selection and crossover, which can efficiently exploit the solutions currently found. The self-construction module includes an adjusted mutation operation and three additional operations. This module is the main force to explore the solution space which can eliminate the inefficiency of using classical GA operations to explore the solution space. Experimental results show that the propose algorithm performs better than the other existing approaches.**

***Keywords*—SET *k*-cover problem, disjoint set covers problem, wireless sensor network, genetic algorithm, coverage**

I. INTRODUCTION

A wireless sensor network is composed of many sensors with limited energy supply. Therefore, it is important to save energy for the sensors to extend their lifetime. One basic idea for energy saving is to separate sensors into different groups such that every group of sensors can handle the job independently. When one group is working, the other groups can be set to sleep.

Usually, the job for the sensors to do is to cover some targets and make sure that the target is under control. There may be many targets to be covered. Usually the position of sensors can be computed carefully to make sure that the sensors can work efficiently, but sometime it is difficult to place the sensors into the positions prearranged. The reason can be various, such as the number of the sensors is too large to be placed one by one or the position is difficult to be approached. So the sensors are randomly deployed and the positions of the sensors are located after the deployment. Cardei and Wu [1] presented a survey to coverage problems in WSNs. According to the category introduced in [1], the problem addressed in this paper belongs to the area coverage problem, which means that the target to be covered is a continuous area. Only considering the sensing mechanism, the objective of the problem is to construct energy-efficient networks and maximize the network lifetime. The problem is described as follows.

Given the positions of a number of randomly deployed sensors, every sensor covers the area within its sensing range. The optimization goal is to find the maximum number of disjoint sets of the sensors, in which every set covers the whole area completely.

As defined in [2], the above problem is called the SET *k*-cover problem, which is also called the disjoint set covers problem [3]. This problem has been proven NP-complete [2][3].

Various approaches have been proposed for solving the SET *k*-cover problem. Slijepcevic and Potkonjak [2] proposed a greedy approach called the "most constrained - minimally constraining covering (MCMCC)" heuristic. MCMCC is greedy in essence and in many cases can not find a good enough solution. A "maximum covers using mixed integer programming (MC-MIP)" heuristic was proposed by Cardei and Du [3]. MC-MIP is infeasible when the scale of the problem is large as the running time of the algorithm increases exponentially. Besides the above deterministic approaches, a GA based approach named "genetic algorithm for maximum disjoint set covers" (GAMDSC) is proposed by Lai et al. [4]. GAMDSC works well only when the number of targets is small. When the number of targets increases, using simple crossover and mutation to explore the solution space as GAMDSC does is ineffective.

A new genetic algorithm (GA) [5]-[15] based approach, which is called the flowing-GA, is proposed to solve the SET *k*-cover problem in this paper. The design goal of the flowing-GA is to apply a more efficient strategy to explore the solution space besides taking advantage of a multi-agent system. The goal is achieved by using a GA interaction module and a self construction module.

The design of the interaction module is straightforward by using the traditional GA operations such as selection and crossover, whereas the design of the self construction module is much more critical.

In the flowing-GA, a ***feasible*** solution is defined as a solution composed of a number of cover sets, of which at most only one cover set cannot cover the whole area completely. This structure is maintained all the time. The self construction module is designed based on the following consideration. For any feasible solution with a number of complete cover sets and an incomplete cover set (if the solution has no incomplete cover set, it can be taken as an empty incomplete cover set), every cover set is considered as a vessel. Sensors can flow from one vessel to another as long as the structure is maintained. A number of flow strategies thus can be carried out to make the flowing of sensors balanced and efficient. The flow of sensors is implemented in a stochastic way in order to inject diversity to the population. This flowing sensors model gives the name "flowing-GA" in this paper.

The proposed algorithm overcomes the limitation in the classical GA and the deterministic approaches. Experimental results show that the flowing-GA outperforms other former approaches in the literature. The analysis made in this paper also provides a deeper understanding of the SET *k*-cover problem by using a factor called "redundancy rate" to estimate the difficulty of the problem.

The remainder of this paper is organized as follows. In Section II, a formal mathematical description of the SET *k*-cover problem and its analysis are presented. Section III gives the implementation of the proposed flowing-GA in detail. In Section IV, experimental results and analysis are provided. Finally, a conclusion is drawn in Section V.

II. Problem Definition and Analysis

A formal mathematical definition of the SET *k*-cover problem is provided as follows:

Let $V=\{v_1, v_2, ..., v_N\}$ defines a set of N sensors, and $Area=\{a_1, a_2, ..., a_N\}$ defines the area that every sensor in V covers. Let I denotes the whole area to be covered. The optimal goal is to find a set $M=\{S_1, ..., S_k\}$ with k maximized, where $S_i \subset V$, $i=1,...,k$, $S_i \bigcap S_j = \varnothing, \forall i \neq j$, $i, j = 1,...,k$ and S_i covers I completely.

Based on the above definition, the area a sensor covers and the target area to be covered can be any shape and size. Some analysis on the problem is made as follows.

A. SET *k*-cover *Problem and K*-cover*age Problem.*

The *K*-coverage problem, as defined in [16], is described as follows. Given an area, if at least k sensors cover every point in the area and at least one point is covered k times, then the area is called ***k*-covered**. The goal of the *k*-coverage problem is to find out k if the area is *k*-covered.

As is described above, the *K*-coverage problem is very similar to the SET *k*-cover problem, but they are different. Fig. 1 provides a simple example to show the distinction between these two problems. In Fig. 1, rectangle DEFG is 2-covered but only get SET 1-coverage. Actually, solving corresponding *K*-coverage problem can provide the original SET *k*-cover problem an upper bound which may not be reached but still useful.

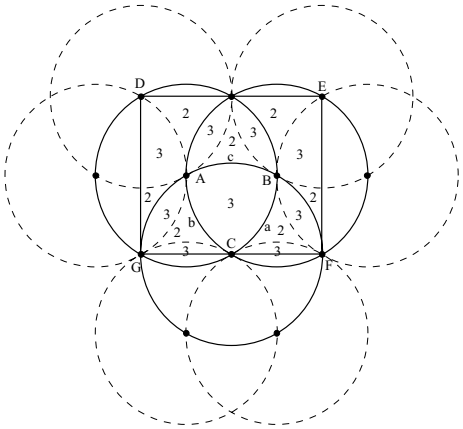

Figure 1. An example of a *k*-covered area but not a SET *k*-covered area. Circles show the area covered by sensors. The number on each field shows the time a field is covered. A, B and C are sensors and a, b and c are 2-covered fields. The square is SET 1-covered but 2-covered.

B. An Influence Factor for Analyzing the Difficulty of the SET *K*-cover *Problem*

It is often taken that the larger the number of sensors is, the more difficult the problem is. But in the SET *k*-cover problem, the number of the sensors is just a factor reflecting the difficulty. Another important factor η, which is called the "redundancy rate", is defined as follows.

Define $\alpha = \sum_{i=1}^{N} S(a_i)$, and let k indicate that the whole area is *k*-covered. Then the redundancy rate is

$$\eta = \frac{\alpha}{S(I)\cdot k} = \frac{\sum_{i=1}^{N} S(a_i)}{S(I)\cdot k}. \quad (1)$$

where α denotes the total area that all the sensors cover. $S(a_i)$ denotes the area covered by sensor i. $S(I)$ indicates the observed area of I. According to the definition, η shows the redundancy rate of sensors on the whole area. The smaller η is, the less redundant sensors are. The lower bound of η is never smaller than 1.

If "I" is a rectangle of $L*W$ and every sensor covers a disk of the same radius R, the expression of η is transformed to

$$\eta = \frac{\alpha}{k\cdot L\cdot W} = \frac{N\cdot \pi R^2}{L\cdot W}\cdot\frac{1}{k} = \frac{N\cdot \pi R^2}{k(L\cdot W)}. \quad (2)$$

Williams [17] proved that the minimum number of M circles to cover a rectangle satisfies the equation as

$$\frac{M\cdot \pi R^2}{L\cdot W} = \frac{2\pi}{\sqrt{27}}. \quad (3)$$

So the minimum number to cover the area k times is $N=kM$. Then the lower bound of η in this situation can be calculated as

$$inf(\eta) = \frac{k \cdot M \cdot \pi R^2}{k(L \cdot W)} = \frac{2\pi}{\sqrt{27}} \approx 1.2092 . \quad (4)$$

If η is very large, the problem can be solved easily and fast even though N is large. But if η is small and N is considerably large, the problem will be extremely difficult to find the best solution. Details will be discussed in section IV.

III. THE PROPOSED FLOWING-GA

In this approach, the two-dimensional area to be covered is modeled as a rectangle of $L*W$ and the area covered by a sensor is modeled as a disk with the same radius R. The term "***field***" is used to indicate the sub area which is separated by the sensing area of the sensors [2]. The upper bound of the SET k-cover problem can be approximated as the result of the corresponding K-coverage problem. In order to facilitate the computation, the rectangle area is divided into grids. After the fields are computed, every field can be taken as a target. The problem is transformed to find a maximum number of cover sets, with every cover set being able to cover all the targets. The calculation of fields can be referred in [2]. The critical fields (these fields being covered minimum times) in a k-covered sensor deployment are also identified. A set of critical lists is thus recorded as the sensors which cover the fields. These works are done as prior computations.

Fig. 2 shows the complete flowchart of the proposed algorithm. Details of the flowchart are explained as follows.

A. Solution Representation for Flowing-GA

Every chromosome is encoded as a list of integers. Each gene of a chromosome represents one sensor (by the index of the gene) and the value of the gene indicates the cover set which the sensor belongs to. Different from the GAMDSC, the representation of chromosomes in the flowing-GA maintains the structure of a feasible solution mentioned above. The sensors having the same gene value in a chromosome must form complete coverage to the area, except for the cover set with the largest gene value. Therefore, the incomplete cover set of a solution, if there is one, is always indexed as the last cover set in its chromosome.

Every individual is represented as a structure {chromosome; fitness; check}. The "fitness" stands for the evaluation value of the individual. The "check" indicates the index of the last cover set, which also stands for the number of the constructed disjoint complete cover sets (cover sets are indexed from 0).

B. Initialization

Initially, a population P composed of NP individuals is created. The chromosomes must be initialized to be feasible solutions, which satisfy the encoding requirement. In the flowing-GA, all the positions of a chromosome are firstly initialized to 0, which means that all sensors are put into a single cover set 0. It must be a complete coverage scheme. Otherwise the deployment of sensors fails.

Then in every chromosome, the redundant sensors in the cover set 0 are randomly moved to form a new set. For the other fields of an individual, the "check" field is set to 1 if the whole set of sensors can make a complete cover set and the "fitness" field is computed as the following evaluation step.

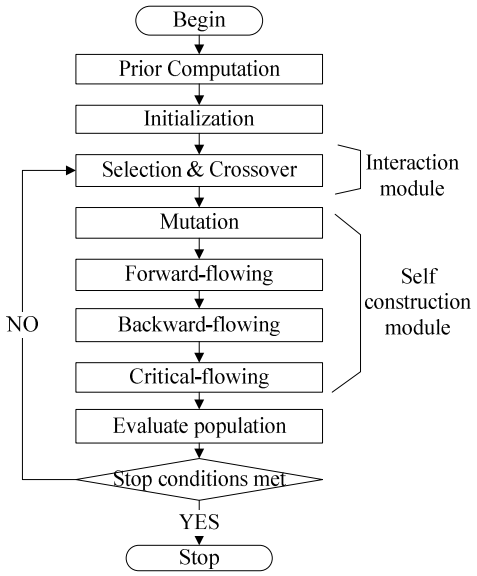

Figure 2. The flowchart of flowing-GA

C. Fitness Evaluation

The fitness function is defined by summing up the total coverage rates of all the cover sets. For an individual p,

$$p.fitness = p.check + coverage\_rate(p.check) , \quad (5)$$

where

$$coverage\_rate(p.check) = \frac{the\ number\ of\ \ fields\ \text{covered by set}\ p.check}{the\ total\ number\ of\ \ fields} , (6)$$

coverage_rate(*p.check*) denotes the coverage rate of the cover set *p.check*, and *p.check* denotes the number of complete cover sets and also the index of the incomplete cover set.

For every individual, the fitness value is updated if the coverage rate of the incomplete cover set equals to 1, which means that the incomplete cover set has become a complete cover set. Therefore, *p.check* is increased by 1, that is, *p.check* = *p.check* +1, which means a new cover set is added and sensors will flow into it in the following generations. The quality of the population is improved as *p.check* increases. In every generation, the individual with the highest fitness value is chosen as the best individual *p*(*best*). If *p*(*best*).*check* equals to the upper bound *k*, the program will be stopped.

D. Interaction Module

The design goal of interaction module is making use of the interaction of individuals to improve the quality of the population. The interaction module is implemented as the combination of selection and crossover. Every time two individuals are selected from the original population, and then a new individ-

ual is created by a uniform crossover. After that, the best one of the three individuals is chosen to the new population. The selection and crossover operation is implemented for generating *NP* new individuals. New individual generated by crossover may not be a feasible solution, in this case it will be abandoned and only its two parents are compared.

E. Self construction Module

The self construction module comprises of a mutation operation and three stochastic greedy search operations, which are the forward-flowing, backward-flowing and critical-flowing. The mutation operation is used for eliminating search bias generated by the greedy search operations. The greedy search operations are designed to construct a solution with more complete cover sets than the original one. Note that the structure of a feasible solution is always maintained. Details are discussed as follows.

1) Mutation

In the flowing-GA, the mutation operation is cooperated with the three additional operations to make the self construction module work like "backtracking". The operation is committed under some conditions. First, it is executed once every *D* generations. During these *D* generations, "forward search" is dominated in optimizing the population. Second, when mutation is performed, the mutation rate *PM* is computed as the number of individuals which have the fitness equal to the best fitness value to the total population size. For each individual, if a randomly generated number in [0,1) is smaller than *PM*, then the sensors in the last cover set of this individual are selected at a probability Δ to flow back to a randomly chosen complete cover set. The structure of a solution is still kept, because only sensors in the incomplete cover set can flow out.

2) Forward-flowing (FF)

The forward-flowing operation is used to select sensors which are redundant (that means if these sensors are removed, the complete coverage is still maintained) from the complete cover sets, and put these sensors into the incomplete cover set. The redundant sensors are flowing from the complete cover sets to the incomplete cover set (Fig. 3).

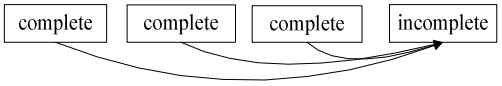

Figure 3. Forward-flowing

The operation is implemented in the following way. For every individual, randomly select a sensor belonging to a complete cover set. If the sensor is redundant, it will be put into the incomplete cover set. This operation is run several times for every individual in every generation.

3) Backward-flowing (BF)

The backward-flowing operation randomly chooses sensors of all cover sets (including the incomplete one) which are redundant and then puts them into randomly selected complete cover sets. That means redundant sensors are flowing from the incomplete cover set into the complete cover sets, or flowing between the complete cover sets (Fig. 4).

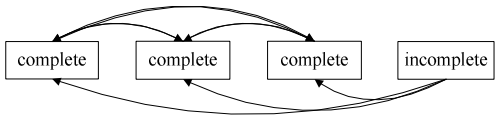

Figure 4. Backward-flowing

The operation is implemented as follows. For every individual, randomly select a sensor. If the sensor is redundant, randomly select a complete cover set which is different from the set that the sensor belongs to. Then put this sensor into the cover set. This operation is also run several times for every individual in every generation.

The difference between the mutation operation and the backward-flowing operation for the incomplete cover set is obvious. In mutation, sensors are selected by a probability, regardless of redundancy. But in the backward-flowing operation, only the redundant sensors can flow out.

4) Critical-flowing (CF)

Each critical list includes the sensors covering a same critical field. Different from GAMDSC, in the flowing-GA, if the incomplete cover set does not have any sensor covering a critical field and there is a redundant one in other cover sets, the redundant sensor will be moved to the incomplete cover set. This operation is used to speed up the construction of a new complete cover set for each individual.

IV. EXPERIMENTS AND ANALYSIS

Two experiments are conducted to examine the performance of the flowing-GA on some moderate test cases and some difficult test cases. The algorithms used for comparison with the flowing-GA are the MCMCC [2] and the GAMDSC [4].

Parameter settings for the experiments are as follows. If not specially stated, all experiments use the same parameters settings as *NP*=3 (for approach flowing-GA), *NP*=100 (for GAMDSC as the author recommended), the maximum value of *G* (number of generations) is not fixed, *D*=100, Δ=0.3, the FF and BF operations in the flowing-GA run 5 times for every individual. The observed area to be covered is a 50*50 rectangle area. All experiments are run on a computer with P4 2.8GHz CPU.

A. Experiments on Randomly Generated Examples

The goal in this experiment is to compare the time consuming and the solution quality for the three algorithms in solving some randomly generated test cases, which is of moderate difficulty (this means most of the algorithms can solve the test cases within the predefine time). There are 9 test cases with a different number of sensors (*N*) or a different radius value (*R*). Every experiment is run 30 times independently. Table I shows the experimental results. The "field" column denotes the number of fields of each test case. The "Best solution" column of GAMDSC shows the best solution found by GAMDSC. The "rate" column shows the rate of finding the upper bound. In this experiment, if GAMDSC cannot obtain the optimal value, the best result in the 200^{th} generation is recorded. In the flowing-GA, once the optimal solution is found,

the algorithm terminates. It can be observed in Table I that GAMDSC runs a significantly longer time than the flowing-GA and the results of GAMDSC are much worse than the flowing-GA. Note that GAMDSC is originally proposed for solving point-coverage problems, which involves a very small number of target points. However, the test cases in this experiment are area-coverage problems with much larger scales of the number of sensors and targets. Table I shows the performance of GAMDSC is unsatisfactory (it fails to find the best solution in 8 of 9 test cases), which indicates GAMDSC is not suitable for solving area-coverage problems. When solving area-coverage problems, both MCMCC and the flowing-GA can find the optimal solutions for the test cases. The time used by the flowing-GA is shorter than that of MCMCC.

TABLE I. TEST RESULTS FOR CASES WITH DIFFERENT *N* OR *R*. ALL SENSORS ARE GENERATED RANDOMLY

Test case				Flowing-GA				GAMDSC				MCMCC	
N	***R***	***Field***	***Upper bound***	***Time(ms)***	***G***	***Solution***	***Rate***	***Time(ms)***	***G***	***Best solution***	***Rate***	***Time(ms)***	***Solution***
1000	5	6076	5	**4911.467**	142.6667	**5**	100	136339.6	200	2	0	332828	**5**
1000	8	2498	17	**7556.767**	313.4333	**17**	100	135678.6	200	6	0	773515	**17**
500	8	2400	7	**2255.8**	174.7	**7**	100	63989.07	200	5	0	71469	**7**
500	10	1586	15	**20611.43**	2155.6	**15**	100	65587.1	200	8	0	115969	**15**
400	10	1556	9	**897.3667**	94.8	**9**	100	48977.6	200	8	0	51234	**9**
400	15	676	23	**965.6667**	133.2	**23**	100	48119.73	200	19	0	75578	**23**
300	15	673	16	**473.4667**	82.6333	**16**	100	29764.13	200	14	0	31469	**16**
300	20	400	32	**611.4**	116.6	**32**	100	30299.43	200	27	0	41469	**32**
100	20	385	7	**43.2**	16.4	**7**	100	278.1333	9.2667	**7**	100	1375	**7**

For the data set with 400 sensors, tests are conducted with the value of *R* increasing from 1 to 20. The number of cover sets found by the three algorithms is shown in Fig. 5. Both the flowing-GA and the MCMCC can find the optimal solutions for all these *R* values so they are drawn together.

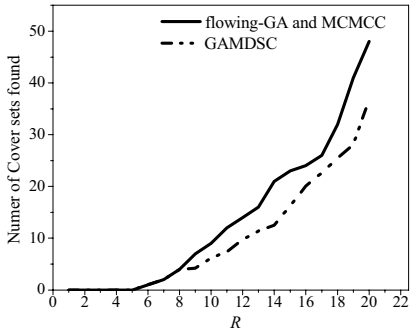

Figure 5. Number of cover sets with *R* increasing from 1 to 20 for the test case *N*=400

Usually in a random deployment of sensors, most fields are covered densely while some others are covered sparsely, because the randomly deployed sensors may not be uniformly distributed. If there are many redundant sensors in the area, complete cover sets are easier to be made. So tests should be conducted to check the situation when there are fewer redundant sensors. The redundancy rate η proposed in section II will be useful.

B. Performance Comparison on Difficult Test Cases

In this experiment, the goal is to compare the performance of the algorithms when the redundancy rate η in the problem is reduced. Test data sets are generated with different η values. They are generated in an iterative way. First a data set is generated with randomly deployed sensors, and then the best solution is found through the flowing-GA. After that the program keeps running for 200 generations to make some redundant sensors flow into an additional set. The sensors in the additional set are removed thus a new deployment case is generated. The procedure goes several rounds, every round generate a deployment case which is less redundant. All the test cases generated in the same procedure share the same upper bound of complete cover sets.

TABLE II. RESULTS OF TEST DATA GROUP 1 AND DATA GROUP 2

	Test case detail			Flowing-GA			MCMCC	
	N	**η**	**UB**	**S**	**Avg G**	**AvgT(ms)**	**S**	**T (ms)**
Data group 1 *R*=5	1000	6.3	5	**5**	177	**5750.533**	**5**	259500
	650	4.1	5	**5**	260	**4320.833**	**5**	62766
	564	3.6	5	**5**	636	**7947.9**	**5**	39015
	482	3.0	5	**5**	2591	25392.2	4	**20812**
	427	2.7	5	**5**	24270	193427.1	4	**12953**
Data group 2 *R*=8	560	5.0	9	**9**	448	**6576.033**	**9**	143078
	372	3.3	9	**9**	795	**5956.267**	**9**	30016
	305	2.7	9	**9**	10315	62883.4	7	**15266**
	280	2.5	9	**9**	40589	224024.5	7	**11250**

In this experiment, two groups of data sets for the tests are generated from different initial data sets independently. Every test case is run 30 times. The results and the η values of the tests are shown in Table II ("UB" means "upper bound", "S" means best solution found). As the GAMDSC works not as well as the other two algorithms, which have been shown in the previous experiments, only the performances of flowing-GA and MCMCC are compared.

As it can be seen from the test results in Table II, the proposed flowing-GA finds the best solutions in all test cases, whereas the MCMCC cannot find the best solutions in test cases N=482, 427, 305, 280. For the cases that both algorithms can find the best solutions, the flowing-GA performs better than the MCMCC by consuming shorter time. The experimental results show that the flowing-GA performs well even in solving difficult test cases. Fig. 6 shows the curves of time used when the value of η decreases. With the best solution unchanged, the smaller η is, the more difficult to find the best solution. A reason for this phenomenon is that when the value of η decreases, the set that a sensor should be put in becomes more and more restricted.

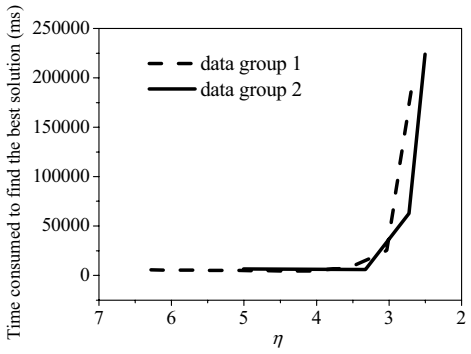

Figure 6. Curves of the time used when η decreases.

V. CONCLUSIONS

In this paper a GA based approach called the flowing-GA is proposed to solve the SET k-cover problem. The flowing-GA is composed of two key modules: the interaction module and the self construction module. With the interaction module, the advantage of the GA as a multi-agent system is used to enhance the quality of the population. The proposed self construction module, which is designed to generate solutions in a flowing model, is much more efficient to generate complete set covers for a solution. A simple example is provided to denote the difference between SET k-cover problem and the K-coverage problem. A factor called "redundancy rate" is proposed in this paper to denote the hardness of a SET k-cover problem. Experimental results shows the algorithm works quite well when applied to the problem.

Although the flowing-GA works well in these test cases, it still needs to be enhanced further. When the problem becomes more difficult, the time consumed by the flowing-GA to find the best solution increases rapidly. For the future work, more controlled heuristic operations should be applied to the algorithm, in order to make the algorithm capable for more difficult SET k-cover problems.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation (NSF) of China under Project 60573066, in part by the NSF of Guangdong under Project 5003346, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, China, in part by the National Natural Science Foundation of China (NSFC) Joint Fund with Guangdong under Key Project U0835002, in part by the National High-Technology Research and Development Program of China no. 2009AA01Z208. Jun Zhang is the corresponding author, e-mail: junZhang@ieee.org.

REFERENCE

[1] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks," Computer Communications vol. 29, pp. 413–420, 2006.

[2] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in ICC, Helsinki, Finland, 2001, pp. 472–476.

[3] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," Wireless Networks, vol. 11, pp. 333–340, 2005.

[4] C.-C. Lai, C.-K. Ting and R.-S. Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications," in IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 3531–3538.

[5] J. H. Holland, Adaptation in natural and artificial system, Ann Arbor, The University of Michigan Press, 1975.

[6] J. Zhang, H. S. H. Chung and W. L. LO, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," IEEE Transactions on Evolutionary Computation, vol. 11, no.3, pp. 326–335, 2007.

[7] J. Zhang, W. L. Lo, and H. S. H. Chung, "Pseudocoevolutionary genetic algorithms for power electronic circuits optimization," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol.36, no.4, pp. 590–598, 2006.

[8] Y. Lin and J. Zhang, "A novel geometric center design method for genetic algorithm optimization," in IEEE SMC 2008, Singapore, 2008, pp. 1446–1453.

[9] X.-M. Hu, J. Zhang, and J.-H. Zhong, "An enhanced genetic algorithm with orthogonal design," in IEEE World Congress on Computational Intelligence (WCCI 2006) and CEC 2006, Canada, 2006, pp. 3174–3181.

[10] J. Zhang, H. S. H. Chung, W. L. Lo, S. Y. R. Hui, and A. Wu, "Implementation of a decoupled optimization technique for design of switching regulators using genetic algorithms," IEEE Transactions on Power Electronic, vol.16, no.5, pp. 752–763, 2001.

[11] Y. Lin and J. Zhang, "An Isoline Genetic Algorithm," in IEEE Congress on Evolutionary Computation, Norway, 2009, pp. 2002–2007.

[12] J. Zhang, J.-H. Zhong, and X.-M. Hu, "A Novel Genetic Algorithm with Orthogonal Prediction for Global Numerical Optimization," in SEAL 2008, LNCS 5361, pp. 31–40, 2008.

[13] J. Xiao, Y.-P. Yan, Y. Lin, L. Yuan and J. Zhang, "A Quantum-inspired Genetic Algorithm for Data Clustering," in IEEE Congress on Evolutionary Computation 2008, Hong Kong, 2008, pp. 1513–1519.

[14] Y. Lin, J. Huang and J. Zhang, "New Evaluation Criteria for the Convergence of Continuous Evolutionary Algorithms," in IEEE Congress on Evolutionary Computation 2008, Hong Kong, 2008, pp. 2431–2438.

[15] T. Huang, J. Huang, and J. Zhang, "An Orthogonal Local Search Genetic Algorithm for the Design and Optimization of Power Electronic Circuits," in IEEE Congress on Evolutionary Computation 2008, Hong Kong, 2008, pp. 2452–2459.

[16] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in WSNA'03, San Diego, California, USA, 2003, pp. 115–121.

[17] R. Williams, The geometrical foundation of natural structure: A source book of design, Dover Pub. Inc., New York, pp. 51–52, 1979.