World Scientific Research Journal (WSRJ) ISSN: 2472-3703
www.wsr-j.org

Location Optimization of Actuator Nodes Based on Genetic Algorithm

Dejun Peng[a], Yanjun Wang, Yangsuo Wan

School of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China

[a]pdjdejun@163.com

Abstract: In the Wireless Sensor and Actuator Network (WSAN), the actuator can be placed anywhere within the sensor communication range. In order to provide maximum communication coverage, this paper considers the communication connection between the actuator nodes, the priority of the different sensors and then puts forward the optimization method of location distribution of actuator nodes based on genetic algorithm. Based on this approach, a visualization software was developed for customizing the deployment scenario. The experimental results show that the improved genetic algorithm has strong adaptability to different constraint control, and the coverage is up to 100% under unconstrained and dynamic constraints.

Keywords: wireless sensor and actuator networks, location distribution, communication coverage, genetic algorithm

---

1. INTRODUCTION

Wireless Sensor and Actuator Network (WSAN) is a kind of derivation of Wireless Sensor Network (WSN) [1-4]. Traditional WSN can only sense the external environment (realized by sensor nodes) simply, can not make real-time autonomous decisions on the sensing results, and can not meet the needs of large-scale intelligent management in some applications. With the development of automatic control technology, people hope that WSN can automatically make decisions and independent controlling reacting to the external events of environment under unattended conditions, so that it can become a kind of intelligent control network system. WSN is a special wireless ad-hoc network composed of a large number of sensor nodes and actuator nodes [5, 6].

In the WSN, the sensor node is a low cost, low power, multi-functional wireless sensor devices, which contains data acquisition, data processing and data transmission functions, but these functions are in need of the consumption of the sensor's energy, affecting the life of the sensor. Through deploying resource rich actuator nodes, the method of transferring complex calculations and controls from the sensor node to the actuator node can save the energy of the

sensor. However, the cost of actuator is more expensive than the sensor, and take up more space, and the deployment is more complex, so that it is impossible to deploy many actuators. The position of the actuator node is determined by the communication coverage of the sensor to the surrounding sensor nodes and the delay time of the data reception, and the expected target is achieved with as few actuators as possible [7]. The authors [8] consider not only the minimum number of actuator nodes to reach the maximum communication coverage of the sensor nodes, but also the internal communication connections between the actuator nodes. But the average coverage is less than 80% with the C2AP method which is introduced to deploy actuator nodes, so there is still much room for improvement. And the introduced algorithm can not consider the priority of different sensors, some high priority sensors can not be bound to the corresponding actuator.

Melodia [9] proposes an event-driven sensor-actuator coordination framework. They also discuss the actuator-actuator coordination problem and propose a localization solution based on the competition mechanism. In [10,11], the genetic algorithm is used to search for the optimal solution, but the communication connection between the actuator and the actuator is not taken into account, which can not meet the requirements of some tasks that require the cooperation between the actuators to complete the control task. In [7, 12, 13], simulated annealing algorithm is adopted to solve the problem of actuator node deployment, but it does not consider some constraints in the actuator deployment, such as the communication load of each actuator node.

In this paper, by studying the actuator node location optimization problem, a genetic algorithm based solution is proposed. Accounting for the relevant constraints in the actual deployment process, the communication connections between the actuators, and the priority levels of the sensors, the sensor node is bound to the corresponding actuator node to the the maximum extent, and the communication coverage between the actuator node and the sensor node is improved.

2. LOCATION OPTIMIZATION MODEL

The main goal of the optimization problem is to find a set of minimum cross points. The point in the set is the position where the actuator node is to be placed. With the minimum of actuators, the maximum communication coverage can be achieved under the premise of satisfying the related constraints.

Divide a space into $M \times N$ cells of length H and width W, as shown in Fig. 1. The center position of each cell in Fig. 1 is taken as the candidate position of the sensor node, and the intersection of each dotted line is taken as the candidate node of the actuator node. In some special cases, in order to more accurately represent the location of the actuator node and the sensor node, the cell is divided into smaller cells according to the needs.

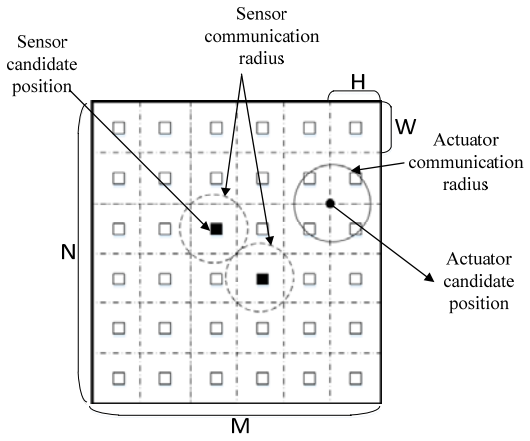


Fig. 1 Space partitioning

The location of the sensor is placed at a predetermined point according to the actual needs, and the relevant information set $\boldsymbol{C}$ of all the sensors is the input of actuator nodes deployment problem. Each element in $\boldsymbol{C}$ is the information set c_i of a sensor node, where $c_i =< S_i, P_i^s(x,y), SCR >, c_i \in \boldsymbol{C}$, S_i ($S_i \in \bar{\boldsymbol{S}}$, $\bar{\boldsymbol{S}}$ is the set of all preset sensor nodes) is the sensor node i , $P_i^s(x,y)$ parameter is the placement of the sensor S_i in the space, (x,y) is the coordinate index of the corresponding position, *SCR* parameter is the communication radius of the sensor.

Define $\boldsymbol{B}_k$ to be the information set of the candidate nodes under the current iteration step k , each element in the $\boldsymbol{B}_k$ is the information set b_j of a certain actuator node, where $b_j =< A_j, P_j^a(x,y), ACR >, b_j \in \boldsymbol{B}_k$, A_j ($A_j \in \bar{\boldsymbol{A}}$, $\bar{\boldsymbol{A}} = \{A_1, A_2, ..., A_k\}$ is a finite set of actuators that can be deployed at most) represents the actuator j . $P_j^a(x,y)$ is the placement of the actuator A_j in the space, (x,y) is the coordinate index of the corresponding position, *ACR* parameter is the communication radius of the actuator.

2.1 Fitness function

In [10,11], each sensor is treated equally when deploying an actuator node, the condition is not applicable if certain special sensor nodes need to be prioritized. In this paper, we give the sensor nodes different weight and pay more attention to the sensor nodes with high weight, so that they are constrained to a given actuator to meet the requirements of different scenarios.

The optimal deployment problem of the actuator node is transformed into a fitness function and 6 constraints. The effect of the fitness function is to evaluate the score of the relevant information set $\boldsymbol{B}_k$ of the candidate actuator node under the current iteration. In order to find a set of minimum actuator nodes satisfying constraints, the main factors to consider are:

(1) In the constraints of the premise, the deployment of the minimum number of actuator nodes;

(2) In the set $\bar{\boldsymbol{A}}$, constrain as many sensor nodes as possible to the actuator node and achieve maximum communication coverage;
(3) The adjacent actuators can communicate with each other;
(4) The actuator nodes are preferentially constrained to the sensor nodes with high weights.

The above factors are reflected in the following fitness function:

$$J_k = \boldsymbol{Z}_k \boldsymbol{R}_k \boldsymbol{I} + \boldsymbol{Z}_k (\boldsymbol{Q}_k + \boldsymbol{W}_k) \boldsymbol{Z}_k^T + Tr\ \boldsymbol{P}_k \tag{1}$$

where $\boldsymbol{Z}_k$ is a row vector of $1 \times n_s$ (n_s is the number of preset sensors) dimension and also is the set of actuators in the current k iteration; $\boldsymbol{R}_k$ is the reward weight matrix of the actuator nodes constrained to the sensor nodes, each element of $\boldsymbol{R}_k$ can be represented as $\boldsymbol{R}(i,j) = r_j \cdot \alpha_{ij}$, $r_j \in \boldsymbol{\Omega}$ is the reward weight of the sensor, the higher the sensor priority, the greater the reward weight; α_{ij} is the binding coefficient, can be expressed as:

$$\alpha_{ij} = \begin{cases} 1, & |P_i^s(x_1, y_1) - P_i^a(x_2, y3)| \le \min(SCR, ACR) \\ 0, & \text{others} \end{cases}$$

$\boldsymbol{Q}_k$ is the penalty weight matrix of the actuator, each additional actuator is penalized accordingly so as to deploy a minimum number of actuators; $\boldsymbol{I} = \{1,1,...,1\}^T$ is a $n_s \times 1$ dimensional column vector; $\boldsymbol{W}_k$ is the communication connection matrix between the actuators. If the actuators A_j and A_m can communicate with each other, that is $|P_j^a(\mathrm{x}_1, \mathrm{y}_1) - P_m^a(\mathrm{x}_2, \mathrm{y}_2)| \le ACR$, then $\boldsymbol{W}_k(j,m) = 1$, otherwise $\boldsymbol{W}_k(j,m) = 0$; $\boldsymbol{P}$ is the penalty weight matrix for unconstrained sensors, $p_j \in \boldsymbol{\Psi}$ is the penalty weight of the sensor, the higher the priority of the sensor is not bound, the greater the penalty weight, The elements in $\boldsymbol{P}$ can be expressed as:

$$\boldsymbol{P}_k(i,j) = \begin{cases} p_j \bullet \beta_{ij}, & i = j \\ 0 \quad , & \text{others} \end{cases}$$

where β_{ij} is the constraint coefficient. If sensor S_i is constrained to an actuator, $\beta_{ij} = 1$, otherwise 0.

2.2 Restrictions

This section will consider six constraints to ensure that the introduced method is more feasible in WSAN. Our optimization goal is to obtain a minimum set of intersection points to deploy the actuator. Since the space has been divided into multiple cells before the algorithm, the number of intersection points have been set up, assuming n, as the deployment of the implementation of the actuator node input. The first constraint is as follows:

$$1 \le N_g \le n/l, 1 \le l \le n_s \tag{2}$$

where N_g is the number of genes in the individual. The constraint (2) ensures that the number of nodes in the minimum cross point set does not exceed $1/l$ times the total number n of intersections.

During deployment, two or more actuator nodes are not allowed to be placed in the same position, so each placed actuator node will have a unique position.

$$P_j^a(x_1, y_1) \neq P_m^a(x_2, y_2), j \neq m, \forall j, m \in \Lambda_k \tag{3}$$

where Λ_k is the numbering set of actuator nodes in B_k. A sensor node may be required to be bound to only one actuator node or not be bound to any deployed actuator point, for which the specified sensor S_i is:

$$\sum_{j \in \Lambda_k} \beta_j \alpha_{ij} = \begin{cases} 0 \\ 1 \end{cases}, S_i \in \bar{S} \tag{4}$$

where β_j is the corresponding column element of Z_k .The constraint (4) ensures that the sensor S_i will only send information to the actuator A_j.

In the deployment, due to various factors, it is allowed that some sensor nodes are not constrained, at this point:

$$(n_s - \sum_{j \in \Lambda_k} \alpha_{ij}) \leq U_{\max} \tag{5}$$

Constraint (5) ensures that the unrestricted sensor node does not exceed a given threshold $U_{\max}$, which is an input to the actuator node deployment problem.

In order to balance the workload of each actuator node, we can use constraint (6) to limit, then for a given actuator A_j:

$$1 \leq \sum_{j \in \Lambda_k} \alpha_{ij} \leq h_{\max} \tag{6}$$

The constraint (6) ensures that the actuator A_j can constrain at least one sensor node, but does not exceed the maximum threshold $h_{\max}$ of the number of sensors that can be bound. $h_{\max}$ can also be used as input to a deployment problem, either as a constant throughout the optimization process, as a static restriction, or as a dynamic restriction in each genetic generation of the GA. The dynamic calculation formula is as follows:

$$h_{\max} = n_s \Big/ \sum_{j \in \Lambda_k} \beta_j \tag{7}$$

Constraint (7) ensures that the workload of each actuator node is balanced, it does not cause some actuator nodes to constrain too many sensor nodes. While some actuator nodes constrain too few sensor nodes, especially in dense topologies. Constraint (7) considerations are important when the preset sensor nodes are located closer together.

3. REALIZATION of GENETIC ALGORITHM

Genetic algorithm (GA) is a method to search the optimal solution by simulating the natural evolution process. The optimal deployment problem described in this paper is considered to be NP (Non-Deterministic Polynomial) problem[12]. The exhaustive method is clearly impractical, the GA can be used to search for the approximate solution of the optimal solution, that is, the approximate optimal solution.

The main steps of GA are as follows:

(1) Set the relevant information set $\boldsymbol{C}$ of the sensor node, actuators set $\bar{\boldsymbol{A}}$ that can be deployed at most, determine the solution space for the optimization problem;

(2) Initialize the constraint related parameter n, n_s, $U_{\max}$, $h_{\max}$, and whether to use the elite strategy flag;

(3) Initialize the GA parameters, set the number of iterations, crossover probability P_c, mutation probability P_v, population size, initial population, calculate the fitness of the individual in the population;

(4) It is judged whether the crossover condition is satisfied. If the condition is satisfied, then crossover, otherwise run Step 5 directly;

(5) Determine whether the mutation conditions meet, if so, then carry out genetic mutation, otherwise run Step 6 directly;

(6) Calculate individual genetic fitness, individual selection in tournament;

(7) Determine whether the stop criterion meets, if so, then output results, the algorithm ends; otherwise return to Step 4;

Algorithm flow shown in Figure 2.

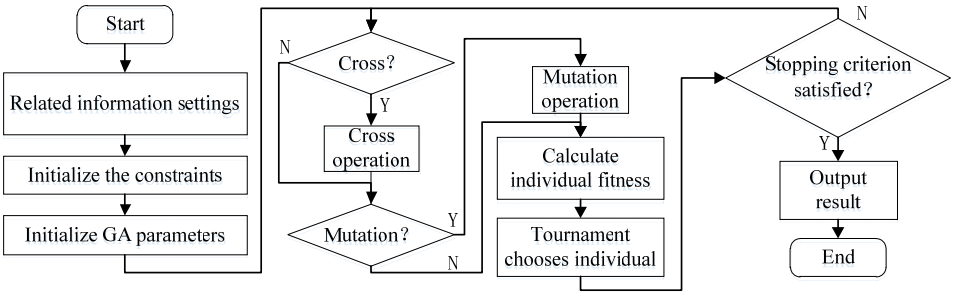


Fig. 2 GA flow chart

In the GA , N_p individuals are generally included. The individual is represented as $x_k = (x_{1,k}, x_{2,k}, ..., x_{N_g,k})$,where $N_g \le n_s$, k is the current gene generation. The GA consists of three main operations:

(1) crossover

The crossover operation mainly produces the experimental individual u_k , as follows:

$$u_{j,k} = \begin{cases} v_{j,k} & rand_j(0,1) \le P_c \text{ or } \mathrm{j} = \mathrm{j}_{\mathrm{rand}} \\ \mathrm{x}_{\mathrm{j,k}} & \text{others} \end{cases} \tag{8}$$

where $j = 1, 2, ..., N_p$; j_{rand} is a random integer from 1 to N_p ; $rand_j(0,1)$ is a random number $[0,1]$ uniformly distributed for each j . The parameter j_{rand} is used to ensure that the experimental individual u_k is different from the target individual x_k . The crossover probability P_c is usually in the range of 0.01 to 0.1.

(2) mutation

In the mutation operation, the target vector v_k is generated for the selected individuals. The mutation strategy is as follows:

$$\text{rand/1: } v_k = x_{r_1,k} + F \bullet (x_{r_2,k} - x_{r_3,k}) \tag{9}$$

$$\text{best/1: } v_k = x_{best,k} + F \bullet (x_{r_1,k} - x_{r_2,k}) \tag{10}$$

$$\text{current-to-best/1: } v_k = x_k + F \bullet (x_{best,k} - x_k) + F \bullet (x_{r_1,k} - x_{r_2,k}) \tag{11}$$

$$\text{best/2: } v_k = x_{best,k} + F \bullet (x_{r_1,k} - x_{r_2,k}) + F \bullet (x_{r_3,k} - x_{r_4,k}) \tag{12}$$

$$\text{rand/2: } v_k = x_{r_1,k} + F \bullet (x_{r_2,k} - x_{r_3,k}) + F \bullet (x_{r_4,k} - x_{r_5,k}) \tag{13}$$

where r_1, r_2, r_3, r_4, r_5 , are chosen randomly from the set $\{1, 2, ..., N_p\}$; $x_{best,k}$ is the best individual in the kth generation; the scaling factor F is a real number, $F \in [0,1]$.

(3) selection

The gene selection operator uses the tournament selection method and combines the elite strategy to ensure that the population fitness is continuously improved. Tournament selection method is more random, there is a greater random error, but there is a high probability that the optimal individual is selected and the worst individual is eliminated. In addition, in order to improve the efficiency of the algorithm, the memory search method is used, that is, the gene which appeared before is added into the memory structure. When the gene reappears, the relevant processing can be done to save time directly with no constraints.

The algorithm has two stopping criteria. First we determine whether the maximum fitness value and the average fitness value of the change are not large and tend to be stable. And then we determine whether the algorithm reaches the maximum number of iterations. As long as one of the two conditions is satisfied, the result is output.

4. SIMULATION EXPERIMENT AND RESULT ANALYSIS

The simulation platform uses Visual Studio 2015 and MATLAB, uses C # to realize the GA and solves the optimal deployment problem of the actuator node.The relevant data are then exported into MATLAB and analyzed.

First, plan the size of the space to be deployed nodes, cell size and other related parameters. These parameters are the inputs to solve the problem of actuator node deployment, as shown in Table 1.

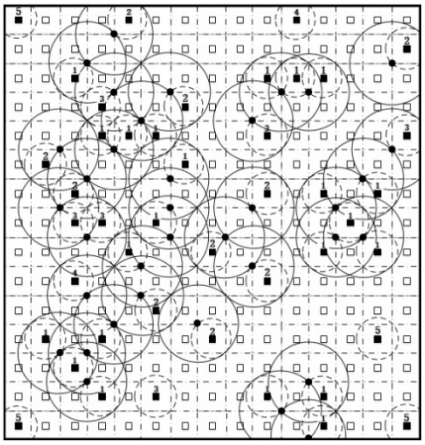
Table 1 Initialization of deployment related parameters

parameters	settings
size of space(m^2)	375×375
the cell size(m^2)	
set the number of sensors(n_s)	44
sensor communication radius(m)	18
actuator communication radius(m)	36
the number of unrestricted sensors allowed, $U_{\max}$	5
crossover probability P_c	0.9~0.96
mutation probability P_v	0.01~0.1
whether to use the elite strategy	yes

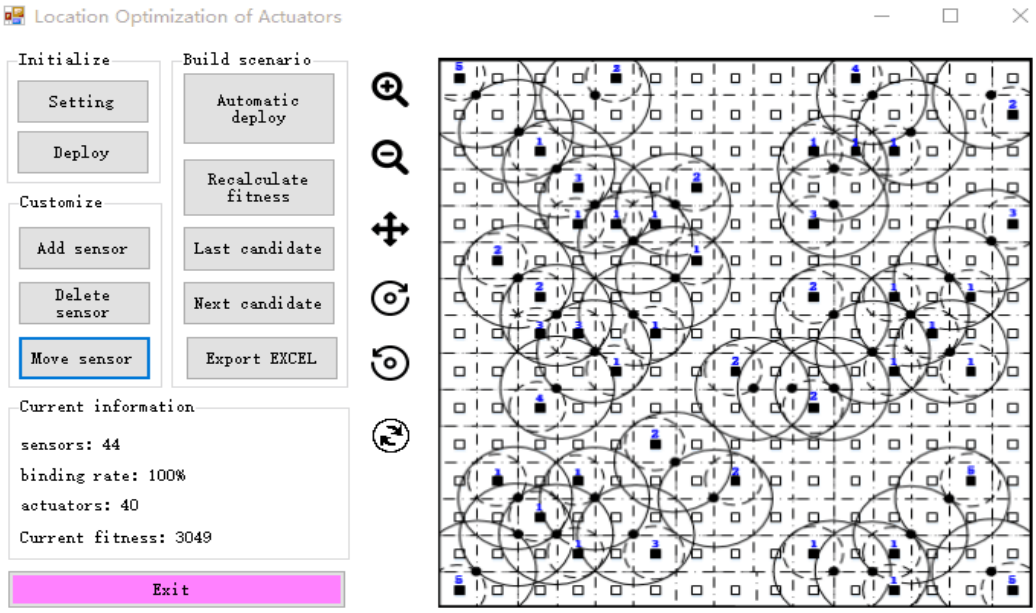
Under the parameters in Table 1, the number of genes in the simulation is $N_g \le 20$,that is $\bar{A} = \{A_1, A_2, ..., A_{20}\}$, decimal encoding is used. Penalty weight is $\boldsymbol{Q} = diag(-50, -50, ..., -50) \in \Re^{44\times44}$, $\boldsymbol{\Omega} = \{a, a-2, a-4, ...\}$, the elements in $\boldsymbol{\Omega}$ are decremented by 2, $a = 100$, up to 44 levels of weight are supported.In other more sensor topologies, the value of a can be increased. Similarly, $\boldsymbol{\Psi}$ can be set by $\boldsymbol{\Psi} = \{-a, -(a-2), -(a-4), ...\}$.

Figure 3 shows the deployment of the actuator node under the dynamic constraints when the crossover probability P_c is 0.9, the mutation probability P_v is 0.05, the initial population size $N_p = 2 * N_g$. In the implementation of the GA, the assigned actuator is preferentially bound to the sensor with the highest priority. In Fig. 3 (a), only the sensor of grade 5 is unbounded in the best individuals of the first generation. Figure 3 (b) is the visualization software interface, showing the optimal deployment location of the actuator node. Actuator node and sensor node

binding rate is 100%. With the visualization software developed, we can also add actuator nodes or move the automatically deployed actuator nodes to achieve more satisfying results. And the changed fitness can be recalculated to help the user decide whether to confirm the change. Finally, the program can be exported to EXCEL file, so that it can be easy to view in the actual deployment and data analysis in MATLAB.



(a) The optimal individual position of the first generation



(b) The visualization software interface shows the optimal position of the actuator

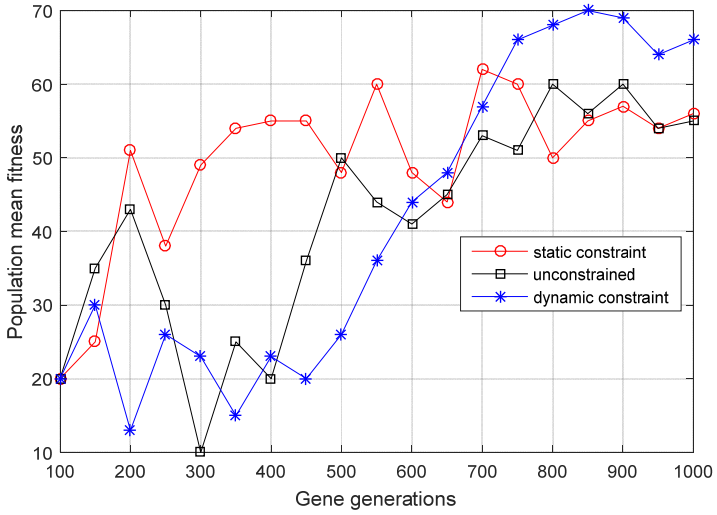
Fig. 3 Actuator node deployment results, in which the dashed circle indicates the sensor node communication range, the solid line circle indicates the automatically deployed actuator node communication range.

Table 2 Experimental comparison under different constraints

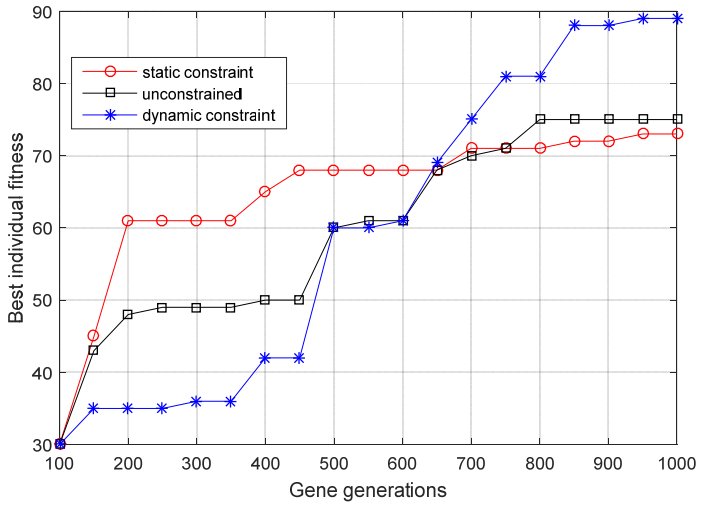
experiment	Static constraint	dynamic constraint	unconstrained
number of sensors	44	44	44
number of actuators	40	40	41
number of unrestricted sensors	6	0	0
number of constrained sensors	38	44	44
actuator and sensor binding rate	86%	100%	100%
running time (ms)	3015	2051	1813

Next, the binding rate and running time of the actuator nodes and the sensor nodes under different constraints will be compared with the initial parameters shown in Fig. 3 by controlling the constraints. The primary purpose of the experiment is to constrain the given sensor node to the actuator node and calculate the optimal position of the actuator node. In the experiment, the binding rate of the actuator and the sensor, and the running time of the algorithm are taken as the criterion of the performance of the algorithm. Table 2 lists the experimental results under three different constraints. The experiment mainly considers the static constraints (ie, $h_{\max}$ is set to constant), the dynamic constraints (dynamically calculated according to constraint (7)), and the unconstrained. Under static constraints, $h_{\max} = 2$ limits the binding of the sensor to an actuator, resulting in a binding rate of only 86%, where $h_{\max}$ is 2 in order to test some extreme cases of deployment. Under dynamic constraints, the actuator can self-adaptively balance the load and increase the binding rate up to 100%. The binding rate reaches 100% and the average running time is the shortest in the unconstrained condition. The main reason is that the time is saved in the constraint judgment of individual genes. The experimental results show that the proposed GA has strong adaptability to different constraints, which is due to the stochastic properties of GA.

Figure 4 shows the average fitness and the best individual fitness for each generation under different constraints. It can be seen from Figure 4 (a) that the average fitness of the population changes drastically. The reason for this situation is that, in order to avoid trapping into the local search, the mutation probability is set to be larger to get more possible individuals, so that the optimal solution is more likely to be obtained. An elite strategy is introduced to preserve the best individual in each generation so as to avoid the pure stochastic search. It can be seen from Fig. 4 (b) that the best individual fitness of each generation population is gradually increased. Under static constraints and dynamic constraints, when the number of iterations is greater than 800, the best individual fitness in the population has no obvious change in the offspring, and the number of iterations at this time can be regarded as the stopping condition. Similarly, the number of iterations of the algorithm can be set to 900 without constraints, which can be used to speed up the algorithm.



(a) Population mean fitness



(b) Best individual fitness

Fig. 4 Population fitness and individual fitness under different constraints

5. CONCLUSION

In this paper, the problem of deployment of actuator nodes is modeled and the optimal deployment is solved by genetic algorithm. We use the related technology to carry on the code realization and demonstrate the deployment result with the visualization form. Through the control of constraints, the binding rate and running time of actuator nodes and sensor nodes are compared under different constraints, and the efficiency of the improved GA is verified. In addition, we export the experimental data into MATLAB, and get the change trend of the average fitness of the population and the best individual fitness of each generation in the GA,

so as to optimize the number of iterations under different parameters, and further improve the running efficiency of GA.

REFERENCES

[1] I F Akyildiz, W Su, Y Sankarasubramaniam, et al, “Wireless sensor networks: a survey”, Applied Mechanics & Materials, 2002, Vol.38(4), p393-422

[2] C Raghavendra, K Sivalingam, T Znati, “Wireless sensor networks”, Kluwer Academic Pub, 2004

[3] J Yick, B Mukherjee, D Ghosal, “Wireless sensor network survey”, Computer Networks the International Journal of Computer & Telecommunications Networking, 2008, Vol.52(12), p2292-2330

[4] J Chen,Q Yu,P Cheng, et al, “Game theoretical approach for channel allocation in wireless sensor and actuator networks”, IEEE Transactions on Automatic Control, 2011, Vol.56(10), p2332-2344

[5] J Fan, J Chen, Y Du, et al, “Delque: A socially-aware delegation query scheme in delay tolerant networks”, IEEE Transactions on Vehicular Technology, 2011, Vol.60(5), p2181-2193

[6] L Shi, P Cheng, J Chen, “Optimal periodic sensor scheduling with limited resources”, IEEE Transactions on Automatic Control, 2011, Vol.56(9), p2190--2195

[7] K Akkaya, M Younis, “COLA: A Coverage and Latency Aware Actor Placement for Wireless Sensor and Actor Networks”, The Proceedings of IEEE Vehicular Technology Conference, 2006, p1 - 5

[8] K Akkaya, M Younis, “C2AP: Coverage-aware and Connectivity-constrained Actor Positioning in Wireless Sensor and Actor Networks”, Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa. IEEE, 2007, p281-288

[9] T Melodia, D Pompili, V C Gungor, et al, “A distributed coordination framework for wireless sensor and actor networks”, ACM Interational Symposium on Mobile Ad Hoc NETWORKING and Computing, MOBIHOC 2005, Urbana-Champaign, Il, Usa, May. 2005, p99-110

[10] D B Jourdan, O L De Weck, “Layout optimization for a wireless sensor network using a multi-objective genetic algorithm”, Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th. IEEE, 2004:2466 - 2470 Vol.5

[11] D B Jourdan, O L D Weck, “Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility”, Proceedings of SPIE - The International Society for Optical Engineering, 2004, Vol.5403

[12] E Niewiadomska-Szynkiewicz, M Marks, “Optimization Schemes For Wireless Sensor Network Localization”, International Journal of Applied Mathematics & Computer Science, 2009, Vol.19(2), p291-302

[13] A A Kannan, G Mao, B Vucetic, “Simulated Annealing based Wireless Sensor Network Localization with Flip Ambiguity Mitigation”, Journal of Computers, 2006, Vol.2(2), p1022 - 1026