Journal of Network and Computer Applications 34 (2011) 1818–1826

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks

Wenjia Liu [a], Bo Chen [b,*]

[a] Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931, USA
[b] Department of Mechanical Engineering – Engineering Mechanics, Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931, USA

ARTICLE INFO

Article history:
Received 13 June 2010
Received in revised form
5 November 2010
Accepted 11 December 2010
Available online 22 December 2010

Keywords:
Multi-objective genetic algorithm
Mobile monitoring agent
Artificial immune system

ABSTRACT

This paper studies optimal control of mobile monitoring agents in artificial-immune-system-based (AIS-based) monitoring networks. In AIS-based structural health monitoring (SHM) networks, the active structural health monitoring is performed by a group of mobile monitoring agents equipped with damage pattern recognition algorithms. The mobile monitoring agents mimic immune cells in the natural immune system and patrol a structure to detect damage patterns using their receptors (feature vectors), damage pattern recognition algorithms, and the dynamic response data of the structure. The optimal control of mobile monitoring agents includes agent generation and distribution. The generation of mobile monitoring agents is optimized to minimize the response time for the mobile monitoring agents to diagnose structural damage in a sub-network and maximize the average affinity of monitoring agents' receptors to the damaged sensor data feature vector. The objective functions for distributing mobile monitoring agents are to increase the detection probability and extend network life by balancing energy consumption of sensor nodes in the network. The presented optimization algorithms are developed using multi-objective genetic algorithms. The impact of the algorithm parameters on the performance of the algorithm is also investigated.

1. Introduction

Monitoring networks have gained wide applications in protecting engineered systems and natural environment from unexpected failures. Due to ever-increasing complexity of systems and unpredictable working conditions, the monitoring systems will need to perform tasks with high quality, adaptability, and autonomy. To meet this requirement, a monitoring paradigm based on artificial immune system (AIS) concept has been proposed (Chen, 2010). In this paradigm, mobile monitoring agents mimic immune cells (such as B cells) in the natural immune system for the anomaly detection and pattern recognition in distributed monitoring systems. Mobile monitoring agents interact locally with monitoring environment, and respond to emerging problems through simulated immune responses. The monitoring tasks are managed automatically by a mobile agent-based network middleware. This bio-inspired monitoring paradigm has been applied to structural health monitoring networks.

The main features of the immune system include adaptive immune response to the invading pathogens and pattern recognition capabilities. When a pathogen invades a host, the host mounts a response that occurs at several levels of biological organization, including genetic, molecular, cellular, tissue, and system level. A number of host cells are called into action, such as B cells, T cells, and antigen presenting cells (Neal and Trapnell, 2007). The adaptive immune response achieves two goals: the number of B cells that are capable of responding to a particular antigen are multiplied through clonal expansion and these new generated immune cells are able to produce a large number of antibodies for binding to the intruders (Delves et al., 2006).

In AIS-based monitoring networks, a mobile monitoring agent is selected for cloning when it detects damage in a sensor unit. Multiple copies of this type of monitoring agents will be created. As a result, the type and amount of mobile monitoring agents are adapted to damage patterns detected in a structure. This paper presents multi-objective optimization algorithms for the optimal control of mobile monitoring agents in artificial-immune-system-based monitoring networks. To minimize the response time for the generated monitoring agents to diagnose structural damage and maximize the average affinity of monitoring agents' receptors to the damaged sensor data feature vector, a multi-objective genetic algorithm is developed to find appropriate number of agent clones and the mutation value for the cloned monitoring agents. The newly generated monitoring agents are sent to sensor nodes close to the location where the damage is detected for further damage diagnosis. The distribution of mobile monitoring agents takes sensor data feature vector and the remaining

* Corresponding author. Tel.: +1 906 487 3537.
E-mail address: bochen@mtu.edu (B. Chen).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.12.004

battery capacity of sensor nodes into consideration to increase the detection probability and extend the lifetime of the monitoring network.

Genetic algorithms (GA) have been widely applied to many optimization problems. In a sensor network for agriculture application (Ferentinos and Tsiligiridis, 2007), genetic algorithm is adopted to design wireless sensor network topologies by optimizing parameters which are related to the power consumption of the sensors. Parameters of network connectivity and application requirements are also considered so that an integrated network is designed. Khanna et al. (2006) design a reduced-complexity genetic algorithm to minimize the power consumption of the sensor system while maximizing the sensing coverage. The genetic algorithm runs periodically to assign functions to the randomly deployed sensor nodes. In Hussain et al. (2007), a genetic algorithm is applied for data dissemination by creating energy-efficient clusters. The authors propose an intelligent hierarchical clustering protocol, which has better performance than the traditional cluster-based protocols. Maslov and Gertner (2006) introduce basic and advanced genetic algorithm implementations and applications in information fusion. Genetic algorithm is also employed in the work of node deployment for wireless sensor networks (Bhondekar et al., 2009) and the GA system decides which sensor nodes should be active and which sensor node should work as the cluster head in the network. To maximize network lifetime, a regression model (Qinru et al., 2006) is applied to find out an energy-efficient configuration for the genetic algorithm to optimize resource allocation.

The rest of the paper is organized as follows. Section 2 introduces an artificial-immune-system-based structural health monitoring network. Section 3 discusses the pattern recognition-based damage detection and classification. Section 4 presents multi-objective genetic algorithms for the optimal control of agent generation and distribution, as well as the impact of algorithm parameters on the system performance. Section 5 concludes the presented work.

2. An AIS-based structural health monitoring network

2.1. AIS-based structural health monitoring

The framework of an AIS-based structural health monitoring network consists of (a) an agent-based network middleware (mobile agent system) (Chen et al., 2006; Chen and Liu, 2010) to support the generation, execution, migration of mobile monitoring agents; (b) clonal selection algorithm for the cloning of mobile monitoring agents who detect structural damage in distributed sensor units; (c) knowledge base for keeping updated representative feature vectors (memory cells) for normal and damage patterns and performing confirmation of damage detected by a mobile monitoring agent; (d) agent interaction protocols, for example, agent communication protocols amongst mobile monitoring agents, knowledge-base agent, and the clonal selection agent for cloning a mobile monitoring agent. Figure 1 shows a small scale AIS-based SHM network deployed on a beam structure. The network consists of three sensor units, a knowledge base, and a network component for clonal selection. A mobile agent system is installed locally in each network component. When a mobile monitoring agent detects damage in a sensor unit (for example, in the middle sensor unit in Fig. 1), it communicates with the knowledge-base agent to confirm the damage. If the damage is confirmed, the mobile monitoring agent sends a request to the clonal selection component for cloning. The cloned mobile monitoring agents are sent to the sensor units close to the location where the damage is detected to find out damage affected area.

To increase damage detection probability, reduce response time, and extend network lifespan, this work studies multi-objective agent control in AIS-based monitoring networks. The optimal control of mobile monitoring agents includes the optimization of agent generation and agent distribution. In agent generation control, two-objective functions are defined to minimize the response time for a group of mobile monitoring agents to perform damage diagnosis in a sub-network and maximize the average affinity level of the generated mobile monitoring agents with the sensor data feature vector. A multi-objective optimization algorithm is adopted to find the appropriate values of clonal rate and mutation value for the control of agent generation. In agent distribution control, a genetic algorithm is employed to search a best candidate sensor node for the deployment of a mobile monitoring agent to increase the damage detection probability and extend the network lifetime by applying a fitness punishment strategy to balance the energy consumption among sensor nodes over a network.

This section introduces a network framework consisting of agent-based network middleware and high computational power sensor nodes to support the agent management and distributed damage diagnosis.

2.2. High computational power sensor node

A high computational power sensor node is designed for the distributed damage diagnosis which requires intensive numerical computation. The high computational power of the sensor node is

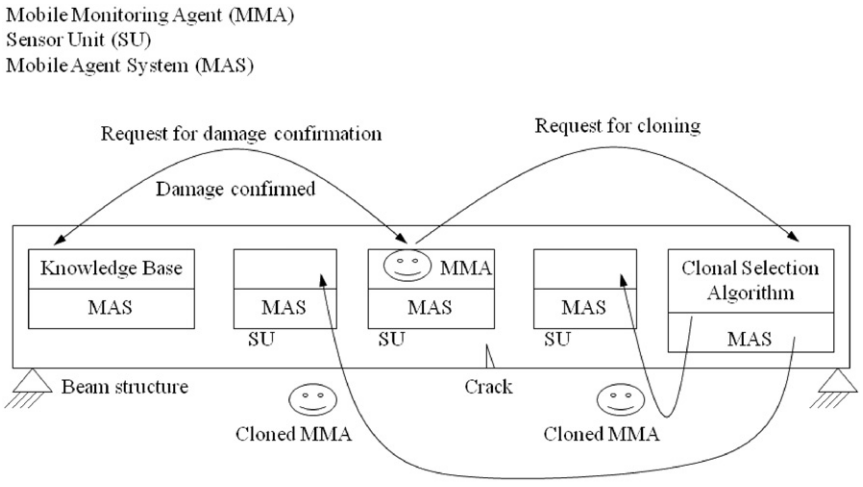

Fig. 1. An AIS-based SHM sensor network.

achieved by the integration of sensor node hardware computing resources and the embedded numerical computing software packages. The sensor node consists of three circuit boards: a SHM sensor board, Gumstix embedded computer, and a WiFi transmission board. The Gumstix embedded computer is one of the world's smallest full function miniature computers. The size of the sensor node is about 4 (in.) × 2.4 (in.) × 0.65 (in.). The sensor board is designed and fabricated by our research group to meet the structural health monitoring sensing requirement (Chen and Wang, 2008). The sensor board connects to a numbers of sensors, including accelerometers, strain gauges, humidity, and temperature sensors.

A numerical library module is integrated in the Gumstix-based sensor nodes. The numerical library module provides computational building blocks to construct SHM analysis algorithms. It contains CLAPACK library and a Utility Function Library. The CLAPACK library is a C version of LAPACK library, which provides routines for solving systems of linear equations, linear least-squares problems, eigenvalue problems, and singular value problems. The utility functions in the Utility Function Library are designed to perform subtasks of SHM analysis or numerical computation that is not available in CLAPACK library, for example, Fast Fourier Transform. The existing open source numeric libraries such as Numerical Recipes in C and the GNU Scientific Library (GSL) can be used to implement these utility functions. The numerical accuracy of these open source libraries is comparable with commercial software packages, such as Matlab.

2.3. Mobile agent-based sensor network middleware

A mobile agent is a software agent that is capable of migrating from one node to another in a network and resumes the execution in the new node. The migration and execution of mobile agents are supported by a mobile agent system. We adopt Mobile-C as the mobile agent system (Chen et al., 2006, 2009, 2008) in AIS-based monitoring networks. The main components of Mobile-C include agent management system, agent communication channel, directory facilitator, agent security manager, and agent execution engine. The effectiveness of Mobile-C has been verified in a number of systems, including a retrofitted automation work cell (Nestinger et al., 2010), a real-time traffic detection system (Chen et al., 2009), a mobile robotic system (Chou et al., 2007), and a structural health monitoring network (Chen and Liu, 2010). Mobile-C in sensor nodes can host both stationary agents and mobile agents. Stationary agents are those staying in the sensor nodes where they are created, such as knowledge-base agent and the clonal selection agent. Mobile agents are those created during the system operation for monitoring purpose. In a mobile agent-based monitoring network, a remote user can dispatch mobile monitoring agents to sensor nodes in the network. These monitoring agents equip with data analysis and damage diagnosis algorithms and can roam over the network and perform damage diagnosis at sensor nodes where they visit.

3. Pattern recognition-based damage detection and classification

3.1. Feature extraction

The structural damage patterns are represented by feature vectors extracted from the dynamic response data of a structure (Chen and Zang, 2009). The feature vector of a time series sensor data is formed by coefficients of an auto-regressive (AR) model of the time series. To reduce environmental effects, sensor data Z are standardized by

$$y_{ij} = \frac{z_{ij} - \mu_i}{\sigma_i}, \quad j = 1,2,\ldots,n,$$

where μ_i and σ_i are the mean and standard deviation of the time series $\bar{z}_i$. To extract feature vectors for a local area, time series sensor data sets from multiple sensors are reduced to lower dimensions by the Principal Component Analysis (PCA) method. The compressed time series X is then fitted to an auto-regressive (AR) model of order p as

$$x_k = \sum_{i=1}^{p} \alpha_i x_{k-i} + r_k, \quad k = p+1,\ldots,n \tag{1}$$

where α_i, $i = 1, 2, \ldots, p$ are the coefficients of the AR model; the vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_p)^T$, a collection of the AR coefficients, is selected as the feature vector of the sensor data Z.

3.2. Damage detection using memory cells

In artificial immune damage pattern recognition, each damage pattern is represented by a set of representative feature vectors (Chen and Zang, 2009). These representative feature vectors are called memory cells for the corresponding damage pattern. Each mobile monitoring agent carries memory cells during distributed damage diagnosis. In a structural damage detection test (Chen, 2010), a scaled steel bridge was used. Sensor nodes and accelerometers were mounted on the beams along the bridge, six sensor nodes each side. Each sensor node was located about 1 meter away from its neighbor sensor nodes. Structural damage was simulated by removing one cross member at the center of the bridge. When a mobile monitoring agent arrives at a sensor node, it reads the acceleration data from sensor node and builds an AR model for the sensor data. Based on the AR coefficients, the feature vector of the sensor data is formed, and the Euclidean distances from sensor data feature vector to the memory cells of the monitoring agent are calculated. Classification algorithms such as k-nearest neighbor (kNN) algorithm can be used to detect if damage is presented in the structure. For a given sensor data feature vector x, the nearest neighbor rule is summarized as follows (Theodoridis and Koutroumbas, 2008). (1) Out of the N memory cell feature vectors, identify the k-nearest neighbors to the sensor data feature vector x. The number of k is general not to be a multiple of the number of classes M. (2) Out of these k samples, identify the number of vectors k_i that belong to class ω_i, $i = 1,2,\ldots,M$, $\sum_i k_i = k$. (3) Assign x to the class ω_i with the maximum number k_i of samples.

4. Optimization of agent generation and distribution

Multi-objective optimization method is used to control agent generation and distribution in this study. A multi-objective optimization problem has a number of objective functions that need to be minimized or maximized under a number of constraints. The general form of a multi-objective optimization is described below (Deb, 2001):

$$\begin{aligned} &\textit{Minimize/maximize} \quad f_m(X), \quad m = 1,2,\ldots,M;\\ &\textit{subject to} \quad g_j(X) \geq 0, \quad j = 1,2,\ldots,J;\\ &h_k(X) = 0, \quad k = 1,2,\ldots,K;\\ &x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1,2,\ldots,n. \end{aligned} \tag{2}$$

where $f_m(x)$ are M number of objective functions; $g_j(x)$ and $h_k(x)$ are inequality and equality constrains; and the n solutions $X = (x_1, x_2, \ldots, x_n)^T$ are restricted by lower and upper boundaries $x_i^{(L)}$ and $x_i^{(U)}$.

There are a number of computational techniques and methods for solving optimization problems, such as hill climbing (Russell and Norvig, 2003), ant colony optimization (ACO) (Dorigo et al., 1996), particle swarm optimization (PSO) (Kennedy and Eberhart,

1995; Poli, 2008), and genetic algorithms (GAs) (Goldberg, 1989). Hill climbing is a relatively simple optimization technique for searching a local optimal solution. Hill climbing works in some situations although more advanced optimization algorithms may result in better solutions. In Hill climbing algorithm, an initial solution is randomly chosen. The algorithm then searches the neighborhood of the current solution. If the neighborhood solution is better than the current solution, the neighborhood solution is used to substitute the current solution. This process is repeated until no more improvement can be made to the current solution. Ant colony optimization is inspired by the behavior of ants seeking a path between their colony and a source of food. Good solutions are found through the cooperation of the artificial ants. ACO is typically applied for problems of searching good paths through graphs, such as the traveling salesman problem and packet routing in networks. Particle swarm optimization is initially developed based on the social behavior of bird flocks. PSO optimizes a problem by having a population of candidate solutions and moving these particles around in the search-space according to simple mathematical formula. The movements of the particles are guided by the best found positions in the search-space. PSO method performs well for some continuous optimization problems. Genetic algorithm is a population-based optimization algorithm. It maintains a population of candidate solutions. Due to its population-based nature, it belongs to the family of global optimization algorithms. Multiple candidate solutions in each iteration and global optimization characteristics of the genetic algorithm make it appropriate for our application since we can apply fitness punishment strategies on candidate solutions and find global optimal solutions.

The basic idea of genetic algorithm is from evolutionary theory: species have different capabilities of living in the nature. In genetic algorithm (Goldberg, 1989), each solution will be represented in the form of strings. A string is called chromosome and consists of a number of digits, which represents one or more features of the solution. There should be an encoding strategy working well enough so the chromosome is able to carry all the features of the solution. Actually, one solution is corresponding to one type of chromosome. A group of chromosomes are called population which is usually randomly initialized. After a number of generations of evolution, a single solution will dominate the other solutions. Each generation of chromosomes is created through three processes: selection, crossover, and mutation. There exist a number of multi-objective optimization implementations. A fast and elitist multi-objective genetic algorithm, NSGA-II (Deb et al., 2002), is one of those algorithms which has been applied to many applications. NSGA-II employs non-dominated sorting incorporating elitism. The goal of NSGA-II algorithm is to find a population of solutions that are as close as possible to the Pareto-optimal front and as diverse as possible. The NSGA-II algorithm consists of four steps. The first step combines parent and offspring, and performs a non-dominated sorting to identify different front for each solution. The first front is a non-dominated set in the current population and the second front is dominated by the individuals in the first front, and so on. In the second and third steps, a new parent set is formed based on the rank of front level and the crowding distances. The solutions with higher order of front levels will firstly be selected into new parent set. The selection of solutions at same front level is based on crowding distances. The solutions with large crowding distances will be included to increase the diversity of the solution population. In the fourth step, the selected parents generate offspring by using the crowed tournament selection, crossover, and mutation operations.

4.1. Optimization of agent generation in artificial clonal selection process

In AIS-based monitoring networks, adaptive immune response is a mechanism to manage the amount and type of monitoring agents through clonal selection. Fig. 2 shows the clonal selection process in which the activated mobile monitoring agent is cloned and mutated. The cloned mobile monitoring agents are sent to sensor nodes close to the location where the damage is detected to find out the damage affected area. Assume that the local communication uses Zigbee and the long distance communication uses WiFi. The cloned mobile monitoring agents will visit N number of sensor nodes for further damage diagnosis.

The number of cloned monitoring agents, n, depends on the clonal rate CR and the affinity between the feature vector of the mobile monitoring agent and the sensor data feature vector where the damage is detected. The number of cloned agents can be calculated by

$$n = round(CR * aff(\beta,\alpha)) \tag{3}$$

where the CR is an integer value used to control the number of agent clones allowed for the activated mobile monitoring agent. Let $ma \cdot f = \beta = (\beta_1,\beta_2,\ldots,\beta_p)^T$ and $sd \cdot f = \alpha = (\alpha_1,\alpha_2,\ldots,\alpha_p)^T$ denote the feature vector of a mobile monitoring agent and the feature vector of the sensor data, respectively. The affinity between the feature vector of the mobile monitoring agent and the sensor data feature vector is defined as

$$aff(\beta,\alpha) = 1 - \frac{1}{2} \times dist(\beta,\alpha) \tag{4}$$

where $dist(\beta,\alpha)$ is the Euclidian distance between vectors α and β as

$$dist(\beta,\alpha) = \sqrt{\sum_{i=1}^{p} (\beta_i - \alpha_i)^2} \tag{5}$$

The cloned monitoring agents are mutated to increase the diversity of the generated mobile monitoring agents. The mutation is performed by mutating the feature vectors of the cloned

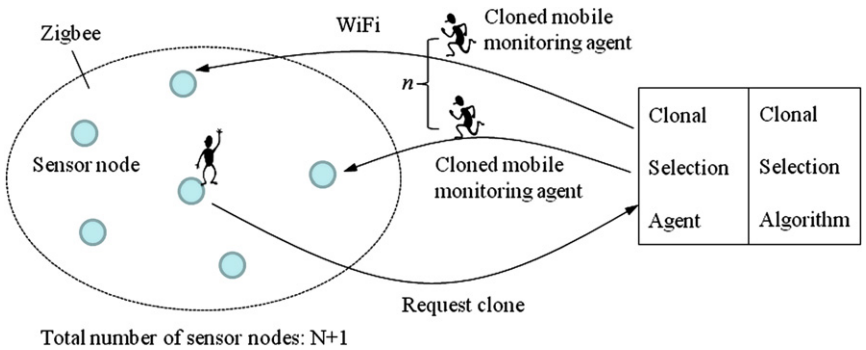

Fig. 2. Artificial immune response.

monitoring agents. Let $(ma_{mutated}).f = \gamma = (\gamma_1, \gamma_2, \ldots, \gamma_p)^T$ denote the feature vector of a mutated monitoring agent. The mutation is performed as

$$\gamma = \beta + MV * (\phi_1, \phi_2, \ldots, \phi_P)^T \quad (6)$$

where MV is the mutation value and ϕ_i is a normal random value within the range of $[-1, 1]$.

The goal of the presented multi-objective optimization algorithm is to find appropriate values of CR and MV with following objective functions.

1. Minimize response time: the response time is defined as the time needed for the cloned mobile monitoring agents to further diagnose structural damage at the N number of sensor nodes close to the location where the damage is detected. The response time includes both mobile agent transmitting time and damage diagnosis computational time:

$$T = nST_w + T_c + \left[\frac{(N - N\%n)}{n} - 1\right] * ST_z + \left[\frac{(N - N\%n)}{n} - 1\right] * T_c + \left\lceil \frac{N\%n}{n} \right\rceil * [ST_z + T_c] + T_{ga} \quad (7)$$

where T_c is the computational time for a mobile agent to execute damage diagnosis program in a sensor node. T_w and T_z are the time needed for transmitting one byte of data in WiFi and Zigbee networks. T_{ga} is the time needed for the genetic algorithm to find non-dominated solutions. S is the size of a mobile agent in bytes. The number of cloned mobile monitoring agent is n. Modulus function $N\%n$ finds the remainder of the division N/n. Since $n < N$, n number of cloned agents needs to move $[((N - N\%n)/n) - 1]$ times in the Zigbee network and part of the cloned agents needs one more move to cover N number of sensor nodes.

2. Maximize average affinity level (minimize average distance) among the feature vectors of cloned mobile monitoring agents and the sensor data feature vector:

$$D = \frac{1}{n}\sum_{i=1}^{n} dist((ma_{i,mutated}) \cdot f, sd \cdot f) = \frac{1}{n}\sum_{i=1}^{n} dist(\gamma_i, \alpha) \quad (8)$$

where $dist((ma_{i,mutated}) \cdot f, sd \cdot f)$ is the Euclidean distance between the feature vectors of the ith cloned mobile monitoring agent and the sensor data feature vector.

3. The optimization is subjected to following constraints:

$$n \le N; \quad 0 < CR \le 30; \quad 0 < MV < 1 \quad (9)$$

The range of CR is determined from previous research results. When the value of CR is less than 30, the number of memory cells is within a reasonable range.

4.2. Non-dominated solutions for two defined objective functions

To find out non-dominated solutions for defined objective functions, the fast and elitist multi-objective genetic algorithm, NSGA-II, is used for the simulation of this optimization problem. The values of simulation parameters are listed in Table 1. The non-dominated solutions found by the NSGA-II algorithm are shown in Fig. 3. The corresponding solution values form a decision space as shown in Fig. 4. From the decision space, we can see that when the mutation value MV is high, the value of clonal rate CR is relative small. When the MV value is small, the CR value is high.

To find out good solutions for the system, the impact of algorithm parameters on the system performance is investigated. Two criteria are usually used to evaluate the goodness of solutions (Deb, 2001): (1) the non-dominant solutions should be as close to the Pareto front as possible and (2) the non-dominated solutions are as diverse as possible. The diversity helps to achieve more uniform distribution of non-dominated solutions along the Pareto front. Fig. 5 shows that non-dominated solutions move towards the origin as the number of generation increases. This means that the larger the number of generations is, the closer the non-dominated solutions to the Pareto-optimal solutions are. The distance between consecutive solutions can be used to evaluate how well the solutions are uniformly spaced (Deb, 2001). The spacing metric S

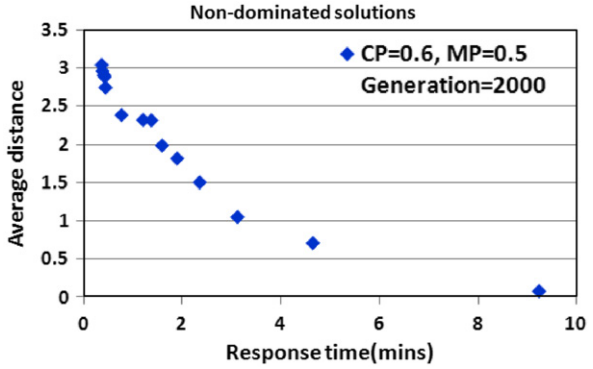

Fig. 3. Non-dominated solutions.

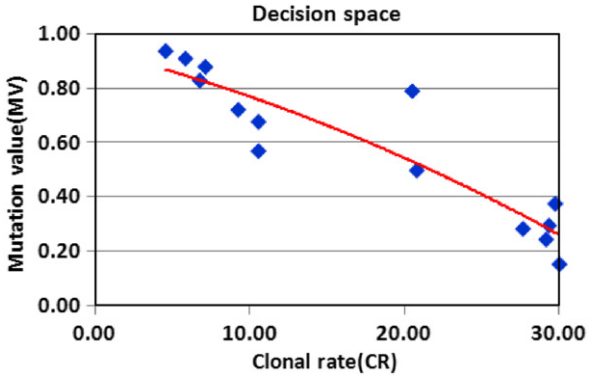

Fig. 4. Decision space.

Table 1
Simulation parameters.

Simulation parameters	*Value*	*Unit*
Population size	40	
Number of generations	2000	
Crossover probability (CP)	0.6	
Mutation probability (MP)	0.5	
T_c – computational time for a mobile monitoring agent to execute damage diagnosis program	55	s
T_w – time needed for transmitting one byte of data in WiFi network	0.727	μs
T_z – time needed for transmitting one byte of data in Zigbee network	0.032	ms
T_{ga} – time needed for the genetic algorithm to find non-dominated solutions	4	s
The size of a mobile monitoring agent in bytes, S	1900	byte

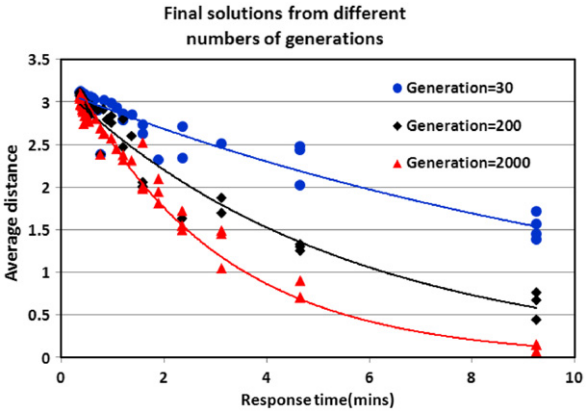

Fig. 5. Final solutions with different number of generations.

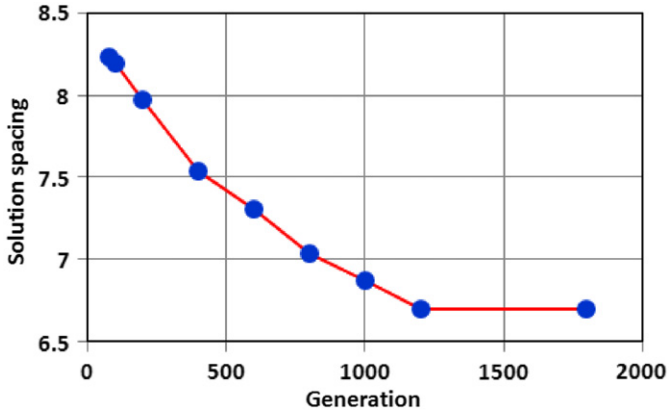

Fig. 6. Spacing of non-dominated solutions vs. number of generations.

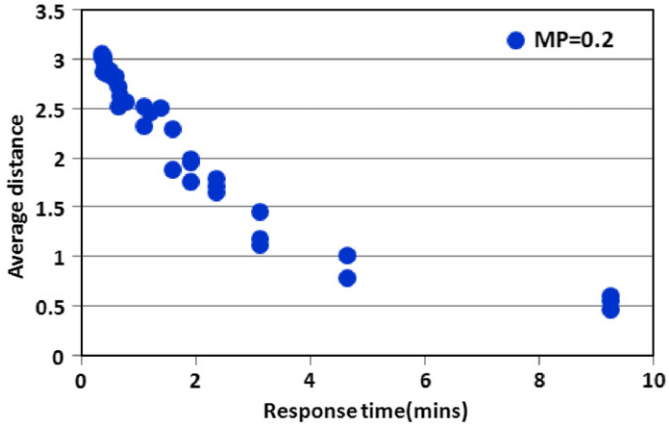

Fig. 7. Final solutions when mutation probability is 0.2.

is defined as follows:

$$S=\sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i-\overline{d})^2} \tag{10}$$

where Q is the number of non-dominated solutions, $d_i=\min_{k\in Q\wedge k\neq i}\sum_{m=1}^{M}|f_m^i-f_m^k|$ and $\overline{d}$ is the mean value of the above distance measure $\overline{d}=\sum_{i=1}^{|Q|}d_i/|Q|$. In the equation of d_i, M represents the number of objective functions. In our application, f_1 represents response time and f_2 represents average affinity level.

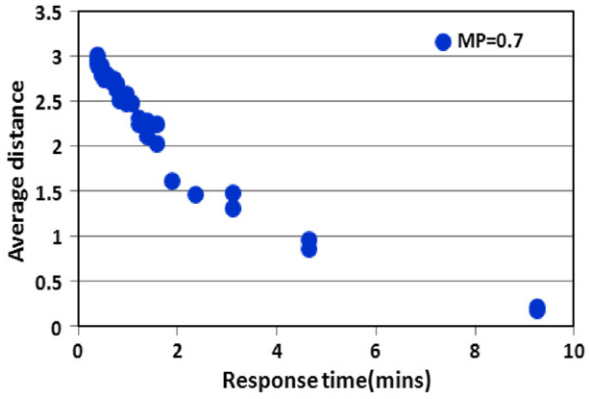

Fig. 8. Final solutions when mutation probability is 0.7.

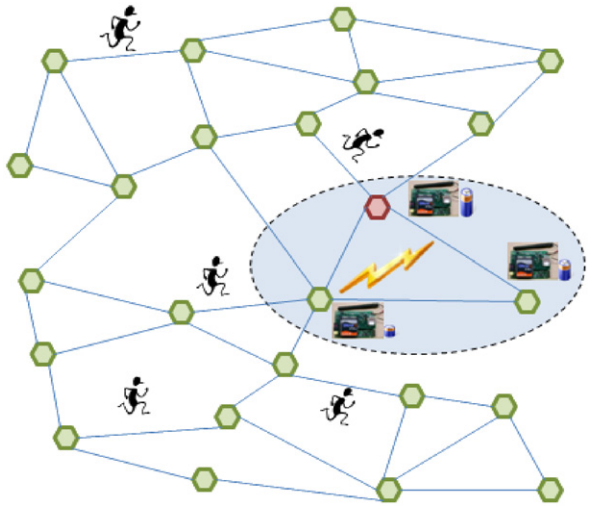

Fig. 9. Agent distribution based on affinity and remaining battery capacity of sensor nodes.

A smaller value of S indicates more uniform spacing of non-dominated solutions. Fig. 6 shows that the spacing decreases when the number of generations increases. When the number of generations is above 1200, the value of S remains at similar level. In addition to number of generations, the mutation probability of the genetic algorithm also impacts the spacing of the solutions. Comparing with Figs. 7 and 8, the larger value of the mutation probability has better solution spacing. This is because the larger mutation probability improves the diversity of solutions.

4.3. Optimization of agent distribution

Optimization of agent distribution is to find out which sensor node is the best candidate for a mobile monitoring agent to visit so that the damage detection probability can be increased and monitoring network lifetime can be extended. Due to energy constraints, each sensor node has its own lifespan. The remaining battery capacity in each sensor node impacts the lifetime of the whole monitoring network. The goal of the optimization algorithm is to distribute mobile monitoring agents to the node where they are needed for damage detection and prolong the network lifetime.

Fig. 9 shows the basic concept of the presented GA algorithm for optimizing agent distribution. Assume that there is more than one sensor node in the region where the abnormal condition is occurred. Based on the fitness values (the distances amongst the sensor data feature vectors and the memory cells of a mobile

monitoring agent), the GA algorithm determines at which sensor node the mobile agent has the highest probability to detect anomaly. The algorithm also takes the remaining battery capacity of each sensor node into consideration when making the agent distribution decision. Sensor nodes with low remaining battery capacities have low privilege to accept a mobile monitoring agent. This distribution strategy extends the overall lifespan of a monitoring network.

4.3.1. Battery energy model

For battery-powered sensor nodes, the remaining battery capacity of a sensor node can be calculated by Eq. (11) (Peng and Pedram, 2006):

$$RC = SOC \cdot SOH \cdot DC \tag{11}$$

where *SOC* (state of charge) indicates how much energy left in a battery; *SOH* (state of health) is the ratio of the full charge capacity of a battery to its design capacity (DC). The equations for calculating *SOC*, *SOH*, and *DC* are shown in Eqs. (12), (15), and (16) (Peng and Pedram, 2006). The *SOC* is defined as

$$SOC = 1 - \frac{\left[\frac{1}{b_1} - \left(\frac{1}{b_1} - SOH^{b_2} \cdot DC^{b_2}\right)\exp\left(\frac{\Delta v_m - \Delta v}{\lambda}\right)\right]^{\frac{1}{b_2}}}{SOH \cdot DC} \tag{12}$$

where $\Delta v = VOC_{init} - v$ and $\Delta v_m = VOC_{init} - v_{cut-off}$, VOC_{init} indicates initial open-circuit voltage; $v_{cut\text{-}off}$ indicates a voltage threshold below which the end of discharge is considered being reached; $\lambda = (2RT)/(nF)$, where R indicates gas constant; T indicates current temperature; n indicates number of electronics transferred; and F indicates Faraday's constant; In Eq. (12), b_1 and b_2 are functions of discharge rate and temperature, respectively (Peng and Pedram, 2006),

$$b_1(i,T) = d_{11}\exp\left(\frac{d_{12}}{T}\right) + d_{13} \tag{13}$$

$$b_2(i,T) = \left(\frac{d_{21}}{T + d_{22}}\right) + d_{23} \tag{14}$$

where $d_{jk}(i) = \sum_{z=0}^{4} m_z(d_{jk}) \cdot i^z$ $j = 1,2$, $k = 1,2,3$, and $m_z(d_{jk})$ are constant coefficients based on curve fitting. The values of SOH and DC can be calculated by

$$SOH = \left\{\frac{1 - \exp((r_n i - \Delta v_m)/\lambda)}{1 - \exp((r_0 i - \Delta v_m)/\lambda)}\right\}^{1/b_2} \tag{15}$$

$$DC = \left\{\frac{1}{b_1}\left[1 - \exp\left(\frac{r_0 \cdot i - \Delta v_m}{\lambda}\right)\right]\right\}^{1/b_2} \tag{16}$$

where $r_n = r(i,T,n_c,T')$ represents the effect of the Ohmic overpotential and electrode reaction potential when the discharged current is constant; i indicates current; T indicates the current temperature; T' indicates the temperature that the battery has experienced in previous cycle; and n_c is the number of electrons transferred during the process of battery charging; if $n_c = 0$, $r_n = r_0$.

Based on above equations, the remaining capacity of a battery can be calculated with three measurable variables: battery voltage v, current i, and temperature T.

4.3.2. Genetic algorithm for searching agent destination

A GA-based algorithm is developed to dynamically search a destination sensor node for a mobile monitoring agent based on the remaining battery capacity of sensor nodes and the sensor data feature vectors. The selected sensor node satisfies two criteria: (1) the Euclidean distance between the sensor data feature vector and the memory cells of the mobile monitoring agent is small; (2) the remaining battery capacity of the sensor node is higher than a pre-defined threshold.

Fig. 10 shows a GA-based algorithm for a mobile monitoring agent to search a candidate sensor node to visit. The sensor data feature vectors and the remaining battery capacities of sensor nodes are used to choose an appropriate sensor node. The output of the genetic algorithm is the ID of selected sensor node. Sensor IDs are encoded into chromosomes in binary format. In the experimentation discussed in this paper, 8 digits of binary code are used to represent the sensor IDs. Initial population with 10 individuals is randomly picked up in the sensor node population pool. Each individual is an 8-bit binary string. The fitness value of the individuals is calculated based on the Euclidian distance between

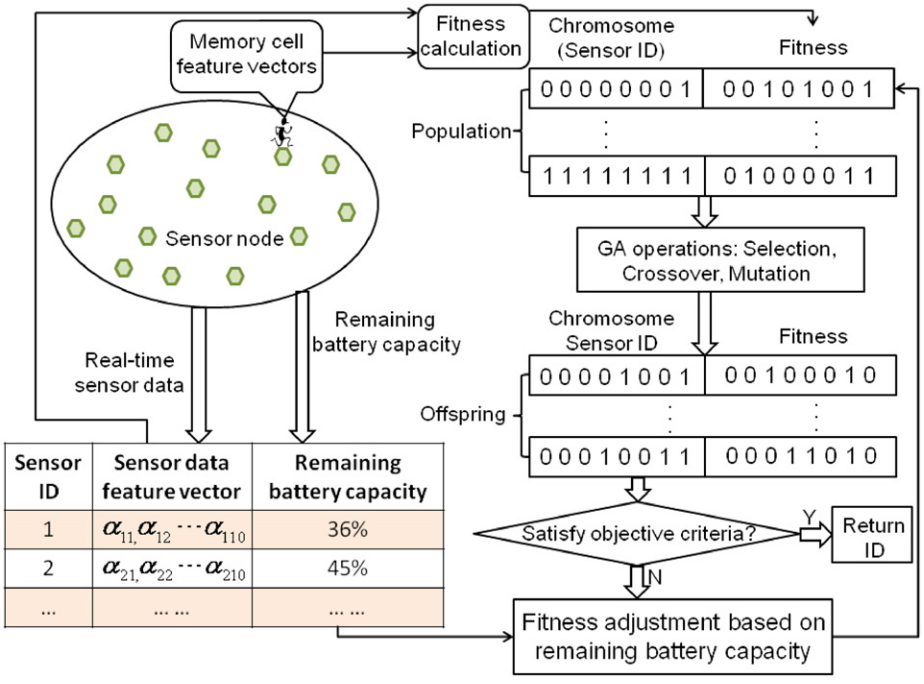

Fig. 10. Genetic algorithm for searching a suitable sensor node.

the sensor data feature vector and the memory cell feature vector of a mobile monitoring agent.

Let $\beta_i = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{ip})^T$ denote the feature vector of the ith mobile monitoring agent and $\alpha_j = (\alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jp})^T$ the sensor data feature vector of the jth sensor node. The Euclidian distance between these two feature vectors can be calculated by

$$dist(\beta_i - \alpha_j) = \sqrt{(\beta_{i1} - \alpha_{j1})^2 + (\beta_{i2} - \alpha_{j2})^2 + \cdots + (\beta_{ip} - \alpha_{jp})^2} \quad (17)$$

Based on fitness values, several sensor individuals with high fitness to the memory cell of the mobile monitoring agent are selected. The selected sensor individuals subject to two GA operations: crossover and mutation, to generate offspring of the parent generation. The impact of crossover probability and mutation probability on the number of iterations to find a destination sensor node and the ratio of good solutions is discussed later in this paper.

To balance energy consumption among sensor nodes in the monitoring network, a fitness punishment strategy is designed in the genetic algorithm. If a sensor node whose remaining battery capacity is below a pre-defined threshold, the fitness value of the sensor node is reduced by multiplying a penalty factor. This punishment greatly impacts the fitness value of the sensor node; as a result, the sensor node will most likely be eliminated from the population of the next generation.

4.3.3. *Effects of fitness punishment strategy on the network lifetime*

To analyze the impact of the fitness punishment strategy on the network lifetime, a simulation is performed to find out appropriate penalty factor values and remaining energy threshold. The network lifetime is defined as the time duration from the beginning of the operation to the moment when one of the sensor node's remaining energy capacity is equal to or less than the defined energy threshold. Assume that 255 sensor nodes are deployed evenly in a 15×15 m^2 area. Each sensor node collects measurement data from sensors. The sensor network periodically dispatches mobile monitoring agents to the network for distributed anomaly detection. The total energy consumption of the network includes sensing energy and mobile agent deployment energy-*Ea*. The energy consumed for deploying a mobile agent consists of mobile agent transmitting energy and computation energy. In our monitoring network, the energy consumption of transmitting a mobile agent to a sensor node and performing damage diagnosis is about 115 times of energy consumption for collecting 2000 data bytes of acceleration data. Since the agent deployment energy is much higher than sensing energy, agent deployment energy consumption is a major factor considered in the simulation. Fig. 11 shows the percentage of network lifetime extension vs. the values of penalty factor and the energy threshold for applying fitness punishment.

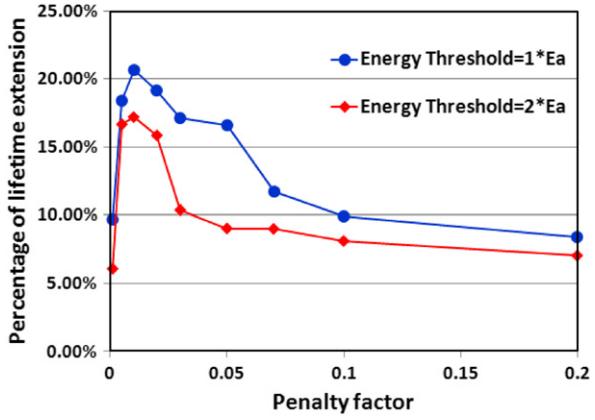

Fig. 11. Percentage of lifetime extension vs. penalty factor values.

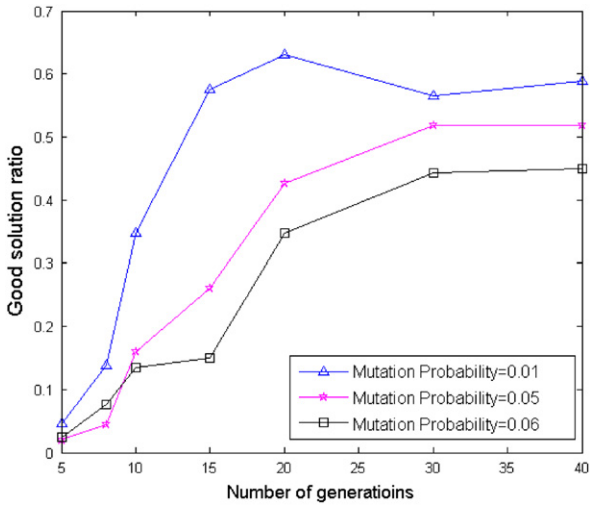

Fig. 12. Good solution ratio vs. mutation probability.

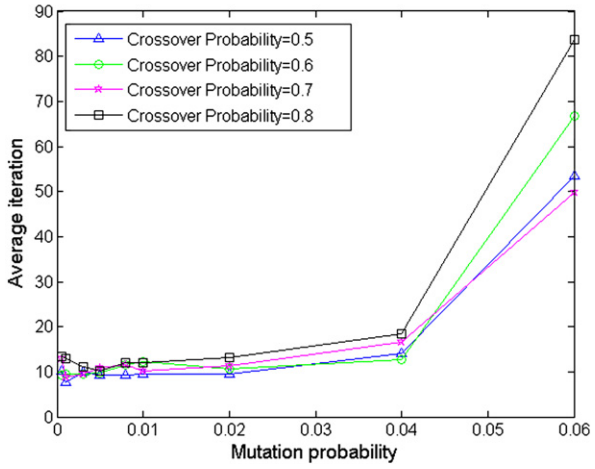

Fig. 13. Average iterations based on crossover probability and mutation probability.

4.3.4. *The impact of crossover probability and mutation probability on the good solution ratio and number of iterations*

The presented GA algorithm is implemented by the Genetic Algorithm Toolbox for Matlab from University of Sheffield. To investigate the impact of parameters on the performance of the algorithm, different values of mutation probability and crossover probability are used to test the ratio of good solutions and the convergence speed. Fig. 12 shows the impact of the value of mutation probability on the good solution ratio with a fixed crossover probability. The horizontal axis represents the number of generations and the vertical axis represents the good solution ratio. Good solution ratio is defined as the ratio of the number of good solutions to the size of population. The crossover probability used in this test is equal to 0.7. From Fig. 12, we can see that the ratio of good solutions increases quickly in the first 30 generations, but, no significant change after that.

Fig. 13 shows the average number of iterations the genetic algorithm takes to find the best candidate sensor node. The horizontal axis represents the mutation probability and the vertical axis represents the average number of iterations from 12 trials. The curves with different colors are corresponding to different values of crossover probability. From Fig. 13, we can see that the number of iterations increases significantly when the mutation probability varies from 0.04 to 0.06. For each crossover probability, the average number of iterations increases as the mutation probability increases.

5. Conclusions

Multi-objective optimization algorithms for the control of mobile monitoring agents in artificial-immune-system-based monitoring networks are presented in this paper. The developed algorithms optimize agent generation and distribution by increasing damage detection probability, reducing response time, and extending network lifespan. The amount and type of generated monitoring agents are tuned to the detected damage. The selected sensor node for a mobile monitoring agent to visit is based on the affinity between the sensor data feature vector and the memory cells of the monitoring agent. In addition, the remaining battery capacity of a sensor node will impact the selection decision. A sensor node with high remaining battery capacity has high probability to receive a mobile monitoring agent. This selection strategy not only ensures the detection effectiveness, but also balances the remaining battery capacity of sensor nodes across the monitoring network. The simulation results show that the number of generations affects the goodness and the spacing of non-dominated solutions. In addition, the crossover probability and the mutation probability will also impact the performance of the algorithm.

Acknowledgements

This research is supported by the National Science Foundation under Grant #: 1049294 and Michigan Tech Research Excellence Fund. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring institutions.

References

Bhondekar AP, Vig R, Singla ML, Ghanshyam C, Kapur P. Genetic algorithm based node placement methodology for wireless sensor networks. International multiconference of engineers and computer scientists 2009 (IMECS 2009). Hong Kong, China: International Association of Engineers; 2009. p. 106–12.

Chen B. Agent-based artificial immune system approach for adaptive damage detection in monitoring networks. Journal of Network and Computer Applications 2010;33:633–45.

Chen B, Cheng HH, Palen J. Mobile-C: a mobile agent platform for mobile C/C++ agents. Software-Practice & Experience 2006;36:1711–33.

Chen B, Cheng HH, Palen J. Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. Transportation Research Part C: Emerging Technologies. 2009;17:1–10.

Chen B, Linz DD, Cheng HH. XML-based agent communication, migration and computation in mobile agent systems. Journal of Systems and Software 2008;81:1364–76.

Chen B, Liu W. Mobile agent computing paradigm for building a flexible structural health monitoring sensor network. Computer-Aided Civil and Infrastructure Engineering 2010;25:504–16.

Chen B, Wang J. Design of a multi-modal and high computation power wireless sensor node for structural health monitoring. 2008 IEEE/ASME international conference on mechatronic and embedded systems and applications. Beijing, China, 2008.

Chen B, Zang C. Artificial immune pattern recognition for structure damage classification. Computers & Structures 2009;87:1394–407.

Chou Y-C, Ko D, Cheng HH. Embeddable mobile-C for runtime support of code mobility in multi-agent systems. International design engineering technical conferences & computers and information in engineering conference. Las Vegas, Nevada: ASME; 2007.

Deb K. Multi-objective optimization using evolutionary algorithms. Wiley; 2001.

Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6:182–97.

Delves PJ, Martin SJ, Burton DR, Roitt IM. Roitt′s essential immunology. 11th ed. Blackwell Publishing; 2006.

Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics. 1996;26:29–41.

Ferentinos KP, Tsiligiridis TA. Adaptive design optimization of wireless sensor networks using genetic algorithms. Computer Networks 2007;51:1031–51.

Goldberg DE. Genetic algorithms in search, optimization and machine learning. Addison Wesley Longman; 1989.

Hussain S, Matin AW, Islam O. Genetic algorithm for hierarchical wireless sensor networks. Journal of Networks 2007;2:87–97.

Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, 1995. p. 1942–8.

Khanna R, Liu H, Chen H-H. Self-organization of sensor networks using genetic algorithms ICC ′06 IEEE international conference on communications. Istanbul, Turkey, 2006. p. 3377–82.

Maslov IV, Gertner I. Multi-sensor fusion: an evolutionary algorithm approach. Information Fusion. 2006;7:304–30.

Neal M, Trapnell BCJ. Go Dutch: exploit interactions and environments with artificial immune systems. In: Flower DR, Timmis J, editors. Silico immunology. USA: Springer; 2007.

Nestinger S, Chen B, Cheng HH. A mobile agent-based framework for flexible automation systems. IEEE/ASME Transactions on Mechatronics 2010;15:942–51.

Peng R, Pedram M. An analytical model for predicting the remaining battery capacity of lithium-ion batteries. IEEE Transactions on Very Large Scale Integration 2006;14:441–51.

Poli R. Analysis of the publications on the applications of particle swarm optimisation. Journal of Artificial Evolution & Applications 2008:10685175 2008:10.

Qinru Q, Qing W, Burns D, Holzhauer D. Lifetime aware resource management for sensor network using distributed genetic algorithm ISLPED′06: Proceedings of the 2006 international symposium on low power electronics and design. Tegernsee, Germany: IEEE; 2006. p. 191–6.

Russell SJ, Norvig P. Artificial intelligence: a modern approach. 2nd ed Upper Saddle River, New Jersey: Prentice Hall; 2003. p. 111–4.

Theodoridis S, Koutroumbas K. Pattern recognition. Academic Press; 2008.