2004

Dynamic Sensor Management Using Multi Objective Particle Swarm Optimizer

Kalyan Veeramachaneni
Syracuse University, Department of Electrical Engineering and Computer Science, kveerama@syr.edu

Lisa Ann Osadciw
Syracuse University, Department of Electrical Engineering and Computer Science, laosadci@syr.edu

Dynamic Sensor Management Using Multi Objective Particle Swarm Optimizer

Kalyan Veeramachaneni, Lisa Ann Osadciw

Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240
(315) 443-1319 (office)/(315) 443-2583 (fax)
kveerama,laosadci@syr.edu

ABSTRACT

This paper presents a Swarm Intelligence based approach for sensor management of a multi sensor networks. Alternate sensor configurations and fusion strategies are evaluated by swarm agents, and an optimum configuration and fusion strategy evolves. An evolutionary algorithm, particle swarm optimization, is modified to optimize two objectives: accuracy and time. The output of the algorithm is the choice of sensors, individual sensor's thresholds and the optimal decision fusion rule. The results achieved show the capability of the algorithm in selecting optimal configuration for a given requirement consisting of multiple objectives.

Keywords: Bayesian decision fusion, particle swarm optimization, multi objective optimization, sensor management

1. INTRODUCTION

Multi sensor systems consist of several sensing options and configurations, each having some specific advantages and drawbacks. When presented with varied sensor resources and many sensor parameters, adaptive configuration of the system presents a complex multi objective optimization problem. Implicit is the requirement for an a priori understanding of the goals of the system. These goals provide the criteria used to evaluate alternative strategies and configurations. Most of the multi sensor systems designed are managed by an ad-hoc sensor management strategy or manually. The system goals, sensors, and surrounding situation, however, are dynamic and hence implies a need to change the configuration of the sensor network. This can be accomplished by designing a multi objective optimization function and a corresponding strategy to solve it. In [4] a strategy is presented that adaptively sets sensor thresholds and associated fusion rules for specified accuracy requirements of a biometric sensor network. This paper expands the objective with multiple requirements such as time.

When the system has several objectives such as time, the communication bandwidth, coverage area of each sensor, sensor decision thresholds, and many other parameters, designing the sensor network configuration becomes a very complex problem not easily solved within the required time frames. The aim of the designer is to maximize the accuracy, reduce communication resource needs, and maximize the sensor suite coverage. There are infinitely many solutions to this problem with each solution resulting from trade-offs between performance parameters.

In this paper an evolutionary algorithm approach is presented that searches the configuration space and finds an optimal configuration. An additional objective of time has been added to increase complexity. The particle swarm algorithm is modified to solve this multi objective problem for a few different priorities of the objectives. Bayesian decision fusion framework as in [11] is used to fuse the decisions from multiple sensors.

The next section presents the multi objective problem this paper solves. Bayesian decision fusion and time constraints are discussed. Section 3 presents the particle swarm optimization algorithm used to solve the multi objective problem.

The problem formulation and the design of the cost function are discussed in this section. Section 4 presents the results achieved by the PSO. Finally, conclusions are presented in Section 5.

2. MULTI OBJECTIVE OPTIMIZATION PROBLEM

In this section the two objectives, accuracy and time, are discussed in detail. This section describes how the objectives are quantified, which is used later in designing the cost function for the optimization algorithm. It should be noted that the design of the cost function is crucial for any optimization algorithm.

2.1 Bayesian Decision Fusion Framework

This paper considers a biometric sensor network. The biometric sensor network is characterized by a set of sensors, which match the data given by the user at the time of identification to the data stored in template during the enrollment process. The template can either be stored in the central database or a smart card owned by the user.

As a brief review, the problem of personal identification can be formulated as a hypothesis testing problem where the two hypotheses are

H_0 : the person is an imposter or

H_1 : the person is genuine.

The conditional probability density functions are $p(u_i|H_1)$ and $p(u_i|H_0)$ where u_i is the output of the i^{th} biometric sensor given the genuine person and the imposter, respectively. The decision made by sensor *i* is

$$u_i = \begin{cases} 0, \text{ person is an imposter} \\ 1, \text{ person is genuine} \end{cases} \quad (1)$$

This decision is made based on the following likelihood ratio test

$$\frac{p(u_i|H_1)}{p(u_i|H_o)} \underset{u_i=0}{\overset{u_i=1}{\gtrless}} \lambda_i \quad (2)$$

where λ_i is an appropriate threshold [14, 15, 16].

The four possible decisions are:

1. The genuine person is accepted
2. The genuine person is rejected
3. The imposter is accepted
4. The imposter is rejected

Accuracy, which is one of the performance parameters, refers to the rates at which the two types of errors occur: false rejection rate (F_{RR}) and false acceptance rate (F_{AR}). We define the error rates as

$$F_{AR_i} = P(u_i = 1|H_0) \text{ and} \quad (3)$$

$$F_{RR_i} = P(u_i = 0 | H_1) . \quad (4)$$

The performance of a detector is often represented in terms of receiver operating characteristics (ROC) or a plot of the genuine acceptance rate versus F_{AR} for a specific decision rule. It should be pointed out that the optimum decision rule is defined as the rule that minimizes the probability of error. The operating point selection uses the biometric sensor's ROC, which is usually available from sensor manufacturers. From this, the fusion of the biometric sensor decisions is analyzed, and the optimum decision fusion rule is selected.

The optimum Bayesian fusion rule allowing access to a building for *N* sensors is [15][16]

$$\sum_{i=1}^{N} \left[u_i \log\left(\frac{1 - F_{RR_i}}{F_{AR_i}} \right) + (1 - u_i) \log\left(\frac{F_{RR_i}}{1 - F_{AR_i}} \right) \right] \begin{matrix} u_g = 1 \\ > \\ < \\ u_g = 0 \end{matrix} \log\left(\frac{C_{FA}}{2 - C_{FA}} \right) \quad (5)$$

where u_i is the local sensor decision from (1), u_g is the global decision, and *N* is the number of sensors. The rule in (5) assumes an equal a priori probability of an imposter and genuine user. There is a total of 2^{2^N} possible fusion rules if all possible combinations of the sensor decisions are considered.

An assumption of Gaussian sensor noise is made so the mean and variance of the noise is all that is required by the sensor models. If new threats or sensor degradations affect the system, the algorithm can react by modifying the optimum rule in response to these changes.

Typically, the F_{AR} and F_{RR} cannot be reduced simultaneously. As the number of sensors increases and the operating points are varied, however, this restriction vanishes by rule selection over different cost regions. Error costs affect the rule in (5) and, consequently, the total cost. Since security is usually the prime objective, a low F_{AR} is usually desired. The user assigns a higher cost to the F_{AR} error in the Bayesian framework to express this security need.

Since the algorithm is developed using a Bayesian framework, a total error cost is defined as a weighted sum of the two global errors, GFAR and GFRR, or

$$E = C_{FA} GFAR + C_{FR} GFRR \quad (6)$$

where C_{FA} is the cost of falsely accepting an imposter individual, C_{FR} is the cost of falsely rejecting the genuine individual, *GFAR* is the global F_{AR}, and *GFRR* is the global F_{RR}. This can be written in terms of a single cost using

$$C_{FR} = 2 - C_{FA} \quad (7)$$

giving

$$E = C_{FA} GFAR + (2 - C_{FA}) GFRR . \quad (8)$$

The optimum Bayesian fusion rule that minimizes the total cost (8) is obtained by selecting the rule to combine single biometric sensor decisions into one decision. The single sensor observations and the corresponding decisions are assumed to be independent. This is the cost function of the PSO which is minimized by selecting appropriate sensor operating points and fusion rule.

The *GFAR* and *GFRR* for the fusion rule can be calculated directly from the fusion rule, (5), sensors' F_{AR}, and sensors' F_{RR}. Let *f* be a binary string that represents the fusion rule of length $\log_2 p$. For two sensors the fusion rule consists of 4 bits as represented in the Table 1.

Table 1.Fusion Rule Formation for 2 Sensors

u_1	u_2	f
0	0	f_0
0	1	f_1
1	0	f_2
1	1	f_3

In Table 1, u_1 is the first sensor decision; u_2 is the second sensor decision. These local decisions are related to the global decision represented by 0s and 1s in place of the f variables. This representation of the fusion rule can be used to compute the global error rates using

$$GFAR = \sum_{i=0}^{p-1} f_i \times \left(\prod_{j=1}^{N} E_j \right) \tag{9}$$

where $E_j = 1 - FAR_j$ if $u_j = 0$; $f_i \in [0,1]$, and $E_j = FAR_j$ if $u_j = 1$.

Similarly,

$$GFRR = \sum_{i=0}^{p-1} (1 - f_i) \times \left(\prod_{j=1}^{N} E_j \right) \tag{10}$$

where $E_j = 1 - FRR_j$ if $u_j = 1$; and $E_j = FRR_j$ if $u_j = 0$.

The distributions assumed in this paper to model the detection process for the biometric sensors is gaussian and given in the Table 2. The distributions are drawn at random and do not pertain to any real biometric sensors.

Table 2.Table Showing Means and Standard deviations of Gaussian Distributions of Sensors

Parameter	Sensor 1	Sensor2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
Mean μ_1	47.37	67.75	50.41	92.40	81.90	47.70
Standard Deviation σ_1	43.86	52.63	26.20	39.39	22.08	47.16
Mean μ_2	144.51	251.20	167.46	268.73	263.49	182.68
Standard Deviation σ_2	12.84	23.00	10.19	40.45	54.09	28.36

2.2 Time Calculations

In this paper sensors are characterized by transaction times, which is time they need to do the processing and/or need any resource (e.g., communication resource) to send the data they processed. The system transaction time is based on the sensors, which are used in the fusion process, and the individual transaction times of those sensors. In this section a method for the modelling of transaction times given the subset of sensors is given. Let the subset of sensors used of the available sensors can be represented as

$$U = \{u_1, u_2, u_3 \ldots\ldots u_N\} \tag{11}$$

where $u_i = 1$, if the *ith* sensor is used, and $u_i = 0$, if the *ith* sensor is not used.

The transaction times for N sensors are given by

$$T = \{t_1, t_2, t_3 \ldots\ldots t_N\} \tag{12}$$

The transaction time for the system can be calculated by considering three cases

Case 1: All the sensors are parallel, which implies that all the sensors can simultaneously take measurements. In such a scenario, the transaction time of the system is the maximum transaction time of the sensors used or

$$T_s = max\{u_1 t_1, u_2 t_2, u_3 t_3 \ldots\ldots\ldots\ldots u_N t_N\} \tag{13}$$

Case 2: All the sensors are serial in operation, which means that at any time only one sensor can work. In such a scenario, sensors work one after another and the transaction time is

$$T_s = \sum_{i=1}^{N} u_i t_i \tag{14}$$

Case 3: Sensors are a mix of parallel subsets in serial (i.e. a combination of case 1 and 2). The problem becomes finding the optimal schedule making use of the parallelism. This is, however, not the focus of this paper. However, let the groups of sensors, which can operate in parallel, be given by

$$G = \{g_1, g_2 \ldots\ldots\ldots g_n\} \tag{15}$$

where *G* is the set of all such groups and g_i is a subset of sensors,

$$g = \{s_1, s_2 \ldots\ldots s_i\} \tag{16}$$

where $1 \leq s_i \leq N$, where *N* is the number of sensors. Each sensor can only belong to one group subset. Transaction time for each group is given by

$$T_g = max(t_{s_1}, t_{s_2}, t_{s_3} \ldots\ldots\ldots t_{s_i}) \tag{17}$$

Transaction time for the system is given by

$$T_s = \sum_{i=1}^{n} T_{g_i} \tag{18}$$

In this paper, Case 2 is assumed, which means that the total transaction time is the sum of the sensors' individual transaction times.

3. PARTICLE SWARM OPTIMIZATION

The particle swarm optimization algorithm (PSO) is a population based evolutionary computation technique originally developed by Kennedy and Eberhart in 1995 [15]. The algorithm lends itself very well in searching through a highly multi modal search space resulting a optimized solution to a D-dimensional problem. In [4], PSO has been used to dynamically set the thresholds for a biometric sensor suite resulting in higher accuracy. As indicated in the previous section multi sensor systems are also as sensitive to other objectives as they are to higher accuracy. In this paper an additional objective, time, has been added to the objective function.

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behavior, has come to be widely used as a bottom up problem solving method in engineering and computer science. PSO has since proven to be a powerful competitor to genetic algorithms [19]. The technique is fairly simple and comprehensible as it derives its simulation from the social behavior of individuals. The individuals, called particles henceforth, are flown through the multidimensional search space, with each particle representing a point in the search space and a possible solution. The movement of the particles is influenced by two factors: local best solution and global best solution. As a result of the first factor, the best solution of the particle, it experiences a pull towards this position as it moves. As a result of the second factor, the best position visited by any particle in the search space, it experiences an additional pull towards this position. The first and the second factors are called cognitive and social components respectively. After each iteration the local best and global best are updated if a more dominating solution (in terms of fitness) is found. This process is continued iteratively until either the desired result is achieved or the iteration count exceeds the limit.

The PSO formulae define each particle in the D-dimensional space as $X_i = (x_{i1}, x_{i2}, x_{i3} \ldots\ldots x_{iD})$ where the subscript i represents the particle number and the second subscript is the dimension. The memory of the previous best position is represented as $P_i = (p_{i1}, p_{i2}, p_{i3} \ldots\ldots p_{iD})$ and memory of the global best is s $P_g = (p_{i1}, p_{i2}, p_{i3} \ldots\ldots p_{iD})$ [17]. After each iteration, the velocity term is updated with the influence from P_i and P_g apparent in [15, 18, 17]

$$V_{id}^{(t+1)} = \omega \times V_{id}^{(t)} + rand(1) \times \Psi_1 \times (p_{id} - X_{id}^{(t)}) + rand(1) \times \Psi_2 \times (p_{gd} - X_{id}^{(t)}), \text{ and} \tag{19}$$

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)}. \tag{20}$$

Constants Ψ_1 and Ψ_2 determine the relative influence of the social and cognition components and often both of these are set to same value to give equal weight to both. The memory of the swarm is controlled by ω.

3.1 Design of the Algorithm

The evolutionary algorithm in this paper outputs the choice of sensors, thresholds for each sensor and the fusion rule. The aim is to maximize the accuracy and minimize the time. It should be noted that each additional sensor helps in achieving higher accuracy, but increases time. The three different cases for calculating the system time given the individual time for the sensors and the subset of sensors used is given in previous section.

In this paper swarm agents are used to evolve the choice of sensors. In other words each agent is a subset of sensors from the sensor suite which are used for fusion. The whole available sensor suite is made available to the agents to make the selection. The agent has a dimension which is equal to the number of sensors in sensor suite. The agent is a binary vector with a '1' at a particular index representing use of that sensor (each index uniquely identifies a particular sensor). Hence the representation is given in (21)

$$S = \{s_1, s_2, s_3 \ldots\ldots\ldots\ldots s_n\} \tag{21}$$

where $s \in [0, 1]$ and 'n' is the number of sensors in the sensor suite.

Each agent then calls particle swarm optimization algorithm evolve the thresholds and optimum fusion rule for this sensor set. The choice of sensors and the corresponding distributions are fed into the PSO algorithm which searches the optimum fusion rule and the individual sensor thresholds. The design of this particle, cost function are described below.

The particle of this problem has *'N+1'* dimensions, where *N* is the number of sensors in the sensor suite. Each of the *N* dimensions is a threshold at which that particular sensor is set. The *'N+1' th* dimension is the fusion rule, which determines how all the decisions from the sensors are fused**.** Hence the representation of the solution is

$$X_i = \{\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, \ldots . \lambda_{in}, f_{in+1}\} \tag{22}$$

The values which need to be evolved by the evolutionary computation technique are thresholds and the fusion rule. It can be argued that the fusion rule is also a function of thresholds and the error costs as in (5), however, evolving fusion rule along with thresholds is much cheaper and effective than otherwise.

The sensor thresholds are continuous. The fusion rule, however**,** is a binary number having a length of $\log_2 p$ bits, where $p = 2^{2^N}$, with a real value varying from $0 \le f \le p-1$. For binary search spaces, the binary decision model as described in [18] is being used. An alternative is to simply evolve a real number representation of the rule. This leads to an additional procedure bounding the resulting real values to lie within the search space. The bounding process results in the particles selecting the rule at the boundary too often. A binary decision model works better for moving through the decision fusion space. In the algorithm instead of evolving the thresholds explicitly, the false acceptance rates *(F_{AR})* are evolved for each of the sensors. Thresholds are calculated from this and then the F_{RR}s are calculated depending on the mean and standard deviation of the sensor noise determined a priori.

The accuracy error for the system which has to be minimized is

$$E_a = C_{FA} \times (GFAR_a - GFAR_d) + (2 - C_{FA}) \times (GFRR_a - GFRR_d) \tag{23}$$

where E_a is the total error cost. $GFAR_a$ and $GFRR_a$ are the global false acceptance and rejection rates. $GFAR_d$ and $GFRR_d$ are the desired global false acceptance and rejection rates.

A flowchart of the algorithm implemented is presented in Figure 1. The PSO outputs the optimal fusion rule, individual sensor settings minimizing the total error cost as in (23). These are fed back to the respective agents. Each agent is characterized by the transaction time which is the sum of all transaction times of the sensors which the agent uses. Each agent has now a configuration and corresponding values for both the objectives: time and accuracy. To evolve the agents a binary PSO is used which is very similar to the continuous PSO and is presented by Kennedy and Eberhart in [18]. For doing the comparisons between the agent's current location and best vectors (previous best locations stored in its memory), the cost of each agent is calculated in the following manner. The agents' present and the previous best values of the total error cost are merged into a single vector. Similarly, the cost values for the time are also merged. Each new vector corresponding to time and accuracy are of dimension 2P, where P is the number of particles. The new vectors are now normalized and the cost for all the agents is calculated from

$$C_i = W_a \times E^n{}_{a_i} + W_t \times T^n{}_{a_i} \tag{24}$$

where $E^n{}_{a_i}$ is the normalized total error cost value for accuracy and W_a is the weight given to the accuracy objective. Similarly $T^n{}_{a_i}$ is the transaction time for the *ith* agent and W_t is the weight given to this objective. The comparisons between the cost values of *present* solutions and the *pbest* solutions found by the agents are done and the *pbest* solutions are updated with the better ones. This merge, normalize and compare strategy is important so as to maintain consistency in updates of *pbest* in successive iterations, directing the agents towards better solutions.

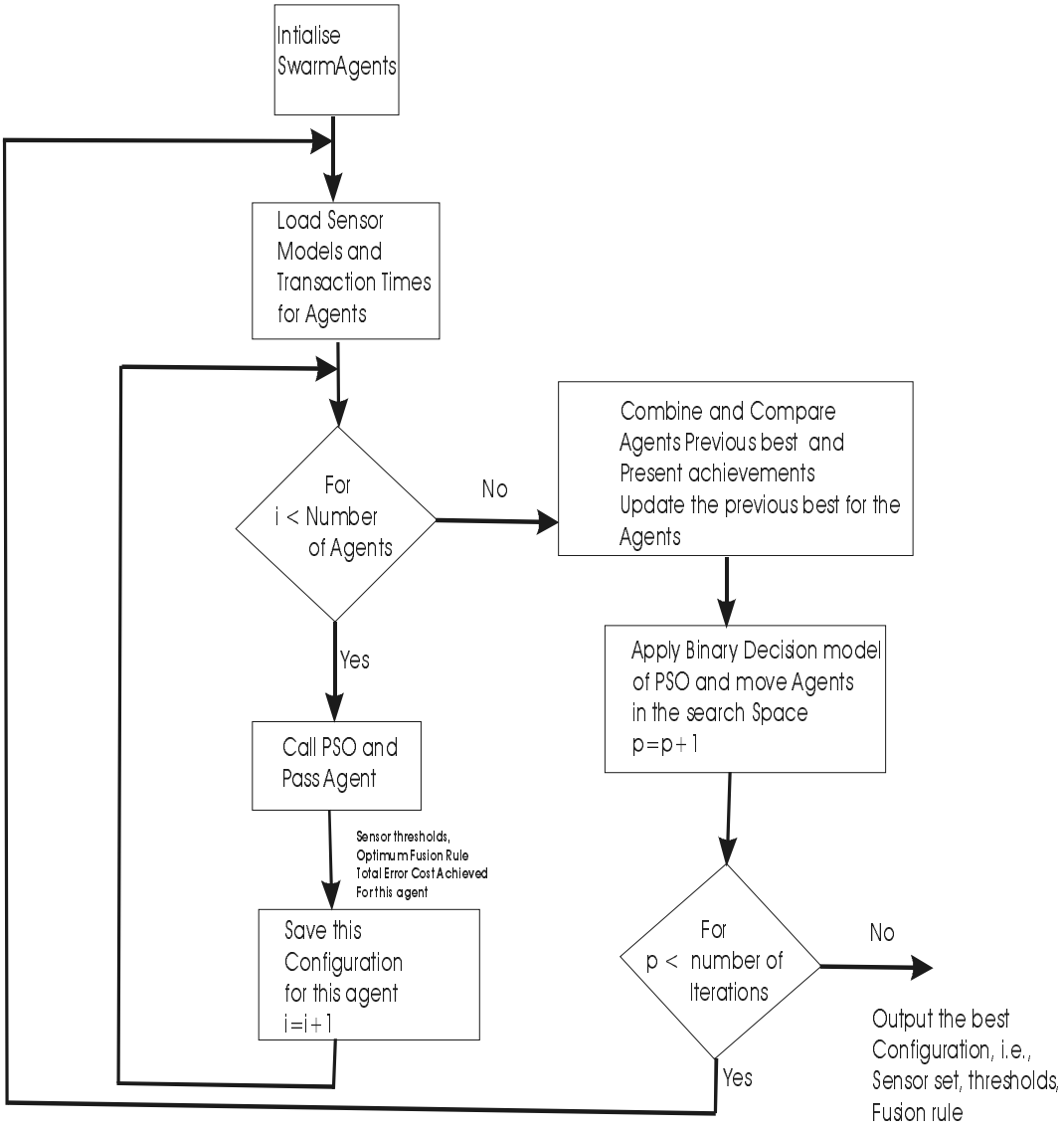

Fig. 1.Flowchart of the Algorithm

4. Results and Discussion

The sensor suite with sensors having distributions as given in Table 2 are made available to the agents. The transaction times for the sensors are given in the Table 6. The sensors present in sensor suite can be sorted based on the accuracy and transaction time values. It can be said that the least the area of the overlap between the two pdfs for a sensor the more accurate the sensor is. Hence based on distributions it can be said that the most accurate sensor is Sensor 3. Sensor 5 and Sensor 2 share the second place and are very close to each other followed by Sensor 4. Sensor 1 is the next highest in accuracy. Sensor 6 is the least accurate and hence worst of all in the sensor suite. Similarly, according to the transaction times the order is Sensor 1, 5, 4, 2, 6,3.

A population of 10 particles for evolving thresholds and optimal fusion rule were used. A population of 10 agents were used for evolving choice of sensors. The PSO evolving the sensors thresholds and fusion rule ran for 1000 iterations

and the binary PSO for agents ran for 20 iterations. The results presented here are the agents and the respective thresholds and fusion rule at the end of the 20 iterations. The total cost value is calculated by normalizing the cost value for accuracy and also cost value for time and then substituting these values in (24). Table 3 presents the results obtained when a weight of 0.8 was given for accuracy and 0.2 was given to the time objective. All the 10 agents consistently selected Sensor 3 because of its highest accuracy among the sensor suite. Also optimal fusion rule for all the agents which have Sensor 3 have Sensor 3 'anded' to the rest of the rule. This is making use of the highest accuracy of Sensor 3. Sensor 6 has not been selected by even a single agent because of its least accuracy as well as very high time. The agent of best performance and of interest is agent 5 with least error value 0.4353. It should be noted that the system will be set at the configuration pertaining to the best agent.

Another interesting observation is that 8 out of 10 agents selected Sensor 2 and Sensor 3 converging to a solution. Agent 5, however, selected Sensors 2,3,5 achieving higher accuracy, since 2,3,5 are all highly accurate sensors.The cost value achieved by this is very close to the cost value achieved by the agents selecting Sensor 2 and Sensor 3.

Table 3.Results for C_{FA}=1.9, C_{FR}=0.1, W_a=0.8, W_t=0.2

Agent #	Sensors	Fusion Rule	Cost Value Accuracy	Cost Value Time	Total Cost Value
1,2,3,4, 6,7,8,10	(2,3)	2&3	8.7937e-5	126.7533	0.5967
5	(2,3,5)	3&(2 or 5)	4.7919e-5	153.6522	0.4353
9	(1,3,5)	3&(1or5)	1.6294e-4	133.4537	0.9737

For the second simulation the weights have been swapped. The results presented in Table 4 present the configurations set by the 10 agents at the end of the 20 iterations. 6 agents selected single sensor configuration i.e., Sensor 1, 2, 6, 4, 5 with very close cost values and the agent selecting Sensor 5 being the best. None of the agents, however, selected single Sensor 3. Either Sensor 5 or Sensor 2 or both of them have been selected by many agents. This is due to the fact that the transaction time for Sensor 5 is the second least, least being Sensor 1. Sensor 5 is, however, far more accurate than Sensor 1. Although Sensor 2 has very high transaction time, it is highly accurate. Hence intuitively this selection of the agents make sense. Sensor 3 has been completely neglected in the selection due to its high transaction time. In this simulation Sensor 6 has been selected by agent 8. This selection has proven better than the two sensor configurations (2,4), (2,5) due to the fact that the transaction time of Sensor 6 alone is less than that of configuration (2,4) as well as (2,5). Also, Sensor 1 alone has proven better than any of the configurations having two sensors due to its least transaction time.

Table 4.Results for C_{FA}=1.9, C_{FR}=0.1, W_a=0.2, W_t=0.8

Agent #	Sensors	Fusion Rule	Error Value Accuracy	Error Value Time	Total Error Value
1	(2,4)	2&4	0.0015	82.7375	0.8045
2	4	4	0.0098	38.772	0.4044
3	5	5	0.0031	26.8989	0.2694
4	(2,5)	2&5	0.0012	70.8644	0.6888
5	1	1	0.0664	23.7670	0.4298
6	2	2	0.0102	43.9655	0.4558
7	(4,5)	4 &5	0.0018	65.6709	0.6404
8	6	6	0.0376	55.432	0.6492
9	(2,5)	2 &5	0.0012	70.8644	0.6888
10	2	2	0.0102	43.9655	0.4558

System is set at the configuration of the best agent which is agent 3 in this case. Table 5 summarizes the system configurations in the two optimum cases from the previous examples.

Table 5.System Setting for Two Different Weights for Different Objectives

System Requirements	Sensors	Fusion Rule	Operating Points
C_{FA}=1.9 C_{FR}=0.1 W_a=0.8 W_t=0.2	2,3,5	3 &(2 or 5)	(0.0142,0.0015), (0.0012, 0.0001), (0.0009, 0.0189)
C_{FA}=1.9 C_{FR}=0.1 W_a=0.2 W_t=0.8	5	5	(3.3733 e -4, 0.0245)

Sensors 2,3,5 have been selected for the first weighting. Sensor 3 is the most accurate sensor and correspondingly the fusion rule 'ands' it with the rest of the rule. Sensor 2 and Sensor 5 are the next accurate sensors in the suite and hence have been selected. An 'or' between these two helps in achieving a better F_{RR} overall than otherwise. Sensor 5 is selected as the setting for the second weighting and since Sensor 5 is dominant sensor in terms of accuracy as well as time when all the single sensors are considered individually.

Table 6.Transaction Times For The Sensors

Sensor #	Transaction Times
1	23.7670
2	43.9655
3	82.7878
4	38.7720
5	26.8989
6	55.4320

The results showed that given the weights for different objectives the agents are able to do the trade-offs and are able to decide an optimal configuration of sensors and their thresholds and optimal fusion rule.

5. CONCLUSIONS

The paper presented an evolutionary algorithm based approach to evolve sensor sets to do fusion for a multi sensor network. It should be noted that the sensors considered in this paper can be replaced by any sensor that simply detects events. These sensors are characterized by a probability of detection and probability of false alarm. The biometric sensor models can be replaced with the receiver operating curves determined experimentally for the actual sensors. This modifies the algorithm for another sensor network application. For example, this algorithm can be used for managing a network of radars, which detect the presence of an aircraft, rain cloud, missile, etc.

The results presented emphasize the algorithms' capability to solve the multi objective optimization problem. The problem dimension presented here is simple, so as to easily demonstrate the algorithm's effectiveness. The dimension of sensor suite can be much larger than the size presented here and the algorithm will efficiently evolve the optimum sensor set and their thresholds and the optimum fusion rule.

In future work the algorithm will be tested on other types of sensor suites in which the sensors can operate in parallel or groupwise in parallel, as presented in section 2. Also addition of other objectives such as communication resource will be considered.

REFERENCES

1. Lloyd G. Greenwald, Harish Sethu, "On Scheduling Sensor Networks", Proceedings of AAAI/KDD/UAI Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton, Alberta, Canada, July 29, 2002

2. N. Srinivas and Kalyanmoy Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms", Journal of Evolutionary Computation, Vol. 2, No. 3, pages 221-348.

3. Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri Mohan, Lisa Ann Osadciw, "Optimization Using Particle Swarm with Near Neighbor Interactions", Genetic and Evolutionary Computation Conference, July 12- 17th, 2003, Chicago, Illinois.

4. Kalyan Veeramachaneni, Lisa Ann Osadciw, Pramod K Varshney, "An Adaptive Multimodal Biometric Fusion Algorithm Using Particle Swarm", Multisensor, Multisource Information Fusion : Architectures Algorithms , and Applications, 23-25 April, 2003, Oralndo, Florida.

5. Thanmaya Peram, Kalyan Veeramachaneni, Chilukuri Mohan, "Fitness-Distance-Based Particle Swarm Optimization", IEEE Swarm Intelligence Symposium, April 24-26, 2003, Indianapolis, Indiana.

6. Lin Hong and Anil Jain, "Integrating Faces and Fingerprints for Personal Identification", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, Dec., 1998, pp. 1295 - 1307.

7. A.K. Jain, S. Prabhakar, S. Chen, "Combining multiple matchers for a high security fingerprint verification system", Pattern Recognition Letters 20 (11-13) (1999) 1371-1379.

8. A.K. Jain, R.M. Bolle, S. Pankanti (Eds.), "Biometrics : Personal Identification in a Network Society", Kluwer Academic Publishers, MA, 1999.

9. Robert W. Frischholz, Ulrich Deickmann, " BioID: A Multimodal Biometric Identification System" , IEEE Computer, Vol. 33, No. 2, February 2000.

10. L.Hong, A.K. Jain, S. Panikanti, "Can multibiometrics improve perfomance?",Proceedings AutoID'99, Summit, NJ, October 1999, pp. 59-64.

11. L.Osadciw, P.K.Varshney, and K. Veeramachaneni, " Improving Personal Identification Accuracy Using Multisensor Fusion for Building Access Control Applications", Proceedings of the Fifth International Conference on Information Fusion, July 2002, Annapolis, Maryland.

12. Steven M. Kay, Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II, Prentice-Hall, Inc., 1998.

13. Ramanarayanan Viswanathan and Pramod K. Varshney, "Distributed Detection With Multiple Sensors: Part I - Fundamentals", Proceedings of the IEEE, Vol. 85, No. 1, Jan., 1997, pp. 54 - 63.

14. Pramod K. Varshney , Distributed Detection and Data Fusion, Springer, New York,1997.

15. Kennedy J. and Eberhart, R., "Particle Swarm Optimization", IEEE International Conference on Neural Networks, 1995, Perth, Australia.

16. Shi Y. H., Eberhart R. C., "Parameter Selection in Particle Swarm Optimization", The 7th Annual Conference on Evolutionary Programming, San Diego, USA.

17. Carlisle A. and Dozier G.. "Adapting Particle Swarm Optimization to Dynamic Environments", Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, pp. 429-434, 2000.

18. Kennedy J., Eberhart R. C., and Shi, Y. H., Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

19. X. Hu, Eberhart R.C., “Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems”, Proceedings of Congress on Evolutionary Computation, 2002. pp. 1666-1670. Hawaii, USA.