SELF-ORGANIZATION OF A HETEROGENEOUS SENSOR NETWORK BY GENETIC ALGORITHMS

ANNA L. BUCZAK, VIKRAM R. JAMALABAD
AlliedSignal, Inc., 101 Columbia Road, Morristown, NJ 07962-1021

ABSTRACT:
This paper describes a novel method for organizing a randomly distributed group of sensors. The group is heterogeneous in the sense that it contains sensors of different types that can detect either unique or multiple target types. 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 targets. 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 targets. We approach the problem in two phases. First, sensors are clustered according to morphology and location. Next, a genetic algorithm determines the optimal combination of sensors to achieve a given objective. The genetic algorithm restricts reproduction, crossover and mutation operations to sensors belonging to the same cluster. We devised a chromosome decoder that changes the original constrained optimization problem into an unconstrained problem. The technique developed is tested on a simulated data set and the results are described.

1. INTRODUCTION

We are dealing with a network of randomly distributed unattended ground sensors (UGS). These sensors are remotely deployed and after deployment their location is known. The sensor network adapts its structure in order to achieve the goals specified by the commander in the mission objective. The complete system consists of modules that perform self-organization, target localization, target tracking, track fusion, Id fusion, communication etc. Self-organization can be viewed as the brain of the sensor network because it directs the other modules. This paper deals exclusively with two of the self-organization modules: clustering and optimization.

2. PROBLEM DESCRIPTION

The system being developed performs self-organization of a sensor network in order to achieve a given mission objective. The mission objective, specified prior to mission execution, defines the surveillance goal of the mission. The mission statement encompasses a description of the geographical area for surveillance and the targets of interest. A soldier issues the mission objective. Mission objectives can range from target detection, to location and identification. Targets include, but are not restricted to tanks, trucks, helicopters, and people. The sensor network adapts its structure in order to achieve the goals specified in the mission objective while minimizing the overall power consumption of the network. It is envisioned that the system may optimize additional network functions later.

The main goal of the self-organization is the optimization of the sensor network. Self-organization has to choose, at each moment in time, the set of sensors that need to be on (and transmit data to the other system modules). From the viewpoint of optimization, we have a multi-objective problem with no unique solution. Therefore, only a Pareto-optimal solution can be achieved. We deal with multi-objective optimization because sensors are chosen to achieve the highest accuracy of detection, location and identification of targets, while minimizing the energy used by them at each moment. The system performs combinatorial optimization choosing tuples of sensors that need to be on. If we need at most one sensor per target, each k-tuple is a possible solution, where k is the number of targets. The number of combinations is described by:

$$\sum_{i=1}^{k} C_i^n = \sum_{i=1}^{k} \frac{n!}{(n-i)!i!} \quad (1)$$

where n is the number of sensors, and k - the number of targets. The resulting search space is of exponential complexity. Figure 1 shows the increase in search space for a problem with 24 targets. The search space itself is non-convex and discontinuous, with mixed-integer parameters and feature type parameters.

There are several analytic and numerical optimization techniques [Cha-97], such as linear and nonlinear programming (SLP, SQP), gradient methods, constraint logic programming and simulated annealing. Unfortunately most of them cannot solve this problem. The two techniques capable of solving the complex optimization problem at hand are simulated annealing and genetic algorithms. We choose to use genetic algorithms because they have been proven to be very useful for optimization due to their ability to efficiently use historical information to obtain new solutions with enhanced performance [Gol-89]. Genetic algorithms are also theoretically and empirically proven to provide a robust search in complex search spaces and they do not get trapped in local minima as opposed to gradient descent techniques.

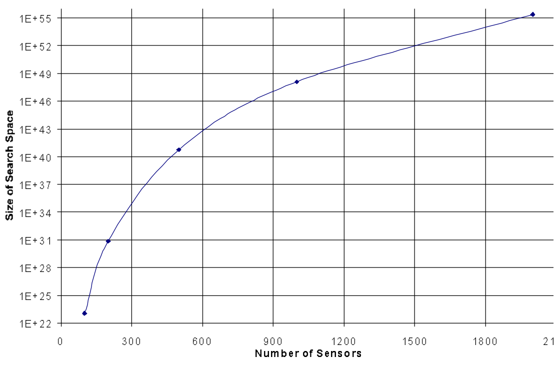

Figure 1. Search space size *vs.* number of sensors.

3. SYSTEM DESCRIPTION

The self-organization system consists of three modules: Initialization, Network Objective Adaptation and Optimization. Initialization prepares the stage for all the other self-organization modules. After sensors are deployed, their location and status are fed back to the initialization module. The module responds by executing the Confidence Factor Assessment sub-module, which provides the commander with visual and numeric measures of the nominal coverage of the space of interest (specified in the mission objective). The network's ability (or inability) to detect or locate targets in all parts of the relevant space provides an early decision point to the commander. While this is being displayed, the Clustering sub-module organizes the sensors into clusters. These clusters discretize the search space and enable the optimization module to run in an efficient manner.

Network Objective Adaptation is a module responsible for translating the broad Mission Statement issued by the commander into a precise Network Objective that will be used by Optimization as its goal. The Network Objective is only valid at a specific moment in time and it evolves rapidly as various events happen in the environment.

The Optimization module is responsible for optimizing the sensor network to a given goal defined by the Network Objective. Optimization of the network architecture to the network objective is performed by a set of Genetic Algorithms (GA). They will be described in detail in sections 6 and 7.

4. CLUSTERING MODULE

The system performs sensor clustering in order to reduce the search space for the optimization algorithm. In a broad sense, clustering is the organization of all the sensors into cohesive groups. Some cardinal rules are adhered to in this procedure. One rule is that sensors that only detect the same type of target can be part of the same cluster. The premise of the clustering procedure is that there is a set of targets in the considered space that are to be located and monitored. The aim is to determine an appropriate set of sensors that can achieve this result while sensors that are out of range of the target are ignored.

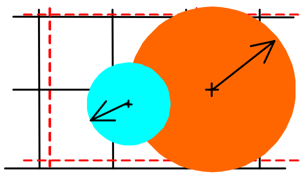

Figure 1: Cluster membership grids. Solid lines refer to a grid for the sensor to the left while dashed lines are a grid for the sensor to the right.

The solution proposed in this paper is to begin by partitioning the area of interest by a temporary grid. The size of the grid is based on the detection range of the particular sensor or sensors being considered. Each cluster of sensors is then associated with a contiguous topological grid space. Clustering is thus based on physically defined regions. The method is applicable to a space of any size or shape. Clusters are target based, *i.e.*, restricted to sensors that can detect the same type of target. Co-located sensors detecting different types of targets will belong to different (albeit overlapping space) clusters. This rule is enforced to ensure a coherent target response. Grid size is determined by a factor times the minimum detection range of the sensors that are able to detect targets of the same type. This allows target-based sensor selection, utilizing only those sensors that can sense a particular target. The range within which an object can be detected by a sensor has been defined as the range vector of that sensor.

The clusters that result from this grouping are fuzzy. A sensor belongs to a particular cluster if the disk defined by its range vector overlaps the grid space associated with that particular cluster. Graphically, consider the smaller disk to the left in Figure 2. The disk represents the sensor's range (the portion of the area that it can "see"). The solid-line grid spaces refer to the clustering grid that has been applied to the particular geographical area. Of the 6 grid spaces shown fully, the sensor overlaps 4 spaces. The sensor, then, is a member of 4 clusters, and it is able to detect targets located in portions of each of these clusters. Figure 2 also shows a larger disk, *i.e.*, a sensor with a larger range, which also happens to detect a different type of target. This larger range sensor is associated with the grid spacing of the dashed lines, which relate to a different type of target. This sensor range intersects two grid spaces and is a member of two clusters that overlap the four clusters of the smaller range sensor. It is important to note that grid spaces for different types of targets can overlap but grid spaces for the same type of target cannot overlap.

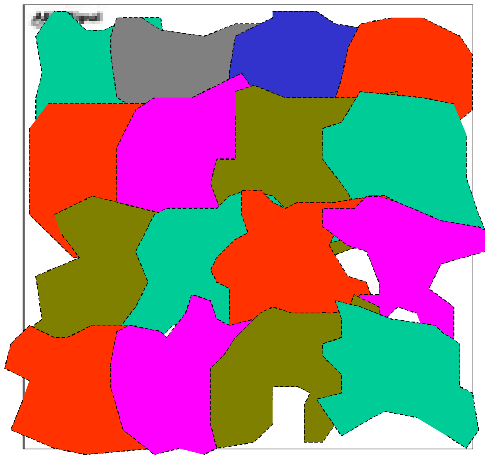
Figure 3. Clusters of sensors in an area of surveillance. Sensors are the cross-marks.

All sensors capable of detecting a target of the same type and associated with a unique grid space form a cluster, as shown in Figure 3. Clusters frequently contain sensors that are located *outside* the physical boundaries of the grid. Conversely, while the grid spaces are non-intersecting, the clusters that are formed do intersect as sensors can belong to more than one cluster. Sensors belong to these fuzzy clusters with a degree of membership between 0 and 1. The membership value is computed as the ratio of the area of intersection of the sensor range circle with the area of the grid space.

5. OPTIMIZATION MODULE

The Optimization module optimizes the sensor network architecture to the Network Objective. It chooses sensors that will be active at a given moment and tells the other algorithms (i.e., tracking, Id fusion) which sensors will report to them. The goal of the module is to choose sensors in such a way that they achieve the highest accuracy of detection, location and identification of specified targets, while minimizing the energy used by the sensor network. Genetic Algorithms are used to achieve this objective.

Each individual of the genetic algorithm population is comprised of several genes. Each of the genes contains one sensor's identification. All the sensors, which are chosen by GA to be active at a given moment, have their identifications coded in the genes. There is a unique identification (integer number starting at zero) associated with each sensor and the genes use a binary encoding for identifications.

Clustering helps the genetic algorithm by restricting the number of sensors that can be chosen as a given gene value, to sensors from the corresponding cluster only. Without clustering we would have to allow for each sensor to be chosen as any gene value, since we would have no prior knowledge that certain sensors cannot sense the target (because the target is outside their detection range, or the sensor is the wrong type). Therefore, clustering reduces the GA search space, allowing GA to run much more effectively.

The GA's internal structure (i.e. number of genes) depends on the network objective. Whenever this objective changes, the number of genes of the GA also changes. Network objective is comprised of suspected targets and required operations associated with them. There is at least one gene that is associated with each suspected target. If the operation is to detect the target, there is one gene associated with this target (see Figure 4a). When the operation is to locate the target, there are as many genes as necessary for location. For example, in case of acoustic bearing sensors this number is three (Figure 4b).

Since in our system a given gene can correspond to any sensor from a given cluster only, the minimum and maximum values that the gene may take depend on the number of sensors in that cluster. However, sensors in one cluster do not have consecutive identifications and this poses a problem for GA. If we allow each gene to take all possible values between minimum and maximum of sensors' identifications from a given cluster, GA will produce many invalid solutions. Only those solutions that have gene values corresponding to sensors (belonging to a given cluster) that can detect a given target are valid. This means that our problem is a constrained optimization problem.

6. GENETIC ALGORITHMS FOR CONSTRAINED OPTIMIZATION PROBLEMS

Optimization problems with constraints are much more difficult to solve than problems without constraints. When optimizing an unconstrained function, any combination of bits in chromosomes is legal. This means that the mutation and crossover operations will always produce a legal offspring. When the problem has some constraints, the crossover and mutation operations on legal chromosomes may produce illegal offsprings. The three most used solutions for solving the constraint problem in genetic algorithms are penalty functions, decoders and chromosome repair [MIC-96].

The problem that we are solving, as mentioned previously, is a constrained optimization problem. Since gene values correspond to sensors' identifications and sensors able to detect a given target have nonconsecutive identification numbers, it is probable that most of GA solutions will be invalid. (This is why the use of penalty functions will not be effective.) We will avoid building illegal individuals by defining a decoder that does not involve specialized crossover and mutation operators. Our decoder involves sensor remapping. All sensors in the cluster associated with a given target that are able to see that target are possible candidates as gene values. Since they have nonconsecutive identifications, we are performing a remapping from their identifications into consecutive integers. We will refer to those consecutive integers as *mapped Ids*. GA operates on mapped Ids. The use of remapping allows changing the constrained optimization problem into an unconstrained one that is much easier to tackle by a GA. Unfortunately it is not possible to perform remapping only once. It has to be redone at each time step, since at each time step the positions of the suspected targets change, and these positions are used in order to determine the cluster and the subset of the sensors from that cluster that are able to detect a given target.

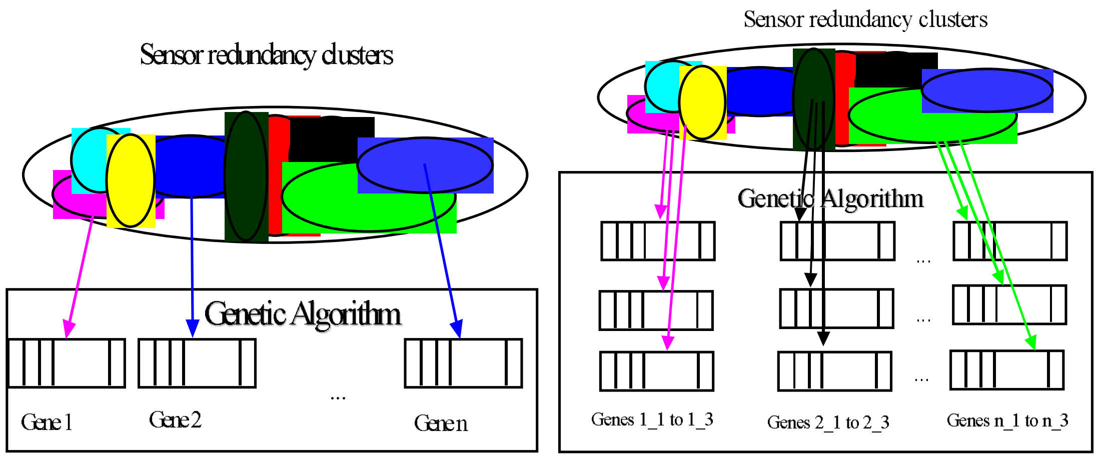

Figure 4. Internal structure of GA for detection (a: One gene per target), and, internal structure of GA for location (b: As many genes per target as the number of sensors necessary for computing location)

GA internal structure (i.e. number of genes) depends on the network objective. Whenever this objective changes the number of genes of GA also changes. Network objective is comprised of suspected targets and required operations associated with them. With each suspected target there is at least one gene associated. If the operation is to detect the target, there is one gene associated with this target (see Figure 4a). When the operation is to locate the target, there are as many genes as necessary for location. For example in case of acoustic bearing sensors this number is three (Figure 4b).

7. GENETIC ALGORITHM CHARACTERISTICS

The implementation of GA is carried through the GeneHunter™ Dynamic Link Library [War-95] of basic GA functions. This library is linked to our C/C++ program. We are using an elitist GA with a population of 50 individuals. The probability of crossover is constant and is set at 0.9 and the probability of mutation is 0.01 (per gene). The GA is run until there is no improvement in the best individual of the population for 30-280 generations. The number of generations without improvement depends on the number of sensors. For a small number of sensors (e.g. 100), the results are satisfactory after 30 generations with no improvement in the fitness function; for large number of sensors (e.g. 2000) there is a need for as many as 280 generations without change. At this point the number of generations without improvement, after which the algorithm stops, was set manually. For the whole system being able to run automatically, a general method for finding the number of generations, has to be developed.

The fitness function describes the power used by the chosen sensors at each moment in time. It has the following form:

$$Fitness = -\sum_{i=1}^{n} power_i \qquad (2),$$

where $power_i$ is the power used by sensor number i. We are using the negative sign in front of the sum because GAs can only perform function maximization.

8. DATA SETS

The fifteen data sets that we performed the tests on ranged from 100 to 2000 sensors. In each data set the sensors were randomly distributed in an area 3-km by 3-km. Sensor power consumption was a random number with uniform distribution from the [1,10] interval. The sensors belonged to three categories, each capable of detecting a particular type of target. About one third of all sensors belonged to each category. The goal was to detect 24 targets belonging to two distinct types. Most of the targets were located in locations on the perimeter of the area under surveillance. Each of the data sets is different, even if it possesses the same number of sensors. The difference lies in the location of sensors and in their individual power consumption.

9. RESULTS

Clustering divides sensors into 259 clusters belonging to 3 categories: 225 clusters for sensors of type 1, 50- for sensors of type 2, and 9 - for sensors of type 3. There are 24 targets: 16 targets that can be detected by sensors of type 2, and 8 targets that can be detected by sensors of type 3. Each target belongs to a different grid space. Since the system's goal is to detect targets, the Genetic Algorithm has one gene per target, resulting in 24 genes per individual.

Table 1 shows the solutions found for our fifteen data sets. Column 1 represents the number of sensors, column 2 - the number of generations needed for convergence, column 3 – the maximum fitness, column 4 the number of different sensors chosen. By looking at column 4 we notice that always less than 24 sensors were chosen: it means that the same sensor is chosen twice or even three times. Those "repeating" sensors can detect targets in adjacent clusters. Power used by the sensor network is minimized when one sensor is chosen for detection of a few targets. Of course this happens only if this sensor uses less power than two sensors from adjacent clusters.

Number of Sensors	Number of Generations	Maximum Fitness	Exclusive Sensors
100	75	-35.417564	13
	85	-46.159046	14
	81	-32.317928	13
200	115	-42.228832	17
	126	-37.034901	15
	325	-31.785740	16
500	195	-43.226357	20
	251	-36.331478	18
	269	-38.128483	21
1000	1207	-33.866329	19
	1128	-40.915699	18
	687	-42.842041	21
2000	790	-45.799641	21
	1407	-40.973480	21
	1068	-43.638779	23

Table 1. GA results for 15 data sets.

Intuitively, the more sensors there are, the higher should be the fitness function of the best individual. This results from the fact that the more sensors there are, the higher should be the number of sensors with low power consumption (since power consumption is always a uniform random number from the same interval). Our results do not indicate this trend. The reason is the exponential growing of search space with the number of sensors, as evident by comparing numbers in Figure 1. The number of generations that the genetic algorithm was run increases (see column 2 of Table 1) at most linearly with the increase of the search space (the linear coefficient does not exceed 1). Since the fitness functions on the average does not become lower for larger number of sensors, it indicates that genetic algorithms should be run longer than they were, in order to obtain quasi-optimal solutions. We still expect the number of generations to increase linearly with the number of sensors. However the coefficient will not be 1 or less as it is now, but will be higher (e.g. 2-3). Another method to improve convergence would be to add a local search at the end of GA run.

10. CONCLUSIONS

This paper describes a system performing self-organization of a sensor network. The goal of the system is to choose sensors necessary to perform the operation described in the network objective while minimizing the power consumption of the entire network. In this paper, special emphasis is placed on the clustering mechanism and the optimization performed by genetic algorithms.

The clustering method developed is simple yet powerful. It suits the problem at hand in which a conventional clustering algorithm is too computationally expensive and not as useful as the one developed. A conventional clustering algorithm groups patterns (in our case sensors) into classes according to the internal similarity between the patterns. While typical clustering algorithms take several passes through the data set prior to completion, our method requires a single pass through the data set. The technique developed in this paper is really not concerned with the similarity of patterns in the cluster. Instead, it is concerned with the fact that a sensor belonging to a specified cluster will be able to detect a target located at a specified position.

The exponential grow of the search space made the problem intractable for most optimization techniques in a reasonable time frame. The fact that the optimal solution is not necessary but a quasi-optimal solution is acceptable allowed the use of genetic algorithms. The development of GA internal structure well suited to the problem being solved and especially the smart decoder devised allowed a fast convergence of the genetic algorithm. The technique developed allows obtaining a quasi-optimal solution in an exponential search space in a linear time frame. This is a substantial achievement making it possible for the system to be used in real time.

11. ACKNOWLEDGEMENTS

This work was supported by AlliedSignal during the execution of a DARPA contract for Technology for Tactical Sensors and Tasking and Control in Small Unit Operations (contract number F04701-97-C-0018), Dr. Ed Carapezza program manager. Support of this program is gratefully acknowledged. We would like also to thank Dr. Richard Burne, the AlliedSignal Program Manager, as well as the whole AlliedSignal–Northrop Grumman team for fruitful discussions.

12. REFERENCES

[Cha-97] V. Chandru, M. Rao, "Combinatorial Optimization", *The Computer Science and Engineering Handbook*, ed. A. Tucker, Jr., CRC Press, 1997 pp. 316-354.

[Dav-91] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, NY 1991.

[Gol-89] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989.

[Mic-96] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1996.

[War-95] Ward Systems Group, *GeneHunter User's Manual*, March 1995.