Sensors Network Optimization by a Novel Genetic Algorithm

Hui Wang[a], Anna L. Buczak[b1], Hong Jin[a], Hongan Wang[a], Baosen Li[c]

[a] *Institute of Software, Chinese Academy of Sciences. wanghui@ios.cn*
[b] *Lockheed Martin Advanced Technology Laboratories, Cherry Hill, NJ 08002*
[c] *Zibo Electric Power Compnay, 61 Xin Cun Xi Lu, Zibo.Shandong, 255032*

Abstract. This paper describes the optimization of a sensor network by a novel Genetic Algorithm (GA) that we call King Mutation C2. For a given distribution of sensors, the goal of the system is to determine the optimal combination of sensors that can detect and/or locate the objects. An optimal combination is the one that minimizes the power consumption of the entire sensor network and gives the best accuracy of location of desired objects. The system constructs a GA with the appropriate internal structure for the optimization problem at hand, and King Mutation C2 finds the quasi-optimal combination of sensors that can detect and/or locate the objects. The study is performed for the sensor network optimization problem with five objects to detect/track and the results obtained by a canonical GA and King Mutation C2 are compared.

1 Introduction

During the last four decades there has been a growing interest in algorithms that rely on analogies to natural phenomena. One type of such algorithms is the Genetic Algorithms (GAs) that imitate the principles of natural evolution [9, 7]. GA has been widely used for combinatorial optimization, structural design, scheduling and other engineering problems [8, 13].

In this paper we are approaching the problem of optimization of a sensor network by Genetic Algorithms from a practical standpoint: we are interested in obtaining the quasi-optimal solutions fast. The sensor network is comprised of randomly distributed unattended ground sensors that are remotely deployed and after deployment their location is known. Objects in a space are monitored by limited numbers of those low cost - low power sensors. The advantages of using several of those sensors outweigh the expected performance degradation since a system of several inexpensive sensors in the same area offers a redundancy that provides acceptable performance. The complete system consists of modules that perform self-organization, object tracking, track fusion, ID fusion, communication,

[1] This work was performed by Anna L. Buczak when she was with Honeywell Laboratories.

etc. [4]. This paper focuses on the optimization of sensor selection performed by genetic algorithms in the Self-Organization module.

2 Optimization of a Sensor Network

We are performing optimization of a sensor network. The network is comprised of remotely deployed unattended ground sensors that can be used for object detection, tracking and identification. A sensor can be used for tracking an object, if this object resides in the sensor's field-of-view (FOV) and if the sensor is turned on. The sensor network adapts its structure in order to achieve the goals specified by a human. Sensor selection is often performed in order to minimize the power consumption of the sensor network, by choosing the sensors that need to be turned on or off at a given moment in time.

The goal of optimization is to find sensors for tracking all the objects identified in network objective (that can be seen as the optimization goal) in a way that optimizes certain metrics. In case of object tracking two metrics should be optimized: the accuracy of object tracking and the power utilization of the sensor network. This multi-objective optimization is performed by Genetic Algorithms. For each object identified in network objective, optimization has to find *m* sensors needed for accurate tracking of objects. The value of *m* depends on the physical characteristics of the sensors used.

Our problem falls in the category of combinatorial optimization problems: the system has to choose tuples of sensors that need to be on. There is a need of one tuple per object and the same sensor can be used for multiple objects as long as these objects are within its FOV. If we have *k* objects and we need *m* sensors per object, 1 to *k* m-tuples are needed. The size of the search space is described by:

$$SearchSpace = \prod_{i=1}^{k} (NumberOfMTuplesForIthObject) = \prod_{i=1}^{k} \binom{n}{m} = \prod_{i=1}^{k} \left(\frac{n_i!}{(n_i - m)! \cdot m!} \right) \quad (1)$$

where ni is number of sensors that can detect object i. The search space is exponentially increasing with the number of sensors and objects, discontinuous, with non-ordered (feature type) parameters.

3 Internal GA Structure for Sensor Network Optimization

In our design each individual of the Genetic Algorithm population is comprised of several genes. Each of the genes contains on sensor's identification. All the sensors, which are chosen by GA to be active at a given moment, have their identification coded in the genes. There is a unique identification associated with each sensor and the genes use a binary encoding for identification.

The GA's internal structure (i.e. number of genes) depends on Network Objective. Whenever this objective changes, the number of genes of the GA also changes. Network Objective includes a list of suspected objects and required operations associated with them. If the operation is to locate the object, there are as many genes as necessary for location, for example in case of acoustic bearing sensors this number is three (Fig1).

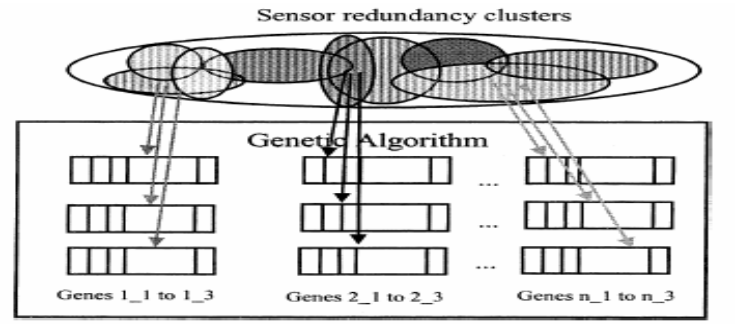

Fig. 1. Internal structure of GA for location

When performing object tracking we encounter a multi-objective optimization problem. The fitness function of GA takes into account both objectives: maximization of the location accuracy (i.e. minimization of the position tracking error) and minimization of the network power consumption. The fitness function has the following form:

$$Fitness = -(w_1 \cdot \sum_{i=1}^{k} E_i + w_2 \cdot \sum_{j=1}^{l} P_j + \sum_{i=1}^{k} PenaltyFor\,Each\,E_i\,Exceeding\,Threshold) \quad (2)$$

where E_i (i=1,2,…, n) are the estimated position errors for i-th object and P_j (j=1,2,…, m) are the power consumption of j-th sensor, k is the number of objects, l is the total number of selected sensors and w_1 and w_2 are weights. The last term is a penalty added for each of position errors exceeding a predefined threshold. This penalty increases significantly the range of population fitness and thus improves GA convergence but solutions that exceed the penalty are still valid. For estimating the position errors (E_i), we are using the GDOP error [6]. The smaller the GDOP error of a sensor triplet, the better the position accuracy of the object will be achieved.

4 Genetic Algorithm with Special Reproduction Operators

The difficulties inherent in GA design are to determine the stopping criterion, the proper GA population size, probabilities of crossover and mutation. The difficulty

in determining the stopping criterion comes from the fact, that GA convergence is problem dependent [6,8,15,17]. Wolpert et al. [15] presented a number of "no free lunch" (NFL) theorems and established that for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class. Our goal is to obtain a quasi-optimal solution in the shortest possible time for the sensor network optimization problem. We make no claims in this paper to the generality of the GA developed and its speed of convergence for other problems.

4.1 Genetic Algorithm with King Strategy

The King Genetic Algorithm that we developed has been inspired by the reproduction process of the bees. There are three kinds of bees: the queen, worker bees, and drones. If mated with drones, the queen's eggs will become worker bees, otherwise they will become drones. In bees' colonies the queen plays the most important role in generating the offspring: only she can lay eggs. Inspired by this phenomenon, a novel GA that we call King GA, was proposed [14]. In King GA, a special individual, the best individual in the population, is always selected in the reproduction process to be one of the parents. This reproduction process is shown in Fig. 2a.

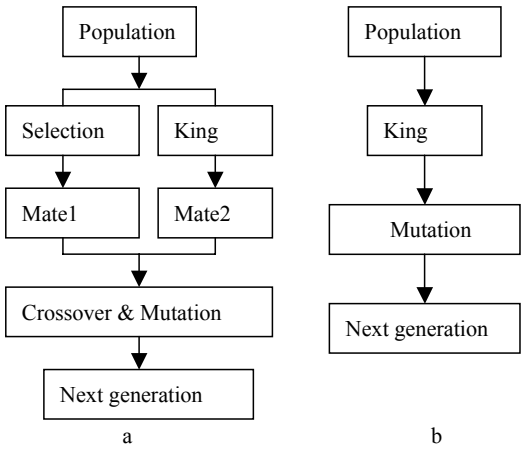

Fig. 2. : The reproduction in a) King GA; b) King Mutation.

4.2 Special Mutation Operator

In Genetic Algorithms, mutation was first introduced as an auxiliary operator to ensure population diversity. Many papers [1,6] pointed out the importance of

mutation, but the mutation methods proposed were very similar, the difference merely being the value of the mutation rate, or whether the rate was constant or adaptive. Our previous experiments with GAs [5] showed that when mutation performs a strong enough search, crossover is not necessary for finding the optimum of multi-modal functions with non-ordered parameters. Therefore we proposed King Mutation, a version of King GA in which only mutation takes place. The reproduction process of King Mutation is shown in Figure 2b.

Mutation in GAs is the process by which one or more genes in an individual are modified. Generally, each gene is chosen for mutation with a probability of mutation P_m that is determined in the initialization step of the genetic algorithm. In the new mutation operator that we are proposing, called Mutation C2, exactly two chromosomes of an individual are randomly selected to be mutated. Any number of genes in a chromosome may undergo mutation. Each gene in a chromosome to be mutated is mutated with probability P_m. King Mutation algorithm with only the mutation of type Mutation C2 is called King MutationC2.

King GA with Mutation C2 is similar to Evolutionary Strategies [16], ES(1+λ). In ES(1+λ) algorithm, there is only one parent which is the best individual in the population; the parent generates λ children; in the next reproduction process, the best individual from the parent and its λ children is selected as the new parent to generate children. So King GA with Mutation C2 is very similar to ES(1+λ), but the mutation method is quite different. In ES(1+λ), the main reproduction operator is Gaussian mutation, in which a random value from a Gaussian distribution is added to each element of an individual's vector to create a new offspring.

There are some GA studies [2,10,12] which are similar to the GA we proposed here. Jones' Crossover Hill-climbing algorithm proposed in [10] is similar to King GA. He compared several algorithms such as Standard GA, Bit-flipping Hill-climbing, and Crossover Hill-climbing. Crossover Hill-climbing algorithm with only one step (CH-1S) obtained the best result. Both CH-1S and King Mutation C2 have no crossover; they both employ only mutation operators and their mutations are quite different from the traditional mutation method. Another similarity is that in both algorithms, the best individual in the population is used for generating offspring. However the mutations performed in King GA with Mutation C2 and in CH-1S are dissimilar; another difference is the population size: CH-1S has a population of 2 individuals only and King GA has a larger population.

5 Experiment Descriptions

In an attempt to examine the quality of the GA proposed, we performed a set of experiments that compared the performance of King Mutation C2 and canonical GA on optimization of a sensor network for five objects. In the experiments performed we used an area of 25 by 25 kilometers with 81 sensors uniformly

distributed. Each sensor's FOV is a circle with a radius of 5 kilometers and there are about 20 sensors that can detect each object. For each of the experiments performed the Percentage of Total Search Space (PTSS) covered by GA was computed using the following equation:

$$\text{Percentage Total Search Space} = 100 * FFE / SS_n \% \quad (3)$$

where SS_n is the whole search space for n objects and the number of sensors identified above. FFE is the actual number of fitness function evaluations performed by GA.

Effectiveness is used to compare the performance of different GAs. For each set of the experiments performed with the same values of n and P the Effectiveness was computed as:

$$\text{Effectiveness} = \text{Number of Optimal Runs} / \text{Total Number of Runs} \quad (4)$$

The experiments with different population size are listed on Table 1. For canonical GA, the crossover rate is set to 0.9 and the mutation rate is set to 1/IndividualLength; for King Mutation C2, the crossover rate is 0 and the mutation rate is set to 1/ChromosomeLength. We performed experiments with population sizes: 5, 10, 20, 50, and 100. For each population size a canonical GA and King Mutation C2 was run 30 times and the results in Table 1 are the average of those runs. Both methods have the same stopping criterion, the algorithms stop iteration if there is no improvement in the fitness function after a certain number of consecutive generations (This number is 5000 in our experiment).

Table 1: Experiment results for 5 objects

	GA Method	Generation#	Fitness	PTSS	Effectiveness
P=5	King Mutation C2	5505	-551.35	5.76E-12	0.80
	GA	8565	-889.86	8.95E-12	0.00
P=10	King Mutation C2	6147	-530.54	1.29E-11	0.95
	GA	10151	-672.98	2.12E-11	0.00
P=20	King Mutation C2	5857	-529.34	2.45E-11	1.00
	GA	11453	-642.60	4.79E-11	0.15
P=50	King Mutation C2	5345	-529.34	5.59E-11	1.00
	GA	11228	-562.07	1.17E-10	0.20
P=100	King Mutation C2	5281	-529.34	1.10E-10	1.00
	GA	10142	-548.54	2.12E-10	0.20

P: Population size;
Generation#: Number of generations.

Canonical GA results are pretty poor for small population sizes. With increasing population size, the fitness achieved by canonical GA becomes closer

to the optimum. The best effectiveness achieved by canonical GA is for the largest population (100 individuals) and is only 0.2 meaning that it is very difficult for the canonical GA to perform optimization for a sensor network with five objects.

Results of King Mutation C2 are much superior to those of a canonical GA: it can obtain quasi-optimal solutions with high probability, the effectiveness being 0.8 and 0.95 for populations of size 5 and 10 respectively. Its effectiveness becomes 1 for population sizes of 20 or larger.

Consistently for each population size, King Mutation C2 gave a better result than the canonical GA: a much higher effectiveness and a higher fitness value. King Mutation C2 also covered roughly two times smaller search space (PTSS) than the canonical GA in each case. Small PTSS is very important in real-world applications since it leads to the reduction of the computation time, allowing for a real time application of the algorithm.

5 Conclusion

This paper describes a system performing self-organization of a sensor network. The goal of the system is to choose sensors necessary to perform object detection or tracking while minimizing the power consumption of the entire network. In this paper, special emphasis is placed on the optimization performed by genetic algorithms.

The exponential grow of the search space (with the increasing number of sensors and objects) makes the problem intractable for most optimization techniques in a reasonable time frame. Genetic Algorithms are chosen for the task, given their high robustness in complex search spaces. In case of multi-objective optimization problems, such as object tracking, convergence is much more difficult to achieve. With the increasing number of objects, Effectiveness of canonical GAs is rapidly decreasing. The increase of GA search space makes the genetic search of the standard genetic algorithm inefficient and consequently the computation time needed for convergence becomes very large. This makes it necessary to improve the canonical genetic algorithm to speed up the convergence of the algorithm when the number of objects increases.

We proposed a novel Genetic Algorithm with King selection strategy that somewhat imitates the reproduction process of bees. The new King selection strategy, especially when coupled with a new mutation operator (Mutation C2) significantly improves the performance of GA for the optimization of sensor network. The new algorithm is very robust, giving good results for a wide range of population sizes. This is in contrast with traditional GAs where it is very difficult to set the value of population size, crossover and mutation rates.

References

1. H. Aguirre, K. Tanaka, “Parallel Varying Mutation Genetic Algorithms”, Proceedings of the Congress on Evolutionary Computation, Hawaii, USA, May 2002.
2. S. Areibi, “An Integrated Genetic Algorithm With Dynamic Hill Climbing for VLSI Circuit Partitioning”, Genetic and Evolutionary Computation Conference (GECCO-2000), Las Vegas, Nevada, July 2000, IEEE
3. T. Blickle, L. Thiele, “A Comparison of Selection Schemes used in Genetic Algorithms”, Swiss Federal Institute of Technology. TIK-Report 1995.
4. L. Buczak, H. Wang, H. Darabi, M.A. Jafari, “Genetic Algorithm Convergence Study for Sensor Network Optimization”, Information Sciences. Pages 267-282. Volume 133, Issues 3-4. 2001.4.
5. L. Buczak, H. Wang, “Optimization of Fitness Functions with Non-Ordered Parameters by Genetic Algorithms”, Congress on Evolutionary Computation ’2001, Korea.2001.5
6. K. Deb, S. Agrawal, “Understanding Interactions Among Genetic Algorithm Parameters”, Foundations of Genetic Algorithms5, W. Banzhaf, C. Reeves (eds.), Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1999.
7. K. De Jong. “Genetic algorithms are not function optimizers”. Foundations of Genetic Algorithms 2, pages 5--17, San Mateo, CA, 1993. Morgan Kaufmann.
8. D.B. Fogel, Evolutionary Computation – Toward a New Philosophy of Machine Intelligence, IEEE Press, 1995.
9. J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975
10. Terry Jones, “Crossover, Macromutation, and Population-based Search”, Proceedings of the Sixth International Conference on Genetic Algorithms. July 15-19, 1995
11. Kadar, “Optimum Geometry Selection For Sensor Fusion”, Signal Processing, Sensor Fusion and Target Recognition VII, I. Kadar (ed.), SPIE Vol. 3374, pp.13-15, The International Society for Optical Engineering, Bellingham, 1998.
12. Bing Li, Weisun Jiang, “A Novel Stochastic Optimization Algorithm”, IEEE Transactions on System, Man, and Cybernetics, ---Part B: Cybernetics, Vol. 30. No.1, February 2000.
13. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.
14. H. Wang, A. Buczak, H. Wang, “A Novel Genetic Algorithm with King Strategy”, ANNIE 2003, St. Louis, USA. 2003.11
15. D. H. Wolpert and W. G. MacReady. “No free lunch theorems for optimization”. IEEE Transactions on Evolutionary Computation, April 1996.
16. Hans-Paul Schwefel, Evolution and Optimum Seeking. A Wiley-Interscience Publication. John Wiley & Sons, Inc. 1994.
17. M. Srinivas, L.M. Patnaik, “Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms”, IEEE Transactions on Systems, Man and Cybernetics. Vol. 24, No.4, pp. 656-667, April 1994.