PAPER

Modified t-Distribution Evolutionary Algorithm for Dynamic Deployment of Wireless Sensor Networks

Xiaolei LIU[†], *Nonmember*, Xiaosong ZHANG[††a)], *Member*, Yiqi JIANG[††], *and* Qingxin ZHU[†], *Nonmembers*

SUMMARY Optimizating the deployment of wireless sensor networks, which is one of the key issues in wireless sensor networks research, helps improve the coverage of the networks and the system reliability. In this paper, we propose an evolutionary algorithm based on modified t-distribution for the wireless sensor by introducing a deployment optimization operator and an intelligent allocation operator. A directed perturbation operator is applied to the algorithm to guide the evolution of the node deployment and to speed up the convergence. In addition, with a new geometric sensor detection model instead of the old probability model, the computing speed is increased by 20 times. The simulation results show that when this algorithm is utilized in the actual scene, it can get the minimum number of nodes and the optimal deployment quickly and effectively.Compared with the existing mainstream swarm intelligence algorithms, this method has satisfied the need for convergence speed and better coverage, which is closer to the theoretical coverage value.

key words: *t-distribution, evolutionary algorithm, wireless sensor networks*

1. Introduction

Wireless sensor networks (WSNs) are self-organizing networks consisted of a large number of tiny sensor nodes. They are widely used in many fields, such as environmental monitoring, target tracking, data collection, etc. [1]. However, how to give full play to their roles highly depends on the sensors' positions, known as the deployment of the networks [2].

The specific research of deployment is how to use the method of node deployment and routing to make the optimal allocation of resources in WSNs, and then improve the quality of monitoring, sensing and communication and other services, in the case that sensor network node energy and other resources are generally limited. Therefore, the coverage problem reflects the degree of the sensor network node to the designated monitoring area, and it is a basic index to measure the quality of the service of the sensor network. For example, in the use of forest fire in the sensor network, the key problem is how to monitor the fire in the shortest time. As the sensor network environment is usually unknown, sensor deployment is usually not carried out manually. All the sensors mentioned in this paper are uniformly moving.

At present, many scholars have conducted this research, and made many achievements in the wireless sensor networks deployment optimization by using the swarm intelligence algorithms [3]–[5]. Li et al. proposed a method [6] of improved particle swarm optimization to solve the sensor node deployment problem. This algorithm can quickly find an excellent solution, but it is easy to fall into extreme point and ending without the global optimal solution. Liao et al. consider the problem of sensor deployment to achieve complete coverage of the service region based on the ant colony optimization algorithm [7]. It has a good local search capability, but there is a problem that the solving speed is very slow. An approach based on an optimized artificial fish-swarm algorithm for wireless sensor networks deployment optimization scheme is proposed by Wang et al. [8]. However, from the simulation results we can see, with the increase of the number of iterations, the optimization results are not significantly improved. This algorithm tends to be premature. Liao et al. present a sensor deployment scheme [9] based on glowworm swarm optimization to enhance the coverage after an initial random deployment of the sensors. In some cases this algorithm can achieve satisfactory optimization results, however, its stability is poor. Affected by the initial value, the result will occasionally be terrible.

To solve the dynamic deployment problem for WSNs, a new approach which is based on the modified t-distribution evolutionary algorithm (MtDEA) is proposed. By introducing a directed perturbation operator, this approach can not only avoid plunging local extreme, but also accelerate the convergence rate. Using a new geometric sensor detection model, the computational complexity is reduced and the operational efficiency is greatly improved.

The rest of the paper is organized as follows. Section 2 describes the dynamic deployment problem of WSNs and proposes a new sensor detection model. The MtDEA algorithm is introduced in Sect. 3. We present and discussed the simulation experiments and results in Sect. 4. Section 5 concludes this paper and remarks the future work.

Manuscript received November 17, 2015.
Manuscript revised February 16, 2016.
Manuscript publicized March 22, 2016.
[†]The authors are with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, 611731 China.
[††]The authors are with the Big Data Research Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731 China.
a) E-mail: johnsonzxs@uestc.edu.cn
DOI: 10.1587/transinf.2015EDP7464

2. WSN Dynamic Deployment Problem and Sensor Detection Model

2.1 WSN Dynamic Deployment Problem

In the WSN dynamic deployment problem, people typically improve coverage by increasing the number of sensors. However, excessive number of sensor nodes will produce a large number of redundant nodes, causing data transmission conflict, affecting the network stability and wasting the resources. Thus, in the WSN dynamic deployment stage, the number of nodes and network coverage both need to be considered. Algorithm chosen must be able to calculate the optimal deployment using minimal number of sensors under the premise of meeting the specified coverage.

Suppose the same type of sensor is arranged, with the same communication distance and measuring radius. $S = \{s_1, s_2, \ldots, s_n\}$ is a group of wireless sensor set. The coordinates of wireless sensor node s_i are (x_i, y_i) and target area is a two-dimensional rectangular area. For any target point p_j at $\left(x_j, y_j\right)$, the euclidean distance between sensor s_i and target point p_j is shown in Eq. (1).

$$d\left(s_i, p_j\right) = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{1}$$

The detection probability of sensor s_i to target point p_j [10] is shown in Eq. (2).

$$P\left(p_j, s_i\right) = \begin{cases} 1, & d \le r - r_e \\ e^{\frac{-\lambda_1 \alpha_1^{\beta_1}}{\alpha_2^{\beta_2}}} + \lambda_2, & r - r_e \le d \le r + r_e \\ 0, & otherwise \end{cases} \tag{2}$$

where r is the detection radius, r_e is a detection uncertainty radius, λ_1,β_1,β_2 are parameters measuring detection probability, λ_2 is the disturbing effect, $\alpha_1 = r_e - r + d(s_i, p_j)$ and $\alpha_2 = r + r_e - d(s_i, p_j)$.

2.2 Traditional Sensor Detection Model

Consider the computational complexity when building a wireless sensor detection model, the probability of detection in Eq. (2) reduces to:

$$P\left(p_j, s_i\right) = \begin{cases} 1, & d(s_i, p_j) \le r \\ 0, & otherwise \end{cases} \tag{3}$$

The joint measuring probability of all sensor nodes simultaneously detecting the target point p is shown in Eq. (4).

$$P_{union}(S_{all}, p) = 1 - \prod_{i=n} (1 - P(p, s_i)) \tag{4}$$

The area to be measured is generally divided into $l * h$ grids in traditional sensor detection model. Then the cells are simplified as pixels, furthermore the coverage ratio of the WSN is defined as the ratio of the number of cells which can be detected and the total number of cells and shown in Eq. (5).

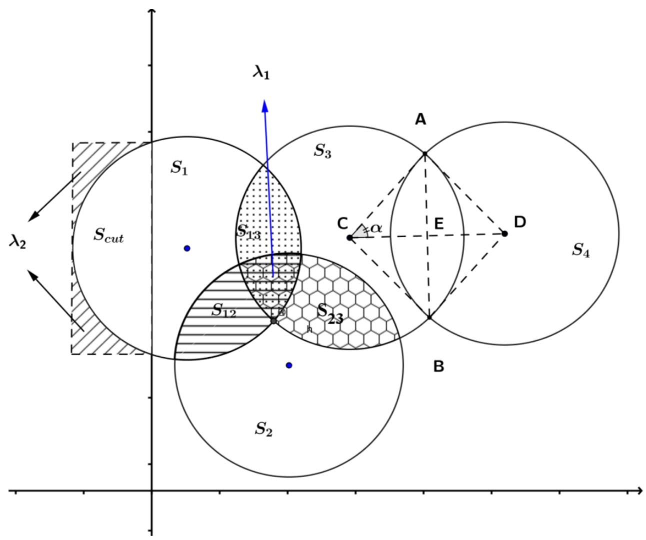


Fig. 1 The schematic diagram of sensor geometry model.

$$P_{area} = \frac{\sum_{x=1}^{l} \sum_{y=1}^{h} P_{union}(S_{all}, p)}{l * h} \tag{5}$$

2.3 Geometric Sensor Detection Model

The traditional sensor detection model has many flaws such as high computational complexity, long computational time and imprecise computational accuracy and so on. To avoid these problems, we propose a geometric sensor detection model. The covering area of N sensor nodes is shown in Eq. (6).

$$\begin{aligned} S_{nsensor} &= \sum_{i=1}^{n} S_i = \sum_{i=1}^{n} (S_{single} - S_{overlap}) \\ &= \sum_{i=1}^{n} (S_{single} - \sum_{j=i}^{n} S_{ij} + \lambda_1) \end{aligned} \tag{6}$$

where S_{single} is the coverage area of a single sensor, S_{ij} is the area of the overlapped portion between the i-th and j-th sensors, d is the distance of the center points between the i-th and j-th sensors. The overlapping coverage areas of more than three sensors are defined as λ_1. In the actual programming calculation, we just need to take λ_1 as a penalty term instead of calculating its specific value. In this way, a lot of sensors can avoid overlapping together.

Here, suppose we need to calculate S_{34} which is shown in Fig. 1.

$$\begin{aligned} S_{34} &= 2 \times (S_{sectorABC} - S_{\triangle ABC}) \\ &= 2 \times (0.5 \times r \times \overset{\smile}{AB} - 2 \times S_{\triangle ACE}) \end{aligned}$$

$$= 2 \times (0.5r \times \frac{2\alpha}{2\pi} \times 2\pi r - 2 \times \frac{1}{2} \times \frac{d}{2}\sqrt{r^2 - (\frac{d}{2})^2})$$
$$= 2r^2 \cdot \arccos\frac{d}{2r} - d \cdot \sqrt{r^2 - (\frac{d}{2})^2}$$

From S_{34} we can derive the formula of S_{ij} which is shown in Eq. (7).

$$S_{ij} = \begin{cases} \sum\limits_{j=i+1}^{n} (2r^2 \cdot \arccos\frac{d}{2r} - d \cdot l), d \leq 2r \\ 0, otherwise \end{cases} \tag{7}$$

where $l = \sqrt{r^2 - (\frac{d}{2})^2}$.

When using the N sensor nodes monitoring a specific area, the real effective coverage area $S_{realcover}$ equals to $S_{nsensor}$ minus S_{cut} which is the area outside monitoring area. And a penalty term λ_2 is introduced which is shown in Fig. 1.

Finally, the WSN coverage P_{area} is defined as the ratio of the real effective coverage area $S_{realcover}$ and the monitoring area $S_{monitor}$ and is shown in Eq. (8).

$$P_{area} = \frac{S_{realcover}}{S_{monitor}} = \frac{S_{nsensor} - S_{cut} - \lambda_2}{l * h} \tag{8}$$

2.4 Contrast Detection Model

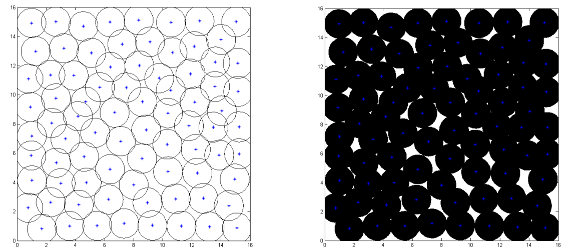
In order to compare the accuracy of the results between the traditional model and the new model, we carried out the following experiment.

1. Save arbitrary time WSN deployment optimization results in image format. The place which is covered is in black and the place uncovered is indicated with white.
2. Save the result of the corresponding sensor layout vector.
3. Respectively use traditional model and new model to calculate the vector to get the coverage.
4. Get the binary image which is obtained through making black and white into matrix 0-1. By calculating the percentage of matrix 0-1, obtain accurate sensor coverage.
5. Compare the results.

One of the WSN deployment optimization schematic diagrams is shown in Fig. 2 (a). Figure 2 (b) is the corresponding binary image and the place which is covered is black and the place uncovered indicated with white. The accurate WSN coverage, obtained by calculating the percentage of matrix 0-1, is 0.4523. 0.4738 and 0.4739 are obtained by the traditional sensor detection model and the geometric sensor detection model.

To further compare the running time and error range between the two sensor detection models, we randomly generated 500 independent deployment diagrams and the calculating results are shown in Table 1.

From the average obtained by 500 times independent deployment results, it is clearly that there is little difference



(a) A WSN deployment optimization schematic diagrams. (b) The corresponding binary image

Fig. 2 The schematic diagram of using binary image to calculate the accurate coverage. The place which is covered by sensors is black and the others is white.

Table 1 The calculating results of the two detection models.

	Traditional Sensor Detection Model(100*100)	Geometric Sensor Detection Model
Elapsed Time/ms (500 times)	848	40
Error Range/%	2.2149	1.9322

between the calculating error of the two detection models. However, the calculating speed varies widely. The speed of geometric sensor detection model is 20 times faster than the speed of traditional sensor detection model. The reason is that the algorithm complexity of traditional sensor detection model is $O(MN)$, where M is the number of grids, while the other one just $O(N^2)$. In most cases, M is larger than N and $O(N^2)$ is the theoretical maximum of our algorithm, it will never be reached.

3. Dynamic Deployment of WSNs with Modified t-Distribution Evolutionary Algorithm

3.1 Deployment Optimization Operator

The deployment optimization operator is responsible for obtaining the sensor deployment which makes the maximum coverage under the situation that the scope of the monitoring region, the number of sensor and the detection radius.

The steps of this operator are as follows.

1. Initialize the population P.
2. Apply the directed perturbation operator for P.
3. Apply the random jump operator for P.
4. Choose the next generation P' by roulette wheel method.
5. If the termination condition is satisfied then the algorithm ends. Otherwise, skip to step 2.

Among them, the most important part is directed perturbation operator, which is obtained by the modified t-distribution. It is known that the t-distribution can be considered as a mixture of normal distribution and gamma distribution, so when n is 1, this distribution reduces to the Cauchy distribution, when n tends to positive infinity, this distribu-

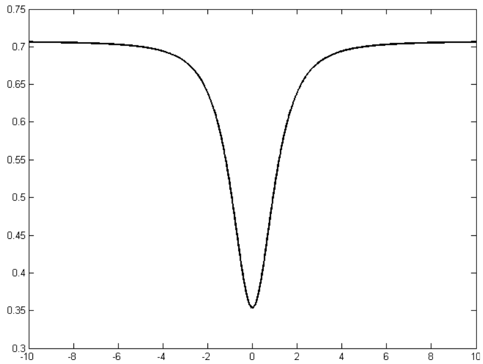


Fig. 3 The modified t-distribution

tion is Gaussian distribution.Therefore, the directed perturbation operator which is based on the modified t-distribution owns both of stability and freedom. It can be a good balance of linear and nonlinear behavior. The image of modified t-distribution is shown in Fig. 3 and the formula of directed perturbation operator is shown in Eq. (9). On the one hand, from Fig. 3 we can see if x is far away from y-axis, that is, the difference between the evolutionary individual and the best individual is large, MtDEA can make poor individual have a relatively faster speed of evolution. This shows the fast convergence at the beginning of the algorithm. On the other hand, when the difference between individuals is small, the heavy-tailed characteristic of the t-distribution is beginning to play a role. At the same time, with the random jump operator, MtDEA can jump out of local optimal value and continue to evolve, and finally get better results than other algorithms.

$$X_{i+1} = X_i + Z_i \tag{9}$$

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \tag{10}$$

$$Z_i = \{\lambda_{ij} \cdot z_{ij}\}, (j = 1, 2, \cdots, n) \tag{11}$$

$$\lambda_{ij} = step \times (x_{ij} - x_{best,j}) \tag{12}$$

$$z_{ij} = \frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi}\Gamma(\frac{n}{2})}[1 + \alpha \times \frac{(\frac{x_{ij}-x_{best,j}}{max-min})^2}{n}]^{-\frac{n+1}{2}} \tag{13}$$

where X_i is the position vector of sensors. If $x_{ij} = k$ means the i-th sensor coordinates of the j-th dimension is k. Z_i is the position vector delta and *step* is the unit perturbation step. If the sensor coordinates x_{ij} is beyond the monitoring area after being directed perturbation, the algorithm will recursively reduce the value of the *step* until the coordinates of new generation meet the requirements. *max* and *min* respectively represent the upper and lower bounds of coordinate x_{ij}. x_{best} represents the coordinate vector of the optimal fitness, and the direction of perturbation is judged by the positive and negative result of $(x_{ij} - x_{best,j})$.

The pseudo code of the directed perturbation operator is as follows.

After the directed perturbation, select some individuals to apply the random jump operator. The pseudo code of the random jump operator is as follows. By random jump, we can avoid the algorithm go into local extreme value and causing "premature".

Algorithm 1 Directed perturbation operator

Require: Population Size *pop_size*, Disturbance Probability dp, Lower bounds Lb, Upper bounds Ub

for $i = 0 \rightarrow pop\_size$ **do** //This loop is for iterating the population.
  if $rand < dp$ **then**
    repeat//If the progeny X_{i+1} doesn't meet the boundary requirements, then compute again.
      Compute:$X_{i+1} = X_i + \lambda_{dir} \oplus P_i'(x)$
    until boundarycheck(X_{i+1}, Lb, Ub)==suitable
  end if
end for

Algorithm 2 Random jump operator

Require: Population Size *pop_size*, Jump Probability jp, Lower bounds Lb, Upper bounds Ub

for $i = 0 \rightarrow pop\_size$ **do** //This loop is for iterating the population.
  if $rand < jp$ **then**
    repeat//If the progeny X_{i+1} doesn't meet the boundary requirements, then compute again.
      Compute:$jump = (rand(0, 1) - 0.5) * (Ub - Lb)$
      Compute:$X_{i+1} = X_i + jump$
    until boundarycheck(X_{i+1}, Lb, Ub)==suitable
  end if
end for

3.2 Intelligent Allocation Operator

The Intelligent allocation operator is based on the deployment optimization operator. By introducing the number perturbation operator, the minimum number of sensors and the corresponding deployment are solved in the case of the monitoring region range and the detection radius of the sensor.

The process of this intelligent allocation operator is shown in Fig. 4. First the algorithm is initialized according to the data of user input, and the chromosome length is obtained by the number perturbation operator, and then it is transformed into a single objective deployment optimization problem. When the single objective optimization operator is pulled out, the intelligent allocation operator can decide whether to satisfy the condition of the final exit, and then choose whether to continue to produce a better chromosome length.

The number perturbation operator is shown in Eq. (14).

$$Num_{i+1} = \begin{cases} maxN, & \frac{Cover(maxN)}{eCover} < 0.95 \\ minN, & \frac{Cover(minN)}{eCover} > 1 \\ pert(Num_i), & otherwise \end{cases} \tag{14}$$

$$pert(Num_i) = Num_i + k(maxN - minN) \times \frac{eCover - Cover(Num_i)}{1 + e^{-10\times\frac{Cover(Num_i)}{eCover}}} \tag{15}$$

where *maxN* is the maximum allowable number of sensors and *minN* is the minimum one. *eCover* indicates the

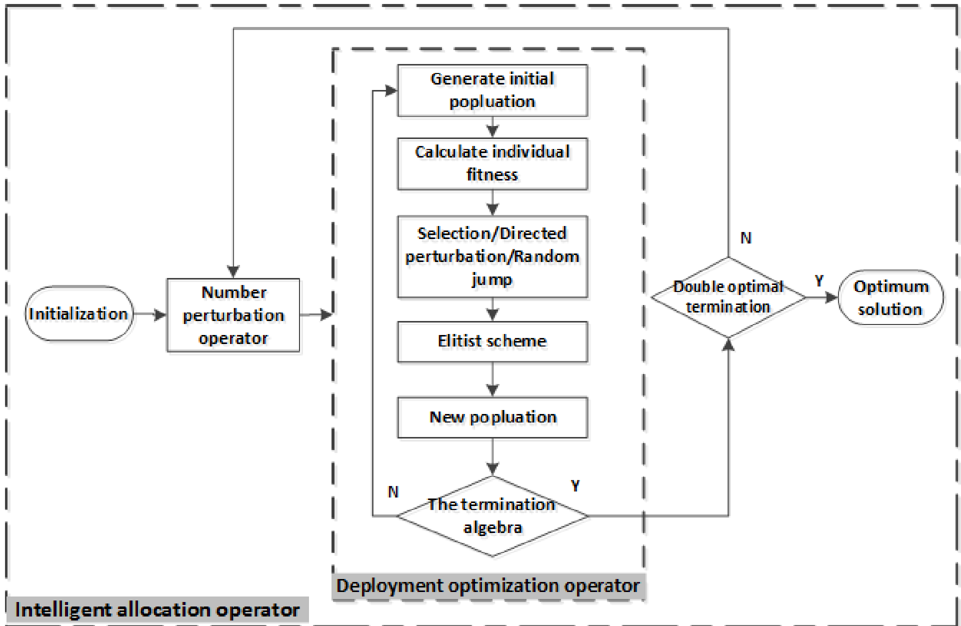


Fig. 4 Flow chart of intelligent allocation operator

expected coverage inputed by user. k is the step size adjustment factor of the number perturbation operator.

The exit condition of the intelligent allocation operator is as follows:

1. If the number of sensors is $maxN$, the $eCover$ is not satisfied, then exit.
2. If the number of sensors is $minN$, the $eCover$ is satisfied, then exit.
3. If the number of sensors which is Num_i is used to achieve the $eCover$ reached 98% to 102%, then exit.

4. Simulation Results

All the experiments are done under the Windows7-64 operating system, Core i3, 8G memory conditions, based on Java simulation.

4.1 Simulation I

A WSN including 80 mobile sensors is simulated. The detection radius of each sensor r is $1m$, the size of area is $256m^2$, the perturbation probability is 0.85 and the jump probability is 0.08. The best dynamic deployments obtained by MtDEA for each number of iterations are shown in Fig. 5.

4.2 Simulation II

A WSN including 80 mobile sensors is simulated and the size of area is $25600m^2$.

Comparison of Modified t-distribution evolutionary algorithm (MtDEA), Particle Swarm Optimization (PSO) [11], Cuckoo Search Algorithm (CS) [12], Artificial Bee Colony Algorithm (ABC) [13] and Firefly Algorithm (FA) [14] in the deployment optimization of WSNs are shown in follows.

Table 2 The parameters of five algorithms.

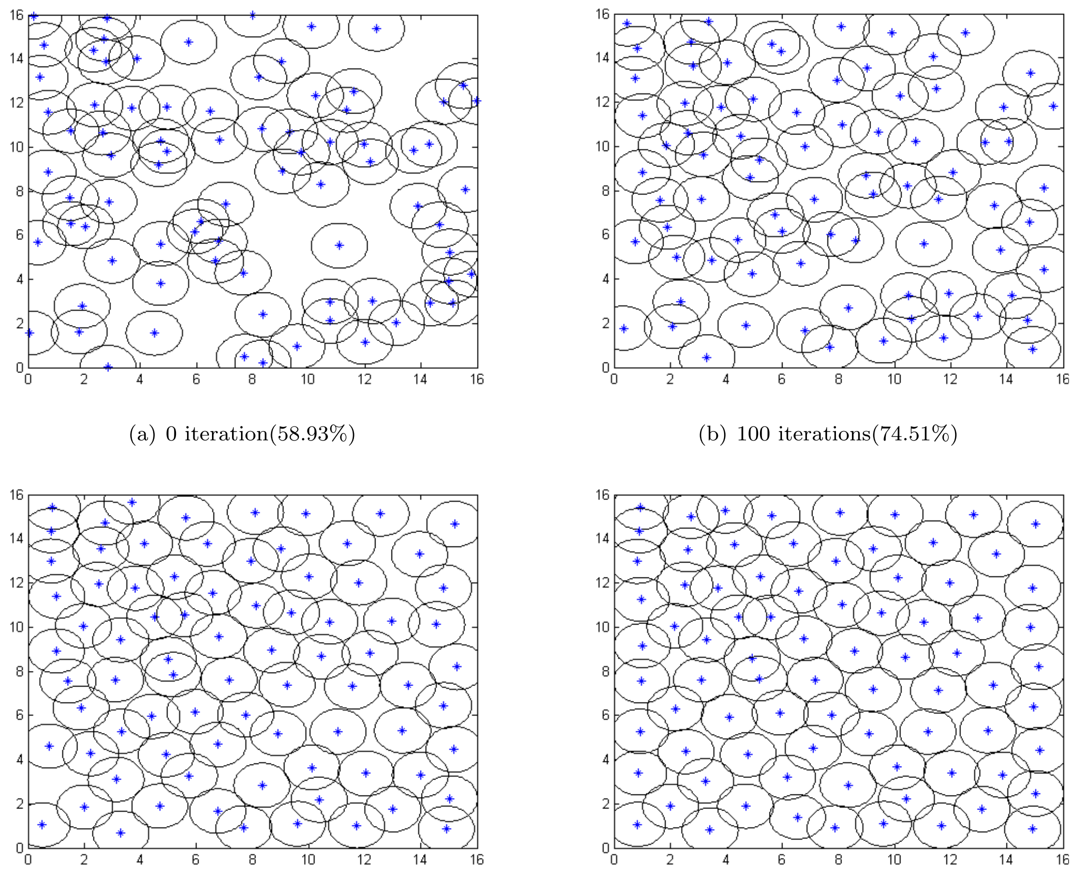
Algorithms	Parameters
MtDEA	Iterations:1000;Population:80;Perturbation Probability:0.85;Jump Probability:0.08
PSO	Iterations:1000;Particle Number:100;Maximum Moving Speed:2;ω:1.0;c_1:2;c_2:2
CS	Iterations:1000;Nest Number:25; p_a:0.25;α:1.0
ABC	Iterations:3000;Population:30;$limit$:50
FA	Iterations:1000;Population:50;γ:1.0;β_0:1;$step$:0.1;

Table 3 The simulation results for average operation time(s).

	MtDEA	PSO	CS	ABC	FA
$r = 9$	6.36	13.2	6.55	11.83	101.46
$r = 10$	8.76	15.17	7.49	14.63	135.32
$r = 11$	9.77	18.83	8.37	18.12	141.23
$r = 12$	10.91	20.87	9.26	20.76	156.51

We carried out 100 separate simulation experiments with five algorithms. The parameters of each algorithm are shown in Table 2. The simulation results for average coverage of each algorithm are shown in Fig. 6 and the average operation time are shown in Table 3. At the same time, we got the convergence curves of the 100 times of each algorithm, and the rate convergence curves of the algorithms are shown in Fig. 7.

The parameters shown in the Table 2 are either experience value or optimal value determined by conducting several preliminary experiments. For both PSO and FA, the parameters are experience value. For CS, the experimental results show that, $nestnumber$=25, p_a=0.25, this combination is able to satisfy most of the optimization problems [13]. For ABC, $limit$ is determined by experience value. Due to



(a) 0 iteration(58.93%)
(b) 100 iterations(74.51%)
(c) 300 iterations(83.14%)
(d) 600 iterations(86.65%)

Fig. 5 Best deployments obtained by MtDEA

the iteration time of each generation of ABC is far less than that of the other algorithms, it will be unfair if we compare it with other algorithms in the same number of iterations. After many experiments, when the number of iterations reaches 3000, ABC can get better results, and the overall computation time is close to that of other algorithms.

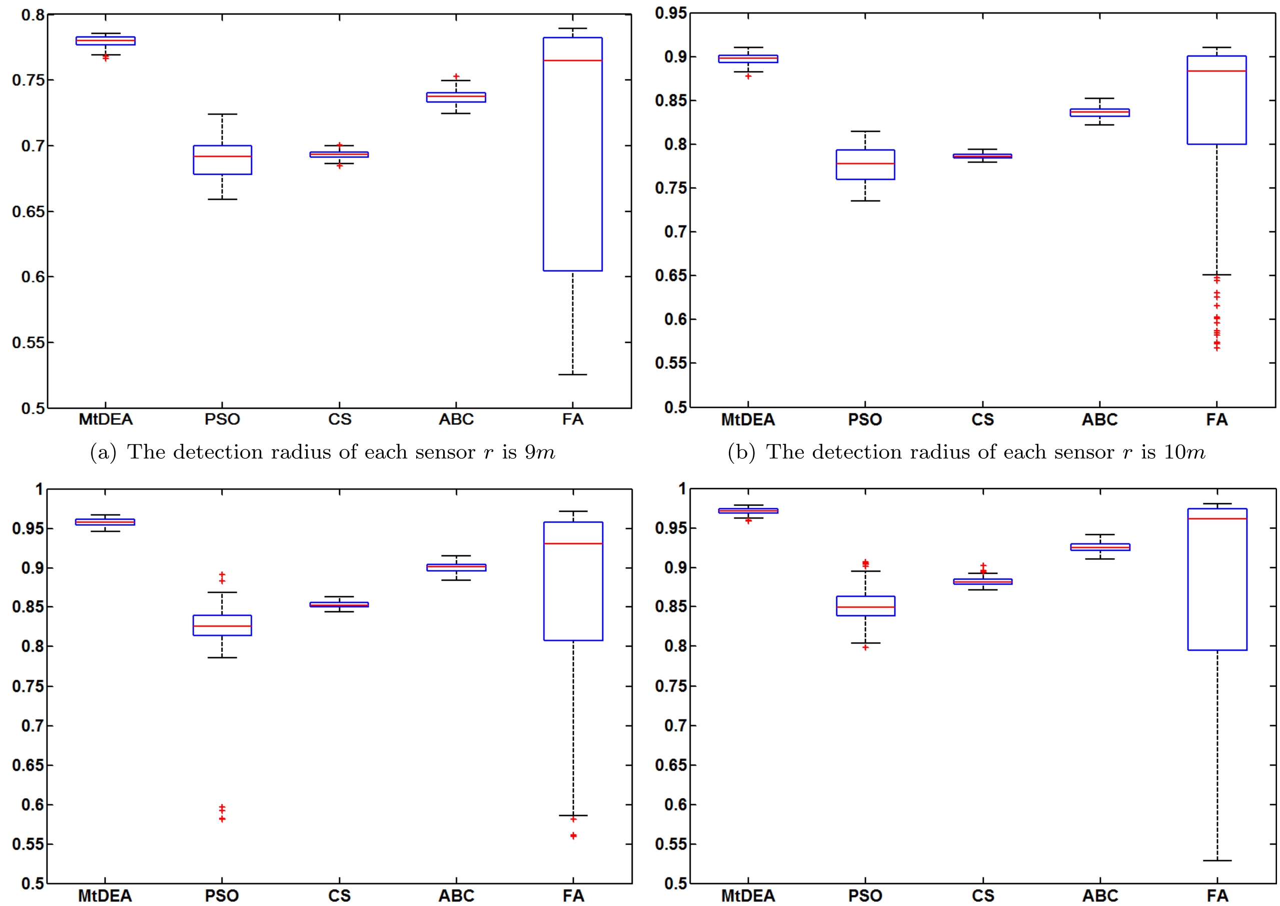
As seen in Fig. 6, MtDEA and FA can calculate out better deployment of WSNs than PSO, CS and ABC. However, the optimization effect of FA is very unstable. The 100 independent solutions of each radius show very strong instability. The optimization results are greatly influenced by the initial random solution, which is easy to fall into local optimum. By comparison, the MtDEA has good stability, and the results of the simulation experiments with different radius have little fluctuation. The optimization results are also good. This is in line with the characteristics of the proposed modified t-distribution evolutionary algorithm.

In comparison with the average running time, MtDEA and CS have great advantages. However, the optimization effect of CS is very poor. Due to the introduction of a number of adaptive parameters, MtDEA is slightly slower than CS, while compared with other algorithms, there are still a lot of advantages, especially for FA, which has better optimization result, and the speed is 15 times higher than it.

Table 4 The simulation results of intelligent allocation operator.

	Sensor number	Coverage(%)	Running time(s)
Intelligent allocation operator	Computational result=>21	90.69	7.17
Deployment optimization operator	Initial input=> 20	88.18	0.93
Deployment optimization operator	Initial input=> 21	90.77	0.96
Deployment optimization operator	Initial input=> 22	91.89	1.08

It can be seen from the fitting line chart in Fig. 7 that MtDEA with nearly 240 iterations has been completed from 55% to 85% of the coverage, the convergence rate is very fast, and the final optimization result which is best of these five algorithms is satisfactory. Benefit from the good distribution characteristics of t-distribution, MtDEA is able to evolve in a relatively large range of distance in the early stage of evolution.When the difference between individuals is small in the late evolution, the heavy-tailed characteristic of the t-distribution start to become effective, with the



(a) The detection radius of each sensor r is $9m$

(b) The detection radius of each sensor r is $10m$

(c) The detection radius of each sensor r is $11m$

(d) The detection radius of each sensor r is $12m$

Fig. 6 The simulation results for average coverage of each algorithm.

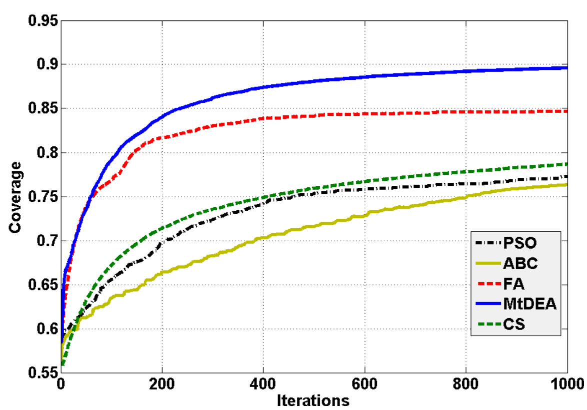


Fig. 7 The fitting line chart for average coverage of each algorithm.

random jump operator, MtDEA can jump out of local optimal and continue to evolve, and get better results than other algorithms.

4.3 Simulation III

A WSN including up to 40 at least 2 mobile sensors is simulated. The detection radius of each sensor r is $1m$, the size of area is $64m^2$, the expected coverage inputed is 90%, the perturbation probability is 0.85 and the jump probability is 0.08. The simulation results are shown in Table 4.

From the experimental results, we can see that the intelligent allocation operator can find the minimum number of sensors needed to meet the desired coverage in a short time, and can give the excellent deployment method of these sensors. The feature of the intelligent allocation operator, which greatly satisfies the actual needs of decision making, has a strong practical significance.

5. Conclusion

The deployment optimization of sensor nodes in WSN is beneficial to improve the coverage and reliability of the WSN. In this study, a modified t-distribution evolutionary algorithm is applied to the dynamic deployment problem in WSNs with mobile sensors. Simulation results show that MtDEA gives good deployment for WSNs and the geometric sensor detection model greatly improves the operation speed. Compared with some other common intelligent optimization algorithms, MtDEA is able to find the optimal solution in a very short period of time, it has faster convergence speed and more stable optimization effect. In future work, we plan to apply MtDEA for dynamic deployment of WSNs, including both mobile and stationary sensors.

Acknowledgments

This paper is supported in part by the National Natural Science Foundation of China under Grant 61572115, Grant 61502086, and Grant 61402080, in part by the Chinese Postdoctoral Science Foundation under Grant 2014M562307.

References

[1] X. Wang, S. Wang, and J.-J. Ma, "An improved particle filter for target tracking in sensor systems," Sensors, vol.7, no.1, pp.144–156, 2007.
[2] R. Mulligan and H.M. Ammari, "Coverage in wireless sensor networks: a survey," Network Protocols and Algorithms, vol.2, no.2, pp.27–53, 2010.
[3] M. Saleem, G.A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions," Information Sciences, vol.181, no.20, pp.4597–4624, 2011.
[4] R.V. Kulkarni and G.K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: A brief survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.41, no.2, pp.262–267, 2011.
[5] J. Chen, S. Cheng, Y. Chen, Y. Xie, and Y. Shi, "Enhanced Brain Storm Optimization Algorithm for Wireless Sensor Networks Deployment," Advances in Swarm and Computational Intelligence, Springer International Publishing, vol.9140, pp.373–381, 2015.
[6] Z. Li and L. Lei, "Sensor node deployment in wireless sensor networks based on improved particle swarm optimization," IEEE International Conference on Applied Superconductivity and Electromagnetic Devices, 2009, ASEMD 2009, pp.215–217, 2009.
[7] W.-H. Liao, Y. Kao, and R.-T. Wu, "Ant colony optimization based sensor deployment protocol for wireless sensor networks," Expert Systems with Applications, vol.38, no.6, pp.6599–6605, 2011.
[8] W. Yiyue, L. Hongmei, and H. Hengyang, "Wireless sensor network deployment using an optimized artificial fish swarm algorithm," IEEE 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE), vol.2, pp.90–94, 2012.
[9] W.-H. Liao, Y. Kao, and Y.-S. Li, "A sensor deployment approach using glowworm swarm optimization algorithm in wireless sensor networks," Expert Systems with Applications, vol.38, no.10, pp.12180–12188, 2011.
[10] S. Li, C. Xu, W. Pan, and Y. Pan, "Sensor deployment optimization for detecting maneuvering targets," IEEE 2005 8th International Conference on Information Fusion, vol.2, p.7, 2005.
[11] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," AIAA journal, vol.41, no.8, pp.1583–1589, 2003.
[12] X.S. Yang and S. Deb, "Cuckoo search via Levy flights," IEEE World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009, pp.210–214, 2009.
[13] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," Foundations of Fuzzy Logic and Soft Computing, pp.789–798, Springer Berlin Heidelberg, 2007.
[14] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," International Journal of Bio-Inspired Computation, vol.2, no.2, pp.78–84, 2010.

Xiaolei Liu received the B.S. degree in Software Engineering from the University of Electronic and Technology of China (UESTC), Chengdu, in 2014. He is currently a graduate student working towards the Ph.D. degree at UESTC. His research interest includes Multi-objective Optimization and Machine Learning.

Xiaosong Zhang received the B.S. degree in dynamics engineering from Shanghai Jiaotong University, Shanghai, in 1990 and the M.S. and Ph.D. degrees in computer science from the University of Electronic and Technology of China (UESTC), Chengdu, in 2011. Now he is a Professor in computer science at UESTC. He has worked on nu- merous projects in both research and development roles. These projects include device security, intrusion detection, malware analysis, software testing, and soft- ware verification. He has coauthored a number of research papers on computer security. His current research involves software reliability, software vulnerability discovering, software test case generation, and reverse engineering.

Yiqi Jiang was born in 1996. He is currently a graduate student working towards the B.S. degree at University of Electronic and Technology of China (UESTC), Chengdu. His research interest includes Multi-objective Optimization and Machine Learning.

Qingxin Zhu received his Ph.D. degree from Ottawa University, Canada, in 1993. In 1995, he did postgraduate research in Electronic Engi- neering Department of the Ottawa University and the Computer Department of the Carlton Uni- versity. He is currently the professor with the University of Electronic Science and Technology of China (UESTC). His major research interests include computer communication technology and net- work security, algorithm design, system stimula- tion, optimizing control and search.