EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Multi-objective optimized connectivity restoring of disjoint segments using mobile data collectors in wireless sensor network

Zhiping Kang[1,2*], Hong Zeng[3], Haibo Hu[2], Qingyu Xiong[4] and Guangyu Xu[4]

Abstract

Wireless sensor networks (WSNs) have been used extensively in a range of applications, which realizes data acquisition, processing, transmission, and analysis in an interesting area. Harsh surroundings and their inherent vulnerability often mean that these networks suffer from simultaneous node failure possibly causing the network to become partitioned into multiple disjointed segments. This in turn can prevent the gathering of data from the sensors and subsequent transmission to the sink, causing the whole network to fail. In this paper, a strategy is presented for restoring multi-objective optimization connectivity of these segments using mobile data collectors (MDCs), by considering the segments as collections of sensor nodes and not as some representative node. Different from existing uses of MDCs for restoration, the delay in data collection and task balance is considered, and the network connectivity and data acquisition path optimization problem are transformed into an improved multi-travelling salesman problem (iMTSP). An improved multi-objective optimization genetic algorithm for solving the optimal collection data collector position and moving paths is proposed, which introduces virtual segments and hierarchical chromosome structure, improved population diversity, and custom coding and decoding. The simulation results show that the proposed method can effectively solve the iMTSP of the Pareto optimal solution and can provide a new strategy for connectivity-restoring technology in WSNs. Compared with NSGA-*II*, the diversity of the proposed gene algorithm represents a clear improvement.

Keywords: Connectivity recovery, Multi-objective optimization, Mobile data collector, Wireless sensor network

1 Introduction

Wireless sensor networks (WSNs) consist of spatially distributed autonomous sensors that monitor certain events and phenomena cooperatively in an area of interest. They have been used extensively in a range of applications, including battlefield surveillance, industrial monitoring, environmental protection, and machine health monitoring. The sensors tend to be limited in size, and cost can result in corresponding constraints on resources such as energy, memory, computation, and communications. Limited resources and harsh environments often mean that sensors are prone to failure or damage. Not only do these failures cause a loss of coverage of the monitored area, but the network may also be split into several disjoint segments, affecting network connectivity and even leading to immeasurable losses in service quality. All this implies that the problem of connectivity restoration from disjoint segments in WSNs is an attractive field of research [1–5].

We propose the use of mobile data collectors (MDCs) to restore network connectivity, which have the function of data acquisition and are mobile, providing more computing, communications, and storage capacity than general sensor nodes. An MDC moves from a segment, collects the data, and then moves to the next segment with path optimization until all segments are traversed. Finally, all data are then transmitted to a base station by the MDC, allowing network connectivity to be restored indirectly. Differently from existing literature on MDCs, this paper considers a segment as a set of nodes in an attempt to find more optional nodes for the path of travel, rather

*Correspondence: kzp@cqu.edu.cn
[1]Key Laboratory of Ministry of Education for Dependable Service Computing in Cyber Physical Society, Chongqing University, Chongqing 400044, People's Republic of China
Full list of author information is available at the end of the article

than using a single representative node; the MDC does not move between adjacent segments, instead it travels all segments in the optimized order. In considering moving MDCs, this paper focuses on two main questions: (1) how to determine the optimal traversal paths, which would make the total visiting path of all MDCs to be the shortest and (2) how to maintain a balance between the moving tasks of different MDCs, such that the moving distance of all MDCs are approximately equivalent.

The main contributions of our proposed restoration strategy can be summarized as follows: (1) it takes account of the constituting nodes of a segment, the shortest moving path, and the moving task balance of MDCs, as such the problem is closer to a real application environment. (2) It translates the optimal path problem of MDCs into a travelling salesman problem and brings about an improved multiple travelling salesman problem known as iMTSP. (3) It introduces virtual segments and a hierarchical chromosome and puts forward an improved multi-objective optimization genetic algorithm to solve the optimal solution of iMTSP. The remainder of this paper is organized as follows. In Section 2, we summarize all the related work in this area. Section 3 describes the iMTSP problem, giving a formal definition. Section 4 presents our approach in detail. Performance evaluations are presented in Section 5, and our conclusions are drawn in Section 6.

2 Related work

2.1 Establishing k-connected topologies

The existing methods are attempts to provide fault tolerance by forming a k-connected topology, rather than focusing on repairing damaged networks [6–8]. In [9], a number of additional relays are deployed to ensure that each sensor node in the initial design has k length-bounded vertex-disjoint shortest paths to the sinks and uses Counting-Paths to minimize the number of deployed relays. Almasaeid et al. [10] tried to exploit the overlap between communication ranges to minimize the number of additional nodes needed to repair k-connectivity.

2.2 Relocating existing nodes

When a specified event occurs, some sensor nodes adjust their position under restriction conditions to restore connectivity [11–13]. In [14], a distributed actor recovery algorithm picks one of the neighbours of a failed node, in order to replace it. The selection of the best candidate is based on the lowest node degree and physical proximity to the failure. Alfadhly et al. [15] proposed a least distance movement recovery algorithm, in which a set of direct neighbours of the failed node move towards the position of the failed node while its original position is replaced with the nearest non-cut vertex actor.

2.3 Placing additional relay nodes

Some schemes employ relay nodes (RNs) to re-establish connectivity, which require additional nodes and may be applied to a large-scale network node failure [16–20]. Chen et al. [21] proposed an algorithm called MST-1tRNP, which constructs the complete graph of terminals and calculates the minimum spanning tree using Kruskal's algorithm. RNs are populated along the tree edges. In [22], the authors presented a three-step algorithm for federating network segments via a triangular Steiner tree approximation called FeSTA. Chen et al. [23] adopted a quadrilateral Steiner tree for finding the minimum relay nodes. Lee et al. [24] modelled the deployment area as a grid with equal-sized cells, proposing a distributed cell-based optimized relay node placement algorithm. Bio-inspired relay node placement heuristics were proposed in [25], where the segments are connected by deploying the RNs similar to a spider weaving its web.

In addition to these methods, Senel et al. [26] used MDCs to restore network connectivity, but the MDCs are only moved back and forth between different partitions to establish bidirectional connection and do not bring any directly acquired data back to the sink node. Different from the existing methods, we first focus on using MDCs to traverse all segments rather than just two adjacent segments, under the conditions that minimize the mobile total distance and ensure that different MDCs have moving-distance equilibrium. On the other hand, we consider the segments as a collection of sensor nodes, which sets them apart from most existing methods.

3 Network model and fundamental definition

3.1 Problem description

Where WSNs have become seriously damaged, a network may be split into different disjoined segments. In this paper, we take MDCs as travelling salesmen, touring segments and collecting data from the segments they visit in order to restore the connectivity of damaged WSNs. As shown in Fig. 1, there are eight segments in a damaged WSN, and only segment S_1 can transfer data back to the user through the Internet. We use MDCs to travel around all segments and transfer the data from segments $S_2, S_3, S_4, S_5, S_6, S_7$, and S_8 to segment S_1 to restore the connectivity of the entire network indirectly. During the process, firstly, the moving distance of each MDC should be as small as possible to reduce moving costs. Secondly, if only one MDC is used, its total movement path will be longer, increasing the data acquisition delay time, which is inappropriate for certain applications. We therefore use multiple MDCs to collect data simultaneously in order to ensure a data acquisition delay within an appropriate range.

The problem is similar to the multi-travelling salesman problem (MTSP), but it also has some key differences

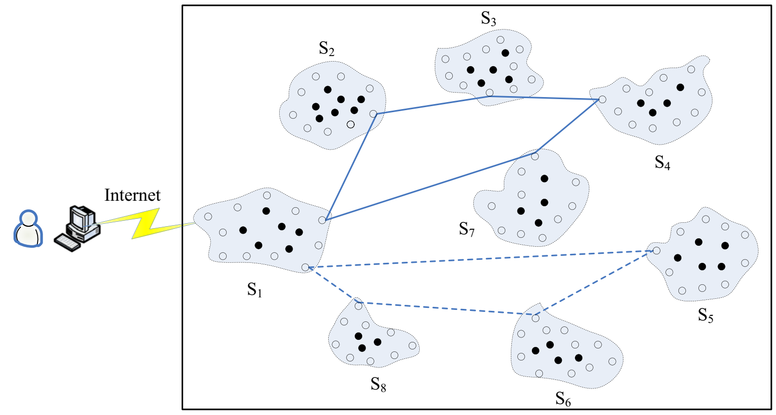

Fig. 1 Principle of connectivity restoring from disjointed WSN by MDCs

as follows: (1) any travelling city (segment) is no longer regarded as an abstract point (data acquisition position) but is a set of points. (2) Different travelling salesmen (MDCs) set out and return to the same city, which is the source city, but may in fact correspond to different points in the city. (3) The source city needs to be specified, and the starting and ending points are the same for the same travelling salesman but may be different for different travelling salesmen. As shown in Fig. 1, the moving path of one MDC is $S_1 - S_2 - S_3 - S_4 - S_7 - S_1$ and $S_1 - S_8 - S_6 - S_5 - S_1$ for another. Although both start and end with S_1, they correspond to different data acquisition positions of S_1. In solving our problem, the MDC has its own unique optimal data acquisition position in each segment, but this position is not the common representative point of a segment as it is in other work. In light of the above, we term the above problem the improved multi-travelling salesman problem (iMTSP). It is clear that iMTSP is a non-deterministic polynomial (NP) problem. If every city is abstracted into a single point, iMTSP is degenerated into a common MTSP.

3.2 iMTSP model

Let us assume that the sensor nodes of the damaged WSN are clustered into n disjointed segments: $S_1, S_2, \ldots, S_n$. The source segment S_1 is the starting and ending segment for MDCs, and only this can transmit the collected data to the user through the base station. $S_i = \{s_{iv}\}, i = 1, 2, \ldots, n, v = 1, 2, \ldots, ni, ni$ is the number of sensor nodes in S_i. Given the intervals of returning data acquisition, m MDCs are required to restore connectivity.

Let the variable $x_{S_{ijk}}$ be:

$$x_{S_{ijk}} = \begin{cases} 1, & \text{The MDC-k moves the path from} \\ & \text{segment } S_i \text{ to } S_j \\ 0, & \text{Otherwise} \end{cases} \tag{1}$$

The variable $y_{S_{ki}}$ is:

$$y_{S_{ki}} = \begin{cases} 1, & \text{The MDC-k visits segment } S_i \\ 0, & \text{Otherwise} \end{cases} \tag{2}$$

$c_{S_{ij}}$ is the moving distance of MDC from segment S_i to S_j:

$$c_{S_{ij}} = \{|s_{iv} - s_{ju}|\}, v = 1, 2, \ldots, ni, u = 1, 2, \ldots, nj \tag{3}$$

Usually in MTSP, $ni = nj = 1$, and $c_{S_{ij}}$ is a unique value. However, in iMTSP, $c_{S_{ij}}$ represents the set of $ni * nj$ Euclidean distances between two arbitrary points, which come from the corresponding segments S_i and S_j, respectively. The iMTSP model can be formalized as follows:

Set the objective function f_1 as:

$$f_1(Z) = min\left(\sum_{k=1}^{N} z_k\right) \tag{4}$$

where

$$z_k = \sum_{S_i=S_1}^{S_n} \sum_{S_j=S_1}^{S_n} c_{S_{ij}} x_{S_{ijk}}, k = 1, 2, \ldots, m \tag{5}$$

The constraint conditions are:

$$\sum_{k=1}^{m} y_{S_{ki}} = \begin{cases} m, & S_i = S_1 \\ 0, & S_i = S_2, , \ldots, S_n \end{cases} \tag{6}$$

$$\sum_{S_i=S_1}^{S_n} x_{S_{ijk}} = y_{S_{kj}}, j = 1, 2, \ldots, n; k = 1, 2, \ldots, m. \tag{7}$$

$$\sum_{S_j=S_1}^{S_n} x_{S_{ijk}} = y_{S_{ki}}, i = 1, 2, \ldots, n; k = 1, 2, \ldots, m. \tag{8}$$

$$X = (x_{S_{ijk}}) \in S^*, S^* \text{ is the branch elimination constraint.} \tag{9}$$

In this model, Formula (4) represents the minimum value of the total moving distances for m MDCs. Formula (5) represents the moving distance of all the paths traversed by each MDC. Formula (6) yields the total moving distances of MDC-k, which comes from the source partition and passes through each segment only once, before returning to the source partition. Formulas (7) and (8) constrain the source and ending segment of each MDC to be the same segment. Formula (9) is used to eliminate incomplete paths, which is detailed described as the Limitations in Section 4.2.

In order to maintain the moving equilibrium of each MDC, we introduce the objective function f_2:

$$f_2(Z) = min\{max(z_1, z_2, \ldots, z_m) - min(z_1, z_2, \ldots, z_m)\} \tag{10}$$

The essence of the iMTSP model is a multi-objective optimization MTSP. Our purpose is to find the Pareto optimal solutions of multi-objective Formulas (4) and (10), under the given MDCs and constraint conditions (6)-(9). In other words, Formula (4) is a sum minimization to ensure that the total distance travelled is minimized, while Formula (10) is a range minimization to ensure the moving task balance of different MDCs.

3.3 Multi-objective optimization

Let $Z_i = (z_1, z_2, \ldots, z_m)_i$ be the feasible solution, and $Z = (Z_1, Z_2, \ldots, Z_w)$ is the solution space of the multi-objective problem iMTSP. The relevant definitions are listed as follows:

Definition 1 (*Pareto dominance*) Z_1 and Z_2 are two arbitrary feasible solutions of iMTSP. Solution Z_1 dominates (or is better than) Z_2 if and only if $\{\forall i = 1, 2, \ldots, n, f_i(Z_1) \le f_i(Z_2)\} \wedge \{\exists j = 1, 2, \ldots, n, f_j(Z_1) < f_j(Z_2)\}$; this is denoted $Z_1 \succ Z_2$.

Definition 2 (*Pareto optimal solution*) The Pareto optimal solution (or non-dominated solution) is the solution that is not dominated by any solution among the set of feasible solutions. The solution $Z^* \in Z$ is the Pareto optimal solution if and only if there is no $Z_i \in Z, i = 1, 2, \ldots, w$ to $Z_i \succ Z^*$.

Definition 3 (*The set of Pareto optimal solutions*) The set of Pareto optimal solutions (or non-dominated solutions) is a collection of all Pareto optimal solutions.

Definition 4 (*Pareto optimal front*) The Pareto optimal front *PF* is the region of all Pareto optimal solutions corresponding to the objective function value, which is denoted as $PF = \{f(Z) = (f_1(Z), f_2(Z), \ldots, f_n(Z)) | Z \in Z^*\}$.

4 Recovery strategy using MDCs

The gist of our proposed algorithm lies in converting the connection recovery problem into iMTSP, by introducing virtual segments and hierarchical chromosomes and then adopting a multi-objective optimization genetic algorithm to solve the optimal solutions of iMTSP. Firstly, it identifies the independent segments using a fuzzy C-mean clustering algorithm (FCMA) in a damaged network, in which internal nodes in the same cluster can communicate with each other. Secondly, it uses MDCs to travel among all segments, which then carry collected data back to the sink. The optimal path problem of the MDCs is transformed into our proposed iMTSP. Thirdly, the problem is solved by an improved multi-objective optimization genetic algorithm based on NSGA-*II* [27], which introduces hierarchical chromosome structures, improved population initialization, and self-defined genetic operating and optimized crowding distance calculation, allowing the optimal moving paths of different MDCs to be obtained.

4.1 Segment identification by clustering

For an unknown deployment, we would use inspection equipment to detect the physical location of the existing active sensor nodes, together with their location information. Considering the limited resources of the nodes, a clustering algorithm is used in the base station or the client. In this paper, the FCMA is used to undertake clustering analysis of the sensor nodes. We assume that the coordinates of the sensor nodes in the network is $X = \{x_1, x_2, \ldots, x_n\}$. $U = [u_{ij}]$ is the fuzzy C-class matrix of the data set X, in which u_{ij} is the membership function of the j^{th} data point for the i^{th} class. The total membership of the j^{th} data point for C-class satisfies the following conditions:

$$\begin{gathered} \forall i, j, u_{ij} \in [0, 1]; \\ \forall \sum_{j=1}^{n} u_{ij} > 0, i = 1, 2, \ldots, c; j = 1, 2, \ldots, n; \\ \forall j, \sum_{i=1}^{e} u_{ij} = 1 \end{gathered} \tag{11}$$

The clustering criterion is the minimization of the clustering loss function $J_m(U, V)$:

$$J_m(U, V) = \sum_{j=1}^{n} \sum_{i=1}^{e} u_{ij}^m d_{ij}^2(x_j, v_i) \tag{12}$$

where V is the clustering centre, m is the weighted index, and $d_{ij}(x_j, v_i) = \|v_i - x_j\|$. After it is initialized, the algorithm is iterated continuously by Formulas (11) and (12), until converging to the minimum solution required realizing the fuzzy clustering.

4.2 Problem transfer, encoding, and decoding

A genetic algorithm (GA) is a type of parallel processing algorithm used to simulate biological evolution theory, which is an effective method for solving the multi-objective optimization problem. The solution process used in a genetic algorithm generally follows the following sequence: (1) conversion of the real problem into genetic sequences by encoding; (2) repeated execution of the basic genetic operations of selection, crossover, and mutation, producing new outstanding individuals, until the requirements are met; and (3) arrival at an approximate solution of practical problem according to decoding rules. Compared with a single-objective GA, a multi-objective GA must adopt special rules to evaluate the individual fitness and sorting.

To satisfy the requirements of problem transfer, we introduce virtual segments and hierarchical chromosomes into the iMTSP. The virtual segments transform the multi-travelling salesman problem into single travelling salesman problem with $N + m - 1$ virtual segments (N is the number of segments, m is the number of MDCs). The virtual segment in iMTSP is different from the "virtual city" in MTSP, which is a set of virtual points in the whole segment, rather than a single representative point. The introduced hierarchical chromosome consists of two level chromosome structures: segment gene and node gene. The segment gene represents the travelled sequence of different segments, and the node gene indicates the visited internal sensor nodes of the corresponding segment. Both are coded using integer numbers, in which the rules and limitations of encoding and decoding are as follows:

Rule 1 Each segment in the network is uniquely identified by SegmentID, which is an integer number from 1 to N (N is the number of segments). Similarly, each node in each segment is uniquely identified by NodeID from 1 to n (n is the number of nodes in the segment). Thus, each node in the network can be uniquely identified by "SegmentID+NodeID".

Rule 2 The corresponding relationship between the segment gene and the node gene lies in the gene position of the node gene and is not a simple one-to-one mapping.

Limitation 1 Each path of the MDCs starts from the source segment. The first gene position of the segment gene is always the source segment, and this does not participate in the subsequent crossover and mutation operation.

Limitation 2 Set $c_{S_{11}} = M$, where M is a positive infinite value and S_1 is the source segment.

In decoding, the segment gene is visited one-by-one from the first gene position to the end position. The first gene position represents the source segment of the first MDC. From the second gene position, the decoding gene position is mapped to the source segment, which indicates the end of the moving path of the previous MDC and the beginning of the moving path of the next MDC. Different from the segment gene, the i^{th} gene position of the node gene stores the traversed NodeID of the i^{th} segment by a MDC.

As an example, let us assume that the number of MDCs is 3, denoted as MDC-1, MDC-2, and MDC-3. There are seven segments: $S_1, S_2, S_3, S_4, S_5, S_6$, and S_7, and the number of sensor nodes in each segment is 10 to 20. S_1 is the source segment, and S_8 and S_9 are the introduced virtual segments, all of which have the same internal nodes. Using a hierarchical chromosome structure, the feasible solution can be described as shown in Fig. 2.

As indicated in Fig. 2, MDC-1 starts from S_1, moves through S_7 and S_4, and returns to S_1 (S_1 and S_8 are the same segments). The visited node NodeID=7 of S_1 comes from the 1^{st} gene position of the node gene, and NodeID=12 of S_7 comes from the 7^{th} gene position of the node gene. Correspondingly, the visited nodes of S_4 and S_8 are NodeID=17 and NodeID=18. The moving path of MDC-1 can be expressed as 1(7)-7(12)-4(17)-1(7). Similarly, the moving path of MDC-2 is 8(18)-6(5)-2(15)-8(18), and the moving path of MDC-3 is 9(3)-5(10)-3(9)-9(3). It is worth noting that Limitation 1 and Limitation 2 could ensure that the encoding is effective and feasible. If we do not limit the chromosome gene encoding, the segment gene may cause the following two cases to arise:

1. The source segment and virtual segments could be distributed in arbitrary gene positions apart from the first gene position. As shown in Fig. 3, there are three MDCs and four closed-loop moving paths in the encoding structure, which would be difficult to decode.

2. The multiple virtual segments could continuously appear in the same chromosome. As shown in Fig. 4, the possible paths of the three MDCs are $S_8 - S_2 - S_5 - S_4 - S_8$, $S_9 - S_9$, and $S_1 - S_2 - S_7 - S_3 - S_1$. The second path is invalid, starting from S_9 and returning to S_9. If we set $c_{S_{11}} = M$ (M is a positive infinity), this kind of chromosome will be eliminated in the population selection of a GA.

4.3 Improved population initialization

Different from NSGA-*II*, the parallel selection method is adopted for random population optimization to improve the convergence speed of a GA. The specific strategies are as follows: (1) a population with M individuals is randomly generated for iMTSP and (2) the M individuals are ordered by different objectives and the best m individuals of each objective function are preserved. If the number of individuals is not achieved, the best $m + 1$ individuals are preserved until M individuals are generated. It is worth

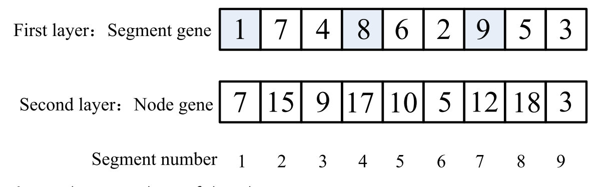

Fig. 2 The encoding of the chromosome structure

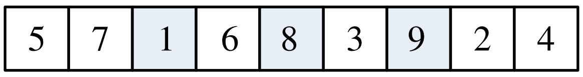

Fig. 3 The first gene position is not the source segment in the segment gene structure

noting that duplicate individuals are deleted in the preservation of optimal individuals. The new population is the initial population of the subsequent genetic algorithm.

4.4 Fast non-dominated sort

This method uses the same non-dominated sort as NSGA-*II*. Each individual $I(i)$ has two parameters, n_i and S_i, where the domination count n_i is the number of individuals that dominate $I(i)$ and S_i contains all the individuals being dominated by $I(i)$. In the outline, the sorting is as follows: Firstly, the individuals with $n_i = 0$ are denoted as the first non-dominated front F_1, which are removed from the population, and the rank of these individuals is set to $I(i)_{rank} = 1$. Then, we continue to look for the non-dominated solution set from the remaining population and denote them as the second non-dominated front F_2; their individuals are ranked as $I(i)_{rank} = 2$. The above procedure is continued using the remaining population, and the third front is identified. This process continues until all fronts are identified. The computational complexity of the sorting is $O(mN^2)$, in which m is the number of objectives and N is the population size.

4.5 Improved crowding distance

Once the non-dominated sort is complete, the crowding distance is assigned. Because the individuals are selected based on rank and crowding distance, all the individuals in the population are assigned a crowding distance value. The crowing distance of NSGA-*II* is calculated as:

$$I(d_i) = \sum_{m=1}^{M} \frac{I(i+1).m - I(i-1).m}{f_m^{max} - f_m^{min}}, \qquad (13)$$
$$i = 2, 3, \ldots, n-1; I(d_1) = I(d_n) = \infty$$

$I(d_i)$ is the crowding distance of individual $I(i)$, and $I(i).m$ is the value of the m^{th} objective function of the i^{th} individual. f_m^{max} and f_m^{min} are the maximum and minimum of the m^{th} objective function of individuals in front F_i. M is the number of objectives. The essence of the crowing distance is to find the Euclidean distance between each individual in a front based on m objectives in m dimensional hyper space.

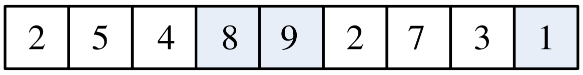

Fig. 4 Successive multiple virtual segments in segment gene structure

However, there are some limitations in calculating the crowding distance in NSGA-*II* as shown in Fig. 5: (1) Individual $I(a_1)$ and $I(b_1)$ are close to each other and are far from other individuals. By contrast, individual $I(d_1)$ and $I(e_1)$ are more dispersed. Using the calculation in NSGA-*II*, the crowding distances of $I(a_1)$ and $I(b_1)$ are greater than $I(d_1)$ and $I(e_1)$. In gene selection, individuals $I(a_1)$ and $I(b_1)$ are retained simultaneously, and individuals $I(d_1)$ and $I(e_1)$ are eliminated. In fact, the ideal selection is that one of $I(a_1)$ and $I(b_1)$ is removed and both of $I(d_1)$ and $I(e_1)$ are retained. (2) Individuals $I(b_2)$ and $I(d_2)$ have the same (or similar) crowding distances, and they would have the same (or similar) genetic probabilities in the selection. However, the ideal situation is that the selection probability of $I(d_2)$ is greater than that of $I(b_2)$, because the uniform distribution of $I(d_2)$ is better than that of $I(b_2)$.

In order to solve these problems, we propose an improved calculation strategy: firstly, we introduce the

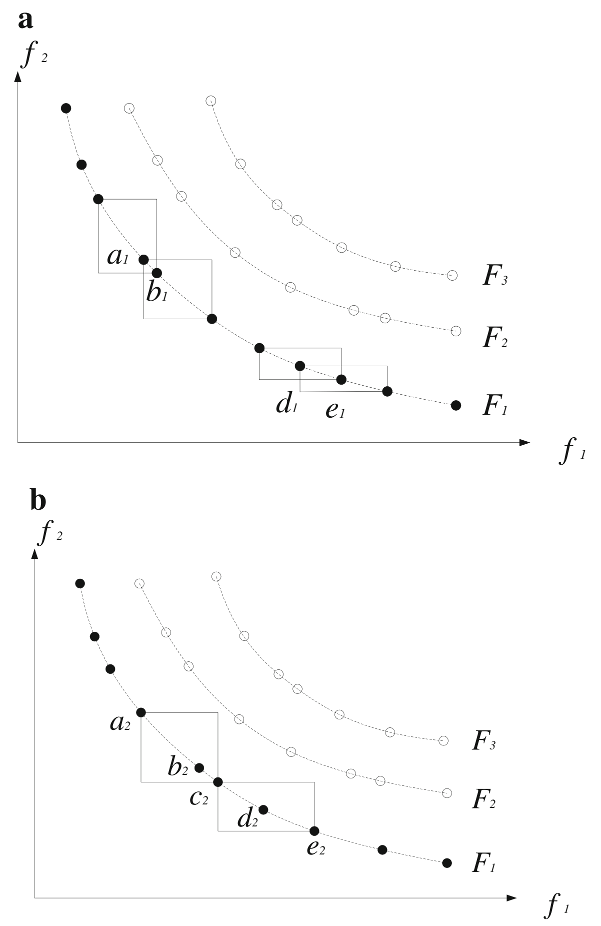

Fig. 5 Limitations of crowding distance calculation in NSGA-II. **a** the crowding distances of $I(a_1)$ and $I(b_1)$ are greater than $I(d_1)$ and $I(e_1)$. **b** $I(b_2)$ and $I(d_2)$ have the same crowding distances

crowding distance threshold of adjacent individuals in the same non-dominated front. If the distance between adjacent individuals is less than the threshold, some of the adjacent individuals are removed according to the elimination strategy. Secondly, the crowding distances are recalculated for the remaining individuals in the same front using Formula (13). Obviously, where the values of crowding distance are higher, this indicates that the distribution of individuals is better.

Crowding distance threshold: set θ_m^k is the threshold on the m^{th} objective function in the non-dominated front F_k. After sorting individuals by arbitrary objectives in front F_k, the threshold is calculated as follows:

$$\theta_m^k = \frac{f_m^{max} - f_m^{min}}{2(n-1)}, m = 1, 2, \ldots, M \tag{14}$$

The threshold is adjusted along with the evolution and the density of individuals. It is higher in the early evolution and becomes smaller in the later one. If $\forall m \in M$, $|I(i+1).m - I(i).m| < \theta_m^k$, then some similar individuals are removed by the following two strategies.

Removing strategy 1 If one of the solutions is a boundary solution in the front F_k, then the non-boundary solution is removed. The main purpose is to keep the boundary solutions and expand the scope of the non-dominated solution as soon as possible.

Removing strategy 2 If both solutions are not boundary solutions, we introduce variables φ_i and φ_j of individual $I(i)$ and $I(j)$ to judge which individual should be removed.

$$\begin{aligned} \varphi_i &= \sum_{m=1}^{M} [I(i).m - I(i-1).m] * [I(j+1).m - I(i).m], \\ \varphi_j &= \sum_{m=1}^{M} [I(j+1).m - I(j).m] * [I(j).m - I(i-1).m], \\ & \qquad j = i+1 \end{aligned} \tag{15}$$

If $\varphi_i > \varphi_j$, individual $I(i)$ will remain and individual $I(j)$ will be removed. In the opposite case, $I(j)$ will remain and $I(i)$ will be removed. As shown in Fig. 6, the distance variance σ_i^2 between individuals and their centre is

$$\begin{aligned} \sigma_i^2 &= \frac{\sum_{m=1}^{M} [I(i).m - I(i-1).m - \frac{I(j+1).m - I(i-1).m}{2}]^2}{M} \\ &= \frac{\sum_{m=1}^{M} [\frac{(I(j+1).m - I(i).m) - (I(i).m - I(i-1).m)}{2}]^2}{M} \\ &= \frac{1}{4M} \sum_{m=1}^{M} [I(j+1).m - I(i-1).m]^2 - \frac{1}{M}\varphi_i \end{aligned} \tag{16}$$

From Formula (16), $I(j+1).m - I(i-1).m$ is a fixed value for individual $I(i)$ and individual $I(j)$. Therefore, the larger the φ_i of individual $I(i)$ is, the smaller the σ_i^2 is. In other words, where the centre deviation between individuals $I(i)$, $I(i-1)$, and $I(j+1)$ is smaller, the preserved individual $I(i)$ is better for maintaining population diversity. It is worth noting that if $I(i)$ and $I(j)$ are repeated individuals, they would also be removed to avoid unnecessary genetic operation.

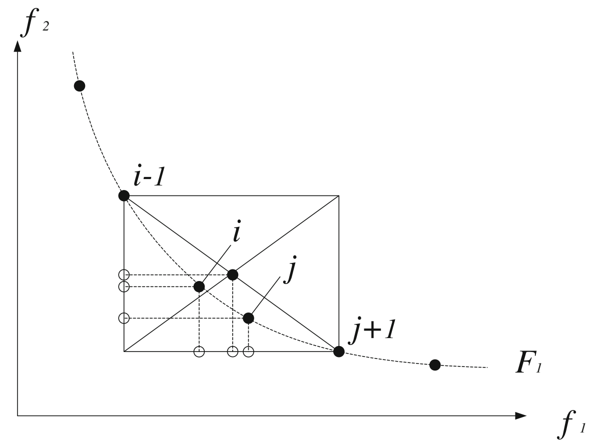

Fig. 6 Removing strategy for adjacent individuals in the same Pareto optimal front

4.6 Selection operator

The same tournament selection operator is used as in NSGA-*II*: (1) if the non-dominated fronts are different between two random individuals, the individual with the smaller front is selected to ensure optimization searching along the non-dominated solution direction and (2) if they belong to the same front, the individual with the greater crowding distance is selected to maintain the population diversity.

4.7 Improved crossover and mutation

Our proposed algorithm differs from NSGA-*II* in respect of the introduced hierarchical chromosome structure. It will use a different crossover and mutation operator based on the characteristics of the segment gene and the node gene. Because the starting segment of any path is the source segment, the first gene position of the segment gene does not participate in the crossover and mutation. However, all the gene positions of the node gene are involved in the genetic operation.

The segment gene uses "Order Crossover Operator (OX)". In essence, a swath of consecutive alleles from parent A drops down, and the remaining values are placed in the child in the order in which they appear in parent B, as shown in Fig. 7.

The node gene uses "Partial Matching Crossover Operator (PMX)". In essence, parent A donates a swath of genetic

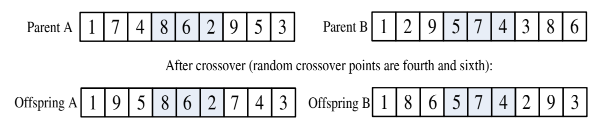
Parent A 1 7 4 8 6 2 9 5 3
Parent B 1 2 9 5 7 4 3 8 6
After crossover (random crossover points are fourth and sixth):
Offspring A 1 9 5 8 6 2 7 4 3
Offspring B 1 8 6 5 7 4 2 9 3

Fig. 7 Order crossover operator used in the segment gene

material, and the corresponding swath from the other parent is sprinkled around in the child. Once this is done, the remaining alleles are copied directly from parent B, as shown in Fig. 8.

The segment gene is mutated by an exchange mutation operator, involving the exchange of two randomly selected genes. The node gene is mutated by an integer mutation operator, which itself is replaced by another gene with a certain mutation probability. The mutation operators are as shown in Fig. 9.

4.8 Elite strategy

The same recombination and selection strategy are used as in NSGA-*II*. The offspring population is combined with the current generation population, and selection is performed to set the individuals of the next generation. Because all the previous and current best individuals are added into the population, elitism is ensured. The population is now sorted based on non-domination. The new generation is filled by each front subsequently until the population size exceeds the current population size. If by adding all the individuals in front F_k the population exceeds N, then individuals in front F_k are selected based on their crowding distance in descending order until the population size is N.

4.9 Complexity analysis

The iMTSP is an improved multi-travelling salesman problem, which belongs to the non-deterministic polynomial (NP) problems. The computational complexity of our proposed algorithm is mainly contained in the clustering algorithm and the multi-objective genetic algorithm, with the latter playing the dominant role. The overall complexity of our algorithm is $O(mN^2)$, in which m is the number of objective functions and N is the number of individuals in the initial population.

5 Performance evaluation

The performance evaluations are described in two different ways: firstly, we provide a verification of the

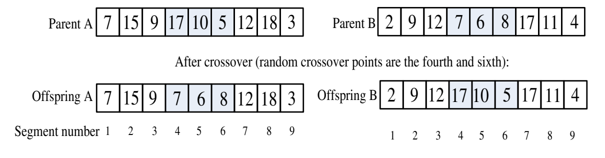
Parent A 7 15 9 17 10 5 12 18 3
Parent B 2 9 12 7 6 8 17 11 4
After crossover (random crossover points are the fourth and sixth):
Offspring A 7 15 9 7 6 8 12 18 3
Offspring B 2 9 12 17 10 5 17 11 4
Segment number 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

Fig. 8 Partial matching crossover operator used in the node gene

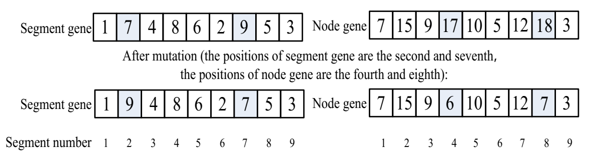
Segment gene 1 7 4 8 6 2 9 5 3
Node gene 7 15 9 17 10 5 12 18 3
After mutation (the positions of segment gene are the second and seventh, the positions of node gene are the fourth and eighth):
Segment gene 1 9 4 8 6 2 7 5 3
Node gene 7 15 9 6 10 5 12 7 3
Segment number 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

Fig. 9 Mutation operator used in the segment gene and node gene

effect of the improved NSGA-*II* algorithm on the distribution of the Pareto optimal solutions; secondly, we illustrate how to use MDCs to restore network connectivity from a damaged sensor network. We use Matlab 7.0 as computing tool to evaluate and analyse the results.

5.1 Evaluating performance of multi-objective optimization

Compared with NSGA-*II*, one of the main improvements of our proposed multi-objective optimization genetic algorithm is the introduction of the adjacent individuals' removal strategy to improve the diversity of the algorithm, known as NSGA-*II*-d. Using the existing metric in [27], we analyse and compare (1) convergence to the Pareto-optimal set and (2) maintenance of diversity in solutions of the Pareto-optimal set between NSGA-*II*-d and NSGA-*II*. The constrained test problems used in the evaluation are shown in Table 1.

Convergence metric γ: it is used to measure the extent of convergence to a known set of Pareto-optimal solutions, which is the minimum Euclidean distance between the chosen Pareto optimal solution and a known Pareto optimal solution. The smaller the value of this metric is, the better the convergence towards the Pareto-optimal front. Set Z is the chosen Pareto optimal solution; Z' is the known Pareto optimal solution. The calculation of the convergence metric is achieved as follows:

$$\gamma = \frac{1}{|Z|}\sum_{z\in Z} min\{|z-z'|, z' \in Z'\} \tag{17}$$

Diversity metric Δ: this measures the extent of the spread achieved among the obtained solutions. The smaller this value is, the more uniform the distribution. Set d_i contains the Euclidean distance between consecutive solutions in the obtained non-dominated set of solutions; d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained non-dominated set; $\bar{d}$ is the average of all distances d_i. There are N solutions on the best non-dominated front. With the solutions, there are $N-1$ consecutive distances. The calculation of the diversity metric is achieved as follows:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{Z-1}|d_i - \bar{d}|}{d_f + d_l + (|Z|-1)\bar{d}} \tag{18}$$

Table 1 Constrained test problems used in the evaluation (all objectives are to be minimized)

Problem	n	Variable bounds	Objective functions	Optimal solutions
SCH	1	[-10^3, 10^3]	$f_1(x) = x^2$,	x∈ [0, 2]
			$f_2(x) = (x-2)^2$	
ZDT1	30	[0, 1]	$f_1(x) = x_1$,	$x \in [0, 1]$,
			$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$,	$x_i = 0$,
			$g(x) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{(n-1)}$	$i = 2, \ldots, n$
ZDT2	30	[0, 1]	$f_1(x) = x_1$,	$x \in [0, 1]$,
			$f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2]$,	$x_i = 0$,
			$g(x) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{(n-1)}$	$i = 2, \ldots, n$
ZDT3	30	[0, 1]	$f_1(x) = x_1$,	$x \in [0, 1]$,
			$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)} \sin(10\pi x_1)]$,	$x_i = 0$,
			$g(x) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{(n-1)}$,	$i = 2, \ldots, n$
ZDT6	10	[0, 1]	$f_1(x) = 1 - exp(-4x_1)\sin^6(6\pi x_1)$,	$x \in [0, 1]$,
			$f_2(x) = g(x)[1 - (\frac{f_1(x)}{g(x)})^2]$,	$x_i = 0$,
			$g(x) = 1 + 9[\frac{\sum_{i=2}^{n} x_i}{(n-1)}]^{0.25}$	$i = 2, \ldots, n$

Each experimental result is averaged over 20 independent runs. The parameter settings are the same for NSGA-*II* and NSGA-*II*-d in the simulation: the population is 100, the iteration number is 1000, the crossover probability is 0.9, and the mutation probability is 0.1. The experimental results are shown in Table 2 and reveal that the diversity of NSGA-*II*-d is significantly better than that of NSGA-*II* for the same individual number and iteration number. The convergence of the two schemes is roughly equivalent.

5.2 Evaluating performance of multi-objective optimization

The experimental raw data were obtained from the TSPLIB of Heidelberg University, but we needed to preprocess these data to generate the testing instance, though this was not directly used for the experimental evaluation. Firstly, we chose a dataset from TSPLIB and divided all the cities into different segments using the clustering algorithm described in Section 4.1. Secondly, if the distance of the city from its own cluster centre was greater than the threshold, it was deleted. The iMTSP test instance was then generated.

The dataset of file ch150.tsp in the TSPLIB library was analysed in the experiment, consisting of 150 cities. After removing the cities that were more than 5000 away from their cluster centre, 52 cities remained, which were then classified into 10 segments. The source segment is the nearest one to the original point. The specific coordinates of the cites are given in Table 3. For convenience, the processed dataset is termed "10ch150.tsp", which represents the dataset ch150.tsp from TSPLIB classified into 10 segments.

The gene parameters are set as follows: the population is 100, the iteration number is 500, the crossover probability is 0.9, and the mutation probability is 0.05. We set the number of MDCs to 3, and two virtual segments (S_{11} and S_{12}) are constructed. Some output from one experiment are shown in Table 4.

In Table 4, we only list the top 15 results sorted by crowding distance in front F_1. The first and second groups

Table 2 Comparing convergence and diversity between NSGA-*II* and NSGA-*II*-d

Metric	Algorithm		SCH	ZDT1	ZDT2	ZDT3	ZDT6
γ	NSGA-*II*	Mean	0.001605	0.024019	0.031318	0.012930	0.047025
		Variance	0.000000	0.000026	0.000026	0.000007	0.000925
	NSGA-*II*-d	Mean	0.001632	0.024487	0.035320	0.014186	0.042749
		Variance	0.000000	0.000014	0.000090	0.000009	0.000684
Δ	NSGA-*II*	Mean	0.436119	0.432705	0.441009	0.619211	1.041551
		Variance	0.001019	0.000531	0.001030	0.001415	0.065354
	NSGA-*II*-d	Mean	0.391144	0.371493	0.422076	0.578343	1.035209
		Variance	0.000770	0.000654	0.019446	0.000721	0.043620

Table 3 *X*- and *Y*- coordinates of data in the file 10ch150.tsp

SegmentID	NodeID	*X*	*Y*	SegmentID	NodeID	*X*	*Y*
1	1	91.054	148.721	6	5	441.782	299.772
1	2	84.492	31.847	7	1	265.446	684.999
1	3	42.855	63.823	7	2	201.152	649.026
1	4	143.827	92.700	7	3	220.047	623.078
2	1	355.153	76.391	7	4	295.925	664.629
2	2	316.573	65.640	7	5	288.487	667.728
2	3	426.350	61.729	8	1	678.757	410.726
2	4	415.193	78.913	8	2	599.053	361.095
2	5	434.344	92.700	8	3	571.737	375.758
2	6	384.927	87.463	8	4	572.763	373.321
3	1	575.841	141.967	8	5	598.348	446.869
3	2	603.285	134.401	8	6	614.592	418.869
3	3	638.489	62.626	9	1	220.004	409.123
3	4	637.719	54.205	9	2	224.108	358.487
4	1	454.008	537.218	9	3	248.218	343.953
4	2	478.054	509.645	9	4	282.609	329.418
4	3	448.579	532.793	9	5	269.464	295.946
4	4	456.317	597.194	9	6	264.355	377.574
4	5	415.043	479.080	10	1	32.083	345.855
5	1	51.683	676.043	10	2	27.658	424.768
5	2	16.928	656.571	10	3	32.168	448.900
5	3	21.716	660.974	10	4	30.266	450.075
6	1	504.515	240.887	10	5	129.335	435.669
6	2	470.680	309.626	10	6	77.716	354.383
6	3	469.291	281.989	10	7	116.211	363.574
6	4	453.388	282.908	10	8	40.525	424.683

Table 4 Some experimental results of the 10ch150.tsp instance when MDCs=3

Case	NodeID(SegmentID in first row, NodeID of the segment in second row)											$f_1(x)$	$f_2(x)$
1	1	10	9	4	2	11	5	7	12	8	6	3	3913.407	2.935
	1	6	5	5	2	4	1	3	4	4	5	1		
2	1	9	1	2	11	10	5	7	4	8	6	3	2705.210	1543.176
	1	5	4	2	1	6	1	3	5	3	1	1		
3	1	2	12	9	11	10	5	7	4	8	6	3	2851.439	1414.994
	1	2	4	5	1	6	1	3	5	3	1	1		
4	1	10	5	7	12	9	2	11	4	8	6	3	3209.066	718.352
	1	6	1	2	4	5	2	4	5	3	1	1		
5	1	10	5	7	11	2	12	9	4	8	6	3	2910.723	1018.340
	1	6	1	2	4	2	4	5	5	3	1	1		
6	1	10	5	7	12	3	9	2	11	4	8	6	3565.864	7.718
	1	6	1	2	4	1	5	2	1	5	4	2		
7	1	10	5	7	12	3	2	11	9	4	8	6	3241.777	317.812
	1	6	1	2	4	1	5	4	5	5	3	2		
8	1	9	12	2	11	10	5	7	4	8	6	3	2779.414	1468.971
	1	5	4	1	1	6	1	3	5	3	1	1		
9	1	10	5	7	9	12	2	11	4	8	6	3	2984.387	1017.518
	1	6	1	3	5	4	2	4	5	3	1	1		
10	1	10	5	7	11	2	12	9	4	8	6	3	2984.927	944.136
	1	6	1	2	4	1	4	5	5	3	1	1		
11	1	10	5	7	11	2	12	9	4	8	6	3	3041.689	887.374
	1	6	1	2	1	2	4	5	5	3	1	1		
12	1	10	5	7	11	2	12	9	4	8	6	3	3142.053	787.010
	1	6	1	2	4	5	4	5	5	3	1	1		
13	1	10	5	7	12	3	2	11	9	4	8	6	3255.311	304.279
	1	6	1	2	4	1	3	4	5	5	3	2		
14	1	10	5	7	12	3	2	11	9	4	8	6	3294.741	264.848
	1	6	1	2	4	2	1	4	5	5	3	2		
15	1	10	5	7	11	2	12	9	4	8	6	3	3108.464	820.598
	1	6	1	2	3	2	4	5	5	3	1	1		

are the two boundary solutions in the front. The data of the first group indicate that the path of MDC-1 is 1(1)-10(6)-9(5)-4(5)-2(2)-1(1), the path of MDC-2 is 11(4)-5(1)-7(3)-11(4), the path of MDC-3 is 12(4)-8(4)-6(5)-3(1) , the minimum of objective function $f_1(x)$ is 2705.210, and the minimum of objective function $f_2(x)$ is 2.935. Generally speaking, when the number of MDCs is 3, the Pareto optimal fronts of the 10ch150.tsp instance are shown in Fig. 10.

It can be seen from Fig. 10 that the Pareto optimal front of iMTSP is discontinuous due to its discrete nature, which is different from the case for some continuous objective functions. For further analysis, we set the number of MDCs to be 2, 3, 4, and 5, respectively, with the same parameters of the algorithm as for file 10ch150.tsp. The experimental results are the minimum for $f_1(x)$ and $f_2(x)$ from 10 independent runs, which show the relationship among the number of MDCs and the total moving distance and the moving range of different MDCs. The detailed data are shown in Table 5.

Considering the variation in the minima of the objective function $f_1(x)$, it can be seen that where the number of MDCs is greater, the total moving distance is also greater. This is mainly due to that the starting and returning segment of each MDC is the same source segment. The larger the number of MDCs is, the higher the cost of returning to the source segment would require. When the number of MDCs is increased from 2 to 5, the total moving distance is increased from 2370.078 to 3761.941. The variation in the minima of objective function $f_2(x)$ shows that the number of MDCs is not directly related to the moving ranges. For example, when the number of MDCs is 2, 3, 4, and 5, the corresponding moving ranges are 0.005, 0.058, 5.260, and 2.409, respectively. The experiment shows that our proposed algorithm is effective and can be used to restore network connection, and it also implies that having more MDCs is not always better; it is necessary to consider each specific application.

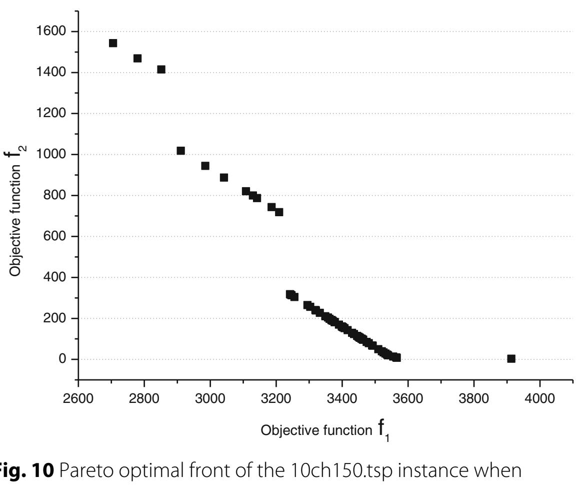

Fig. 10 Pareto optimal front of the 10ch150.tsp instance when MDCs=3

Table 5 Performance analysis of proposed algorithm for different MDCs using the 10ch150.tsp

Number of MDCs	Minimum of $f_1(x)$		Minimum of $f_2(x)$	
	$f_1(x)$	$f_2(x)$	$f_1(x)$	$f_2(x)$
2	2370.078	1670.668	2855.866	0.005
3	2705.210	1543.176	3590.704	0.058
4	3122.901	1549.337	4848.221	5.260
5	3761.941	1318.090	5129.410	2.409

6 Conclusions

The deployment of WSNs in extreme environments is prone to large-scale damage, which can cause disconnected segments and result in a network being unavailable. Different from common solution methods, we propose a new method of multi-objective optimized connectivity restoration using MDCs. The optimal path problem of MDCs is transformed to yield an improved multi-travelling salesman problem iMTSP, and the collection delay and task equilibrium during data collection are considered. The proposed method introduces virtual segments and a hierarchical chromosome structure and then uses an improved multi-objective optimization genetic algorithm based on NSGA-*II* to solve the optimization of the moving path of the MDCs. Simulation experiments show that the proposed method can effectively solve iMTSP and obtain a better diversity of Pareto optimum solutions. This could provide new ideas for WSN connectivity recovery technologies. In further research, we will focus on the following: (1) We will use our proposed algorithms in real situations and improve the convergence speed of our genetic algorithm. (2) We will use other multi-objective optimization algorithms apart from NSGA-*II* to solve optional paths and compare the differences in performance between different algorithms. (3) We will optimize the analysis in an attempt to decrease the complexity of the algorithm.

Acknowledgements
The work was supported by the National Natural Science Foundation of China under the grant 91118005 and the Fundamental Research Funds for the Central Universities of China under the grant 106112014CDJZR098801.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Key Laboratory of Ministry of Education for Dependable Service Computing in Cyber Physical Society, Chongqing University, Chongqing 400044, People's Republic of China. [2]School of Software Engineering, Chongqing University, Chongqing 400044, People's Republic of China. [3]Chongqing Jianzhu College, Chongqing 400072, People's Republic of China. [4]School of Automation, Chongqing University, Chongqing 400044, People's Republic of China.

Received: 5 October 2016 Accepted: 30 March 2017
Published online: 12 April 2017

References

1. S Lee, M Younis, Eqar: Effective qos-aware relay node placement algorithm for connecting disjoint wireless sensor subnetworks. IEEE Trans. Comput. **60**(12), 1772–1787 (2011)
2. YK Joshi, M Younis, in *2013 IEEE Global Communications Conference (GLOBECOM)*. Distributed approach for reconnecting disjoint segments (IEEE, 2013), pp. 255–260. http://ieeexplore.ieee.org/document/6831080/
3. S Chouikhi, IE Korbi, Y Ghamri-Doudane, LA Saidane, A survey on fault tolerance in small and large scale wireless sensor networks. Comput. Commun. **69**, 22–37 (2015)
4. M Younis, IF Senturk, K Akkaya, S Lee, F Senel, Topology management techniques for tolerating node failures in wireless sensor networks: A survey. Comput. Netw. **58**, 254–283 (2014)
5. L Shu, Y Zhang, LT Yang, Y Wang, M Hauswirth, N Xiong, Tpgf: geographic routing in wireless multimedia sensor networks. Telecommun. Syst. **44**(1), 79–95 (2010)
6. B Hao, H Tang, G Xue, in *2004 Workshop on High Performance Switching and Routing, 2004. HPSR*. Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Formulation and Approximation (IEEE, 2004), pp. 246–250. http://ieeexplore.ieee.org/document/1303479/
7. X Cao, X Han, C-C Shen, EL Lloyd, Fault-tolerant relay node placement in heterogeneous wireless sensor networks. IEEE Trans. Mob. Comput. **9**, 1536–1233 (2009)
8. B Khelifa, H Haffaf, M Madjid, D Llewellyn-Jones, in *2009 IEEE Symposium on Computers and Communications*. Monitoring Connectivity in Wireless Sensor Networks (IEEE, 2009), pp. 507–512. http://ieeexplore.ieee.org/document/5202236/
9. L Sitanayah, KN Brown, CJ Sreenan, A fault-tolerant relay placement algorithm for ensuring k vertex-disjoint shortest paths in wireless sensor networks. Ad Hoc Netw. **23**, 1570–8705 (2014)
10. HM Almasaeid, AE Kamal, in *2009 IEEE International Conference on Communications*. On the Minimum k-Connectivity Repair in Wireless Sensor Networks (IEEE, 2009), pp. 1–5. http://ieeexplore.ieee.org/document/5199257/
11. S Lee, MF Younis, AA Abbasi, A localized algorithm for restoring internode connectivity in networks of moveable sensors. IEEE Trans. Comput. **59**, 1669–1682 (2010)
12. A Abbasi, M Younis, U Baroudi, in *IEEE International Conference on Communications*. Restoring Connectivity in Wireless Sensor-Actor Networks with Minimal Topology Changes (IEEE, 2010), pp. 1–5. http://ieeexplore.ieee.org/document/5502448/
13. M Imran, M Younis, N Haider, MA Alnuem, Resource efficient connectivity restoration algorithm for mobile sensor/actor networks. EURASIP J. Wirel. Commun. Netw. **2012**(1), 1–16 (2012)
14. M Younis, K Akkaya, AA Abbasi, Movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE Trans. Parallel Distrib. Syst. **20**, 1366–1379 (2008)
15. A Alfadhly, U Baroudi, M Younis, in *2011 7th International Wireless Communications and Mobile Computing Conference*. Least Distance Movement Recovery Approach for Large Scale Wireless Sensor and Actor Networks (IEEE, 2011), pp. 2058–2063. http://ieeexplore.ieee.org/document/5982851/
16. G Xue, EL Lloyd, Relay node placement in wireless sensor networks. IEEE Trans. Comput. **56**, 134–138 (2007)
17. C Xiuzhen, D Ding-Zhu, W Lusheng, X Baogang, Relay sensor placement in wireless sensor networks. Wirel. Netw. **14**(3), 347–355 (2008)
18. S-F Hwang, W-L Chao, C-L Wu, C-R Dow, in *IEEE International Conference on Software Engineering and Service Science*. 2-Connected Relay Node Placement Scheme in Disjoint Wireless Sensor Networks (IEEE, 2014), pp. 1039–1043. http://ieeexplore.ieee.org/document/6933743/
19. S Kimence, I Bekmezci, Weighted relay node placement for wireless sensor network connectivity. Wirel. Netw. **20**(4), 553–562 (2014)
20. F Senel, M Younis, in *Global Telecommunications Conference*. Optimized Connectivity Restoration in a Partitioned Wireless Sensor Network (IEEE, 2011), pp. 1–5. http://ieeexplore.ieee.org/document/6134397/
21. C Donghui, D Dingzhu, H Xiaodong, L Guohui, W Lusheng, X Guoliang, Approximations for steiner trees with minimum number of steiner points. J. Glob. Optim. **18**(1), 17–33 (2000)
22. S Fatih, Y Mohamed, Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation. Comput. Commun. **34**(16), 1932–1941 (2011)
23. H Chen, K Shi, Quadrilateral steiner tree based connectivity restoration for wireless sensor networks. Chin. J. Comput. **37**(2), 457–469 (2014)
24. L Sookyoung, Y Mohamed, Optimized relay placement to federate segments in wireless sensor networks. IEEE J. Sel. Areas Commun. **28**(5), 742–752 (2010)
25. F Senel, MF Younis, K Akkaya, Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks. IEEE Trans. Veh. Technol. **60**(4), 1835–1848 (2011)
26. F Senel, M Younis, in *IEEE International Conference on Communications*. Optimized Interconnection of Disjoint Wireless Sensor Network Segments Using k Mobile Data Collectors (IEEE, 2012), pp. 492–496. http://ieeexplore.ieee.org/document/6364467/
27. K Deb, A Pratap, S Agarwal, T Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)