Soft Comput (2015) 19:2619–2632
DOI 10.1007/s00500-014-1429-z

METHODOLOGIES AND APPLICATION

Heuristic and genetic algorithms for solving survivability problem in the design of last mile communication networks

Huynh Thi Thanh Binh · Nguyen Thai Duong

Published online: 22 August 2014

Abstract Given a connected, undirected and weighted graph $G = (V, E)$, a set of infrastructure nodes J and a set of customers C include two customer types whereby customers C_1 require a single connection (type-1) and customers C_2 need to be redundantly connected (type-2). Survivable network design problem (SNDP) seeks a sub-graph of G with the smallest weight in which all customers are connected to infrastructure nodes. SNDP has applications in the design of the last mile of the real-world communication networks. SNDP is NP-hard so heuristic approaches are normally adopted to solve this problem, especially for large-scale networks. This paper proposes a new heuristic algorithm and a new genetic algorithm for solving SNDP. The proposed algorithms are experimented on real-world instances and random instances. Results of computational experiments show that the proposed heuristic algorithm is much more efficient than the other heuristics in running time, and the proposed genetic algorithm is much more efficient than the other heuristics in terms of minimizing the network cost.

Keywords Survivable network design · Fiber optic network · Shortest paths · Edge-disjoint paths · Heuristic algorithm · Genetic algorithm

Communicated by V. Loia.

H. T. T. Binh (✉) · N. T. Duong
School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
e-mail: binhht@gmail.com; binh.huynhthithanh@hust.edu.vn

N. T. Duong
e-mail: thaiduongnguyen91@gmail.com

1 Introduction

In the recent years, the increasing of communication demands requires more extended network and the standard quality of service is higher than ever. The customers require not only fast but also reliable connections. The word "reliable" has many meanings, but one of the most important meanings is survivable ability—having at least one back-up connection. It means that if the main connection is down, the network still works fine. Now, due to advantages of the optic cable such as large capacity, small size, light weight, security, it has become popular around the world. In this paper, we only focus optic cable. However, same as the optic cable, the coaxial cable also needs survivability to ensure reliability. So all the service providers need to solve the problem: how to connect to those complex clients with the lowest cost while still ensuring the survivable ability? This is one of the problems in the design of survivable networks.

In this work, we consider the problem of augmenting an existing network infrastructure by additional links (and switches) to connect potential customer nodes. There are two types of customers. In type-1, a standard, a single-link connection is sufficient, while type-2 customers require more reliable connections, ensuring connectivity even when a single link fails. The survivable network design problem (SNDP) can be stated formally as follows.

Input

- A connected, undirected graph $G = (V, E)$, with the non-negative weight function of its edges $w : E \to R^+$.
- Set of infrastructure nodes J and set of customer nodes C, ($J, C \subset V$ and $J \cap C = \varnothing$).
- Set of type-1 customer nodes C_1 and set of type-2 customer nodes C_2, where $\{C_1, C_2\}$ is a partition of

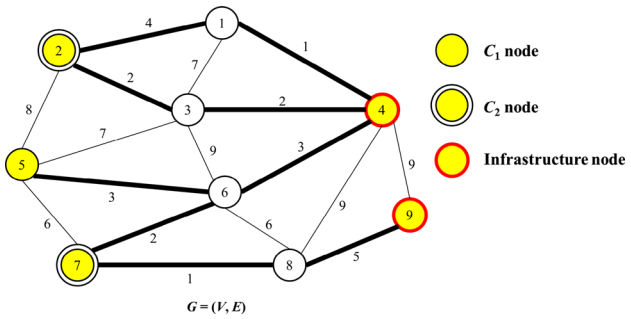

Fig. 1 Illustrate an example of SNDP

C (i.e. $C_1 \cup C_2 = C$ and $C_1 \cap C_2 = \emptyset$), and $|C_1|, |C_2| > 0$.

Constraints

- Simple connection: A customer node u from C_1 is feasibly connected if there exists a path from node u to infrastructure nodes.
- Redundant connection: A customer node u from C_2 is feasibly connected if there exist two edge-disjoint paths from node u to infrastructure nodes.

Output

- A sub-graph $G' = (V', E')$ of $G = (V, E)$, $(V' \subset V, E' \subset E)$ satisfies all the connection requirements posed by the customer nodes C_1 and C_2, and the weight of the sub-graph, $Cost(G') = \sum_{e \in E'} w(e)$ is minimal.

This problem is known to be NP-hard (Kerivin and Mahjoub 2005). Figure 1 illustrates a graph $G = (V, E)$ and an optimal solution (with bold edges) of SNDP in this graph.

The rest of this paper is organized as follows. Section 2 mentions related works of the problem. Sections 3 and 4 present proposed algorithms for the problem. Section 5 evaluates the performance of the proposed algorithms by experiments. We finally present conclusion and future works in Sect. 6.

2 Related work

A survey on methods for SNDP which can be seen as a more general version of our problem can be found in Kerivin and Mahjoub's paper (Kerivin and Mahjoub 2005). Techniques for solving SNDP may be classified into categories: exact methods and heuristic methods.

Exact approaches for solving SNDP are based on mixed linear integer programming. Wagner et al. (2007) modeled this problem as an integer linear program (ILP) by means of an extended multi-commodity network flow (MCF) formulation. With the general purpose ILP-solver CPLEX (IBM 2006), instances with up to total 190 nodes, 377 edges but only 6 customer nodes could be solved to prove optimality, and instances up to 2,804 nodes, 3,082 edges and 12 customer nodes could be solved with a final gap of about 7 %. This approach is unsuitable for larger instances and/or in particular instances with the larger number of customer nodes. In the research of Wagner et al. (2007), this problem is formulated based on directed connectivity constraints. Using a branch-and-cut algorithm, this model could be solved relatively well, and we were able to find proven optimal solutions for instances with up to 190 nodes, 377 edges, and 13 customer nodes. Ljubić et al. (2006) presented an exact method for the price-collecting Steiner tree problem (PCSTP) based on directed connection cuts. Other successful mathematical programming-based approaches include a relax-and-cut method by Da Cunha et al. (2009) and a cutting plane method by Lucena and Resende (2004). However, being deterministic and exhaustive in nature, these approaches could only be used to solve small problem instances (e.g. sparse graphs with the number of customers less than 15).

To solve this problem with larger instances, heuristic methods are especially interested in. Bucsics (2007) used meta-heuristic approaches such as local search and simulated annealing, variable neighborhood descent and variable neighborhood search. These approaches have solved and obtained some significant improvements for some problem instances. Leitner and Raidl (2008) formulated the problem as an abstract integer linear program and apply Lagrangian decomposition to obtain relatively tight lower bounds as well as feasible solutions. Furthermore, hybrid approaches of variable neighborhood search and greedy randomized adaptive search are applicable to larger problem instances. Canuto et al. (2001) described an effective multi-start local search approach based on perturbation of the nodes prizes, where path-relining and variable neighborhood search are used to further improve the obtained solutions. Preprocessing conditions for reducing the number of nodes and edges of a PCSTP instance have been described by Uchoa (2006). In addition, Chapovska and Punnen (2006) discussed the complexity of methods for several variants of the PCSTP. Leitner (2010) used mixed integer programming and hybrid optimization methods to solve this problem. Both approaches are able to derive proven optimal solutions or high quality solutions with small optimality gaps for medium-sized instances within reasonable time.

Most recently, Huynh et al. (Nguyen et al. 2012; Vo et al. 2012) solved the SNDP in the design of last mile communication networks using heuristic algorithms and gave better results than above works. However, the solutions of these heuristic algorithms are still far from optimal solutions in terms of network cost.

This paper introduces a new heuristic algorithm and a new genetic algorithm to solve large instances for SNDP. The proposed algorithms are expected to improve the best cost than previous works.

3 New heuristic algorithm for solving SNDP

In this section, we propose new heuristic algorithm called H-EDP to find the solution for SNDP. The algorithm finds in turn two edge- disjoint paths from each type-2 customer node to infrastructure nodes, and then finds in turn a shortest path from each type-1 customer node to infrastructure nodes. These paths make the sub-graph which is the solution for SNDP. Edge-disjoint shortest pair algorithm (Bhandari 1999) is used to find the shortest pairs of edge-disjoint paths for type-2 customer nodes. Dijkstra's algorithm (Cormen et al. 2009) is used to find the shortest paths for type-1 customer nodes. After H-EDP found a path (two edge-disjoint paths) for each type-1 customer node (each type-2 customer node), the weights of the edges on the found path(s) will be reassigned to 0, i.e. these edges can reuse without increasing the cost of the sub-graph. The pseudo-code of H-EDP heuristic algorithm is represented below.

In the pseudo-code, lines 1–5 create a new node t, representative for set of infrastructure nodes J. Then, we can find paths from customer nodes to t instead of finding paths to the set of infrastructure nodes. Line 6 initializes edge list L, set of the edges of the solution. Lines 7–19 find two edge-disjoint paths for each type-2 customer node by Edge-disjoint shortest pair algorithm and lines 20–28 find the shortest path for each type-1 customer node by Dijkstra's algorithm. Lines 12, 16 and 25 reassign the weights of the edges on the found paths. Customer nodes of each type (lines 8, 21) are selected randomly.

We use Dijkstra's algorithm with Fibonacci heap structure (Cormen et al. 2009) to find paths for $|C_1|$ type-1 customers and use Edge-disjoint shortest pair algorithm (Bhandari 1999) to find paths for $|C_2|$ type-2 customers. Finding a path for a type-1 customer node can be done in $O(|E| + |V| \log |V|)$. Edge-disjoint shortest pair algorithm can also run in $O(|E| + |V| \log |V|)$ to find two edge-disjoint paths for a type-2 customer node. Hence, in the worst case, H-EDP heuristic algorithm runs in $O((|C_1| + |C_2|) \cdot (|E| + |V| \log |V|))$. With the small asymptotic complexity, H-EDP can solve large instances.

Algorithm 1: H-EDP ($G = (V, E)$)

Input: Original graph $G = (V, E)$
Set of infrastructure nodes J
Set of type-1 customer nodes C_1 and set of type-2 customer nodes C_2
Output: Edge list L of sub-graph G'
begin
1. $V \leftarrow V \cup \{t\}$
2. **for each** $u \in J$
3. $E \leftarrow E \cup \{(u, t)\}$
4. $w(u, t) \leftarrow 0$
5. **end for**
6. $L \leftarrow \varnothing$
7. **while** $C_2 \neq \varnothing$
8. $s \leftarrow$ **random-select**(C_2)
9. $(P_1, P_2) \leftarrow$ **edge-disjoint-paths**(s, t, G)
10. **for each** $e \in P_1$
11. $L \leftarrow L \cup \{e\}$
12. $w(e) \leftarrow 0$
13. **end for**
14. **for each** $e \in P_2$
15. $L \leftarrow L \cup \{e\}$
16. $w(e) \leftarrow 0$
17. **end for**
18. $C_2 \leftarrow C_2 \setminus \{s\}$
19. **end while**
20. **while** $C_1 \neq \varnothing$
21. $s \leftarrow$ **random-select**(C_1)
22. $P \leftarrow$ **shortest-path**(s, t, G)
23. **for each** $e \in P$
24. $L \leftarrow L \cup \{e\}$
25. $w(e) \leftarrow 0$
26. **end for**
27. $C_1 \leftarrow C_1 \setminus \{s\}$
28. **end while**
29. **return** L
end

4 Genetic algorithm for solving SNDP

In this section, we propose a genetic algorithm called GA-EDP. The algorithm is expected to give better solutions in terms of minimizing the network cost for SNDP. H-EDP heuristic algorithm is used as an operation to develop the genetic algorithm. This genetic algorithm is presented in detail below.

4.1 Chromosome representation

Each chromosome represents a sub-graph $G' = (V', E')$ which is a solution of SNDP by the edge list of the sub-

graph. Each gene represents the corresponding edge of the sub-graph, i.e. chromosome $x = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$, $(u_i, v_i) \in E' \; \forall i \in \{1, 2, \ldots, n\}$, where n is the number of edges of G'. We can handle each chromosome by a dynamic array. The chromosome size is variable and does not exceed $|E|$, the number of edges of G.

4.2 Fitness function

The fitness function of chromosome x which represents sub-graph G' is defined by: $F(x) = \frac{1}{Cost(G')^2}$.

4.3 Initial population

Initially, each chromosome in population corresponds to a sub-graph. The sub-graph is "randomly" generated. It ensures connection required for all customers and may have an arbitrary network cost. We use three algorithms to generate the initial solutions for SNDP. These algorithms are presented following (Sects. 4.3.1–4.3.3).

4.3.1 Random weight path-based initialization algorithm (RPBA)

In the random weight path-based initialization algorithm (RPBA), we find random weight path(s) to infrastructure nodes for each customer node. All edges of these found paths form an initial solution. To find the random weight paths, we use algorithm 2 below.

Algorithm 2: Rand-weight-path (s, t, $G = (V, E)$)

Input: Graph $G = (V, E)$; $s, t \in V$
Output: Random weight path P from s to t in G
begin
1. **for each** $e \in E$
2. $w(e) \leftarrow$ **random-weight**( )
3. **end for**
4. Use Dijkstra's algorithm to find the shortest path P from s to t in G by edge weights $w(e)$
5. **return** P

end

Algorithm 2 uses Dijkstra's algorithm with Fibonacci heap structure (Cormen et al. 2009) in line 4. Thus, the complexity of this algorithm is $O(|E| + |V| \log |V|)$. The algorithm is used in the random weight path-based initialization algorithm (RPBA) to find paths to infrastructure nodes for customer nodes. The pseudo-code of RPBA is represented below (algorithm 3). With each type-1 customer node, RPBA uses algorithm 2 directly to find a simple path (lines 7–12). With each type-2 customer node, RPBA uses Ford–Fulkerson algorithm (Cormen et al. (2009)) to find two edge-disjoint paths (lines 13–21). In which, algorithm 2 is used in steps of finding augmenting paths of the Ford–Fulkerson algorithm.

Algorithm 3 (RPBA): Random weight path–based initialization algorithm ($G = (V, E)$)

Input: Original graph $G = (V, E)$
Set of infrastructure nodes J
Set of type-1 customer nodes C_1 and set of type-2 customer nodes C_2
Output: Edge list L of sub-graph G'
begin
1. $V \leftarrow V \cup \{t\}$
2. **for each** $u \in J$
3. $E \leftarrow E \cup \{(u, t)\}$
4. $w(u, t) \leftarrow 0$
5. **end for**
6. $L \leftarrow \varnothing$
7. **for each** $s \in C_1$
8. $P \leftarrow$ **Rand-weight-path**(s, t, G)
9. **for each** $e \in P$
10. $L \leftarrow L \cup \{e\}$
11. **end for**
12. **end for**
13. **for each** $s \in C_2$
14. Use Ford-Fulkerson algorithm to find two edge-disjoint paths P_1, P_2 from s to t
15. **for each** $e \in P_1$
16. $L \leftarrow L \cup \{e\}$
17. **end for**
18. **for each** $e \in P_2$
19. $L \leftarrow L \cup \{e\}$
20. **end for**
21. **end for**
22. **return** L

end

RPBA calls algorithm 2 in $|C_1|$ times to find $|C_1|$ simple paths for type-1 customer nodes. Each type-2 customer node needs two steps of finding augmenting paths to obtain two edge-disjoint paths from it (the source) to t (the sink). Thus, RPBA call algorithm 2 in $2.|C_2|$ times to find paths for $|C_2|$ type-2 customer nodes. The complexity of algorithm 2 is $O(|E|+|V| \log |V|)$ so the complexity of RPBA is $O((|C_1|+|C_2|) \cdot (|E| + |V| \log |V|))$.

4.3.2 H-EDP-based initialization algorithm (HEBA)

In this section, we propose the initialization algorithm based on heuristic algorithm H-EDP. This initialization algorithm is called HEBA. The algorithm includes two main steps as follows. Step 1: reassign edge weights of graph $G = (V, E)$ to random values (lines 3–5). Step 2: use H-EDP algorithm to find the solution for SNDP in the modified graph (line 6). The edge list of the solution is returned for the representative chromosome (line 7). The algorithm generally found a worse solution than H-EDP (i.e. it found a sub-graph with the greater cost) but it can find many "random solutions" in the original graph. Same as H-EDP, the algorithm can run in $O((|C_1| + |C_2|) \cdot (|E| + |V| \log |V|))$.

Algorithm 4 (HEBA): H-EDP–based initial algorithm ($G = (V, E)$)

Input: Original graph $G = (V, E)$
Set of infrastructure nodes J
Set of type-1 customer nodes C_1 and set of type-2 customer nodes C_2
Output: Edge list L of the sub-graph
begin
1. $V_1 \leftarrow V$
2. $E_1 \leftarrow E$
3. **for each** $e \in E_1$
4. $w(e) \leftarrow$ **random-weight**( )
5. **end for**
6. Use **H-EDP**($G_1 = (V_1, E_1)$) to find edge list L_1 of the solution for SNDP in graph G_1.
7. **return** $L = L_1$

end

4.3.3 OSSP-based initialization algorithm (OSBA)

Inspired from one-sourced shortest path algorithm (OSSP) proposed by Nguyen et al. (2012), we design an initialization algorithm which is called OSBA (OSSP-based initialization algorithm). This algorithm includes three main steps as follow. Step 1: select randomly node $s \in V$. This node is called "central node." Step 2: find the shortest path tree from s to all terminal nodes (denote terminal nodes including infrastructure nodes and customer nodes C_1, C_2). If the edges of the shortest path tree are used, each customer node has at least one path from it to infrastructure nodes. Step 3: find augmenting paths to satisfy the redundant connection condition for type-2 customer nodes. Dijkstra's algorithm and Ford–Fulkerson algorithm (Cormen et al. 2009) are used in step 2 and step 3 of the algorithm, respectively. The complexity of OSBA is $O(|V| \log |V| + |C_2| \cdot (|V| + |E|))$.

4.4 Crossover operators

Two crossover operators are used in our genetic algorithm. These crossover operators are applied with probability p_c. After a child chromosome is created, it will replace the parent having fitness function greater than the one. The crossover operators are presented in Sects. 4.4.1 and 4.4.2.

4.4.1 Heuristic-based crossover (HBC)

Heuristic-based crossover (HBC) operator between two chromosomes x and y is executed by creating a new graph containing all vertices and edges of the sub-graphs which are represented as x and y, and then finding a new solution in the new graph for the child chromosome by an algorithm solving SNDP. The pseudo-code of this crossover operator is represented below (algorithm 5). We use HEBA initialization algorithm to find the new solution in HBC (line 5). The crossover operator runs in time that is $O((|C_1|+|C_2|)\cdot(|E|+|V|\log|V|))$. Figure 2 illustrates the crossover operator.

Algorithm 5: HBC Crossover (x, y)

Input: Two parent chromosomes x, y
Output: Child chromosome z
begin
1. $G_1 = (V_1, E_1) \leftarrow$ Sub-graph is represented by x
2. $G_2 = (V_2, E_2) \leftarrow$ Sub-graph is represented by y
3. $V^* \leftarrow V_1 \cup V_2$
4. $E^* \leftarrow E_1 \cup E_2$
5. $L \leftarrow$ **HEBA** ($G^* = (V^*, E^*)$)
6. z represents the sub-graph is formed by edge list L
7. **return** z

end

4.4.2 Randomness-based crossover (RBC)

Assume that chromosomes x and y represent the sub-graphs G_1 and G_2, respectively. Randomness-based crossover (RBC) operator between chromosomes x and y is executed to create new solution for their child chromosome, as follow (algorithm 6): with each customer node, select path(s) to infrastructure nodes of it in either G_1 or G_2 randomly (lines 5–10, 16–21) and then, insert the edges of the selected path(s) to the edge list of the child chromosome (lines 11–13, 22–27). This crossover operator runs in time that is $O((|C_1| + |C_2|) \cdot (|V| + |E|))$.

Algorithm 6: RBC Crossover (x, y)

Input: Two parent chromosomes x, y
Output: Child chromosome z
begin
1. $G_1 = (V_1, E_1) \leftarrow$ Sub-graph is represented by x
2. $G_2 = (V_2, E_2) \leftarrow$ Sub-graph is represented by y
3. $L \leftarrow \varnothing$
4. **for each** $i \in C_1$
5. $r \leftarrow$ **random-number**(0, 1)
6. **if** ($r \leq 0.5$)
7. $P \leftarrow$ Path from i to infrastructure nodes in G_1
8. **else**
9. $P \leftarrow$ Path from i to infrastructure nodes in G_2
10. **end if**
11. **for each** $e \in P$
12. $L \leftarrow L \cup \{e\}$
13. **end for**
14. **end for**
15. **for each** $i \in C_2$
16. $r \leftarrow$ **random-number**(0, 1)
17. **if** ($r \leq 0.5$)
18. $(P_1, P_2) \leftarrow$ Edge disjoint paths from i to infrastructure nodes in G_1
19. **else**
20. $(P_1, P_2) \leftarrow$ Edge disjoint paths from i to infrastructure nodes in G_2
21. **end if**
22. **for each** $e \in P_1$
23. $L \leftarrow L \cup \{e\}$
24. **end for**
25. **for each** $e \in P_2$
26. $L \leftarrow L \cup \{e\}$
27. **end for**
28. **end for**
29. z represents the sub-graph is formed by edge list L
30. **return** z

end

4.5 Mutation operator

Mutation operator is applied with probability p_m. Mutation operator for chromosome x is executed by randomly selecting some vertices not belonging to the graph represented by x, and adding these selected vertices with all their adjacent edges to this graph to obtain a new one. Then, we find a new solution on this new graph. Figure 3 illustrates the mutation operator. The pseudo-code of mutation operator is represented below (algorithm 7). H-EDP heuristic algorithm has the small computational complexity, so using it is appropriate to create the new solution in the mutation operator (line 9). Probability to select a new vertex and add its adjacent edges depends on parameter p_s. The computational complexity of the mutation operator is $O((|C_1|+|C_2|)\cdot(|E|+|V|\log|V|))$.

Algorithm 7: Mutation (x, p_s)

Input: Chromosome x
Parameter p_s of probability to select a new vertex randomly
Output: Chromosome x after mutation
begin
1. $G_1 = (V_1, E_1) \leftarrow$ sub-graph is represented by x
2. **for each** $v \in V \backslash V_1$
3. $r \leftarrow$ **random-number**(0, 1)
4. **if** ($r \leq p_s$)
5. $V_1 \leftarrow V_1 \cup \{v\}$
6. $E_1 \leftarrow E_1 \cup \{$adjacent edges of $v\}$
7. **end if**
8. **end for**
9. $L_1 \leftarrow$ **H-EDP**($G_1 = (V_1, E_1)$)
10. x represents the sub-graph is formed by edge list L_1

end

5 Computational results

5.1 Problem instances

The problem instances used in our experiments are the real-world instances and random instances. The real-world instances include 135 Euclidean instances and the random instances include 98 Non-Euclidean instances. The real-world instances are data from a German city and are used in Bachhiesl (2005), Bucsics (2007), Leitner and Raidl (2008, 2010), Nguyen et al. (2012) and Vo et al. (2012). Problem sizes of the real-world instances are summarized in Table 1. All of the real-word instances can be downloaded at: https://www.ads.tuwien.ac.at/people/mleitner/sndp/sndpinstances.tar.gz.

The random instances are sparse graphs (SNDP.Rxxx) with problem sizes such as Table 2. All of the edge weights of the sparse graphs are generated in [5000, 105000] randomly. The random instances are also used in Nguyen et al. (2012) and Vo et al. (2012).

We created two sets of experiments. In the first set of experiments, we compare the performance of the heuristic algorithms: APSP, RNS (Bucsics 2007; Vo et al. 2012) which are

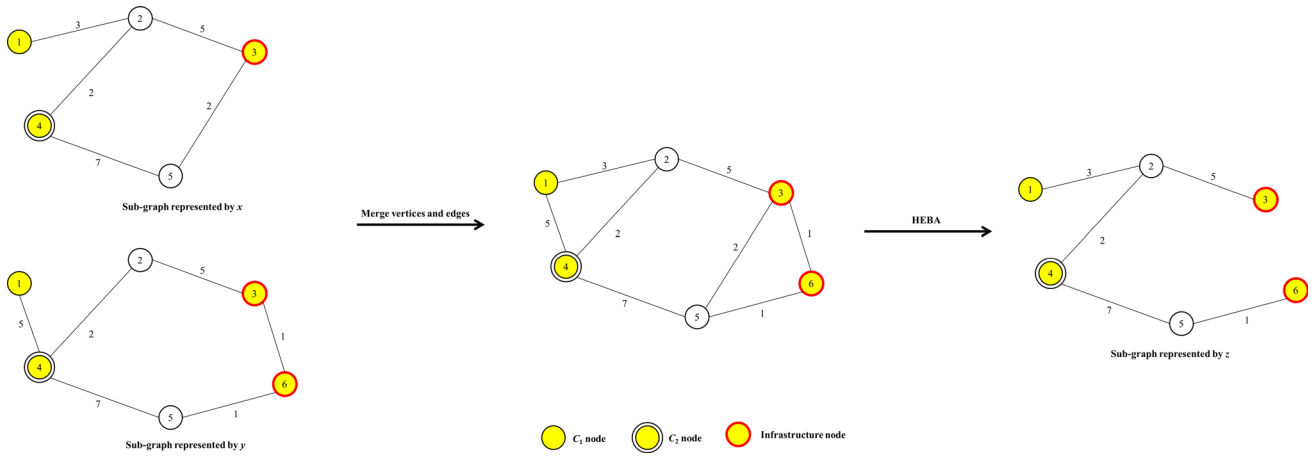

Fig. 2 Illustrate crossover operator HBC

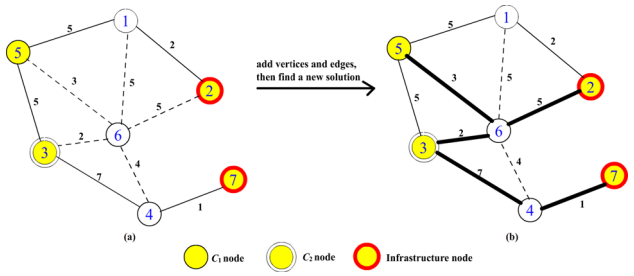

Fig. 3 Illustrate mutation operator. **a** Sub-graph (*solid lines*) represented by x before mutation. Vertex 6 is selected, the dashed lines represent its adj. edges. **b** Sub-graph (*bold edges*) represented by x after mutation

Table 1 Real-world problem instances

Set	#	$\|V\|$	$\|E\|$	$\|C\|$	$\|C_1\|$	$\|C_2\|$
ClgSExtra	25	190	377	5–8	3–5	2–3
ClgSExtra-I2	15	190	377	11–17	7–12	4–7
ClgSExtra-I3	15	190	377	8–12	5–8	3–6
ClgMExtra	25	1,757	3,877	6–10	4–7	2–3
ClgMExtra-I2	15	1,523	3,290	11–14	8–11	3–4
ClgN1BoundI1	20	2,804	3,082	10–14	8–11	3–4
ClgN1BoundI2	20	2,804	3,082	8–12	3–6	4–6

the ones of the best algorithms in the known heuristic algorithms, and H-EDP, GA-EDP on the real-world instances. In the second set of experiments, we compare the performance of these algorithms on the random instances.

5.2 System setting and parameters

In the experiment, with the compared algorithms, the system was run 20 times for each problem instance. In GA-EDP, we use 2000 generations of population, the population includes 200 chromosomes. RPBA, HEBA and OSBA algorithm (Sects. 4.3.1–4.3.3) are used for initializing; the rates of them in the population are 30, 35 and 35 %, respectively; Parameters $p_c = 0.2$, $p_m = 0.05$ and $p_s = 0.75$. HBC and RBC crossover are used with the rate 70 and 30 %, respectively. All the programs were run on a machine with Intel Core i3-540 3.06GHz, 2GB RAM, and were installed by C++ language in Code::Blocks 10.05 IDE.

Table 2 Random problem instances

Set	#	$\|V\|$	$\|E\|$	$\|C\|$	$\|C_1\|$	$\|C_2\|$
SNDP.R1xx	1	190	450	9–9	6–6	3–3
SNDP.R2xx	29	190	500	6–18	4–10	2–8
SNDP.R3xx	10	1,523	3,290	5–7	3–5	2–3
SNDP.R4xx	10	1,757	3,877	5–10	3–7	2–4
SNDP.R5xx	9	2,000	5,000	7–14	4–10	2–4
SNDP.R6xx	20	2,400	5,000	6–13	3–11	2–5
SNDP.R7xx	4	2,800	3,500	7–10	4–7	2–4
SNDP.R8xx	5	3,867	8,477	5–10	3–7	2–4
SNDP.R9xx	10	3,867	46,307	7–14	3–11	2–4

#, number of instances; $|V|$, number of vertices; $|E|$, number of edges; $|C|$, interval of customer number; $|C_1|$, interval of type-1 customer number; $|C_2|$, interval of type-2 customer number

5.3 Results of computational experiments on real-world instances

Figures 4, 5, 6, 7, 8, 9 and 10 represent the best results in 20 running times of algorithms: APSP, RNS, H-EDP, GA-EDP on the sets of real-world instances: ClgSExtra, ClgSExtra-I2, ClgSExtra-I3, ClgMExtra, ClgMExtra-I2, ClgN1BoundI1, ClgN1BoundI2 (Leitner and Raidl 2008; Nguyen et al. 2012; Vo et al. 2012). The figures show that, on the real-world problem instances, the best results found by GA-EDP are better than that found by the other algorithms on almost problem

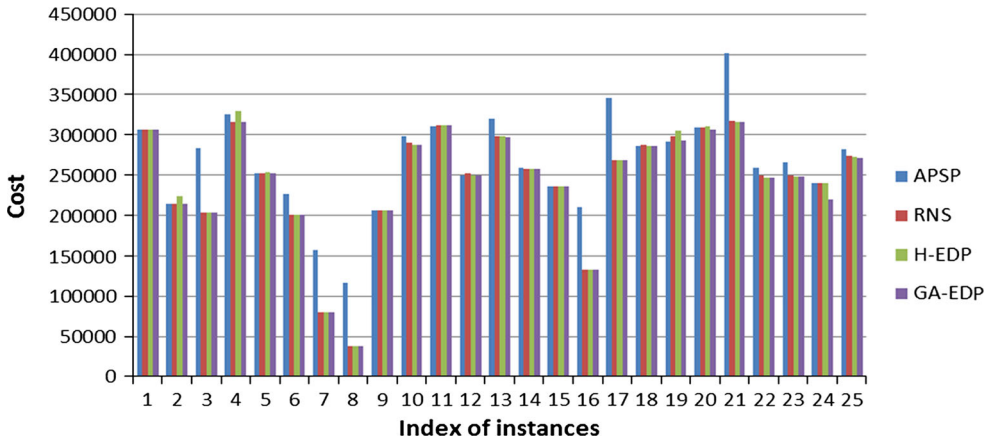

Fig. 4 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgSExtra instances

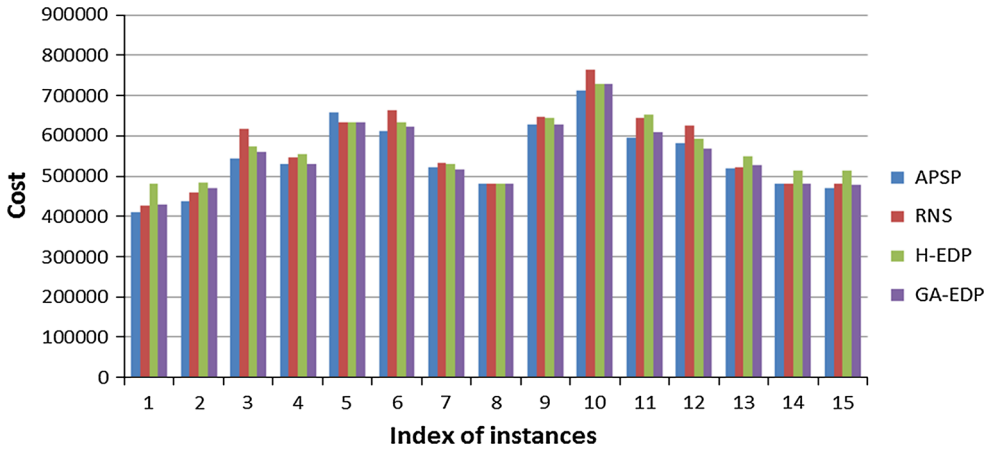

Fig. 5 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgSExtra-I2 instances

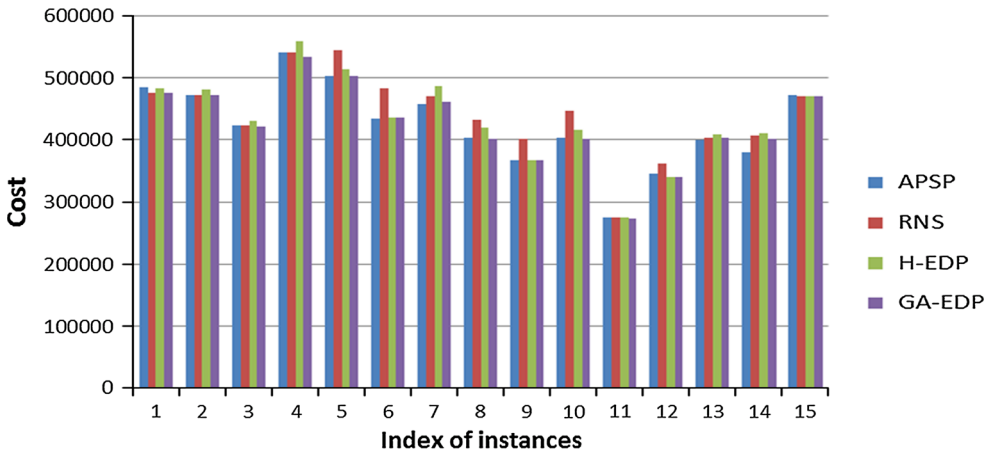

Fig. 6 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgSExtra-I3 instances

instances. This is because GA-EDP has the diverse initialized population which is generated by three initialization algorithms and also has effective crossover and mutation operators developed from heuristics. Moreover, its parameters are selected appropriately.

The best results found by H-EDP are worse than that found by APSP, RNS, GA-EDP. On these real-world instances, due to the small computational complexity, H-EDP is the fastest algorithm (see Table 3). Its average running time is much smaller than that of the other algorithms, (e.g. in

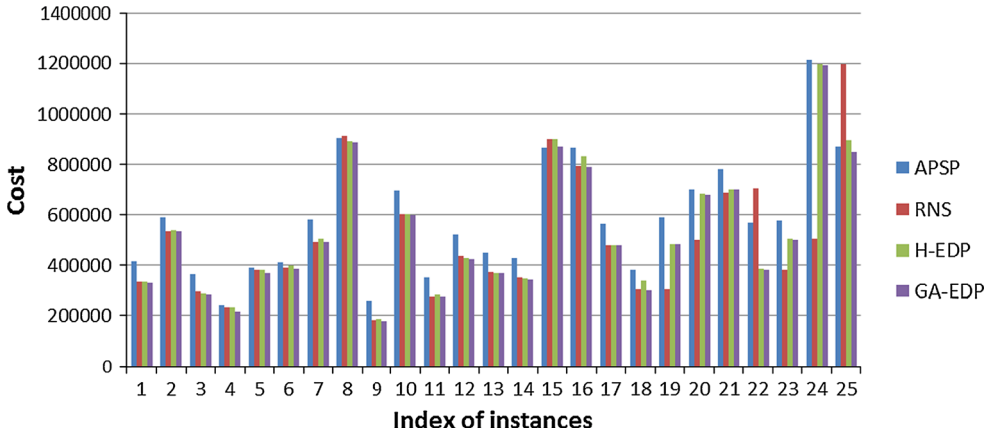

Fig. 7 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgMExtra instances

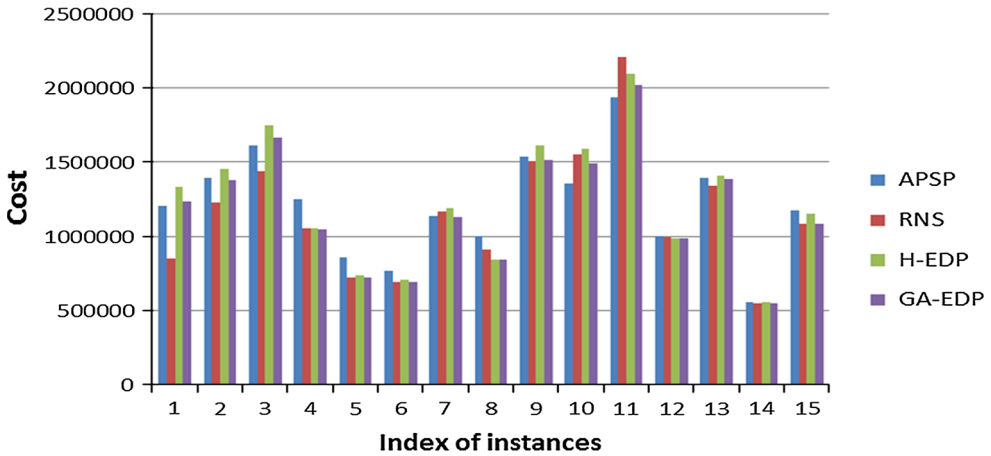

Fig. 8 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgMExtra-I2 instances

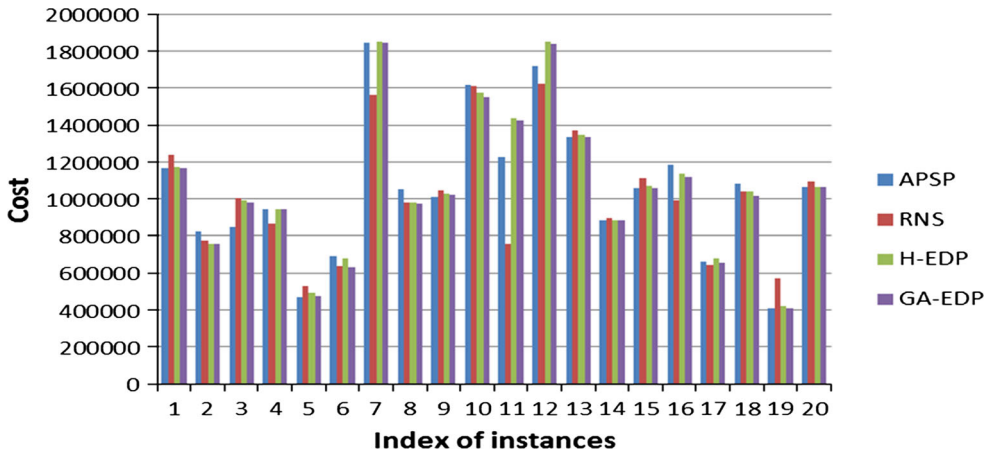

Fig. 9 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgN1BoundI1 instances

ClgN1BoundI2 set, the algorithm spends averagely 0.19 s for each instance that is about 1/500 times compared to RNS, GA-EDP and about 1/10 times compared to APSP). The average running time in the real-world instances of GA-EDP and RNS is almost equivalent, and is the greatest in the compared algorithms. However, both these running time are acceptable for the problem. Figures 11, 12, 13, 14, 15, 16 and 17 represent the average results found by GA-EDP in 20 running times with 95 % confidence intervals (Nakayama 2002). These confidence intervals show that the algorithm is very stable.

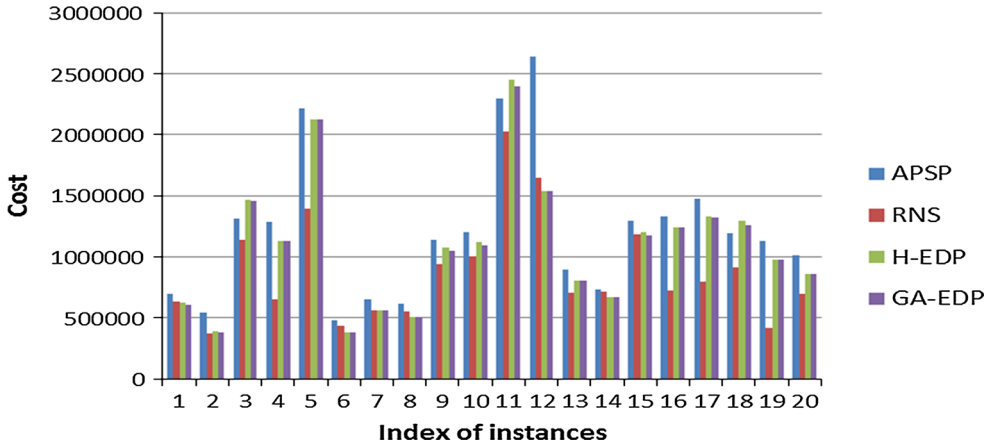

Fig. 10 The best results found by APSP, RNS, H-EDP and GA-EDP on ClgN1BoundI2 instances

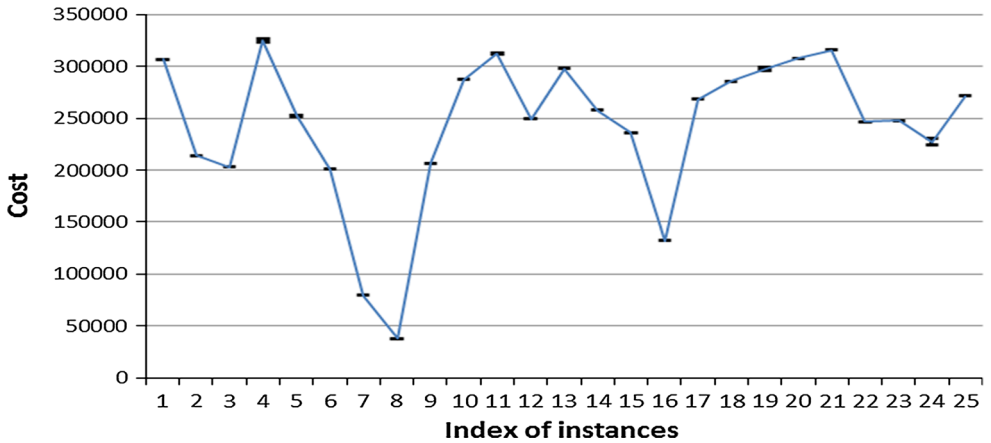

Fig. 11 The average results found by GA-EDP on ClgSExtra instances and 95 % confidence intervals

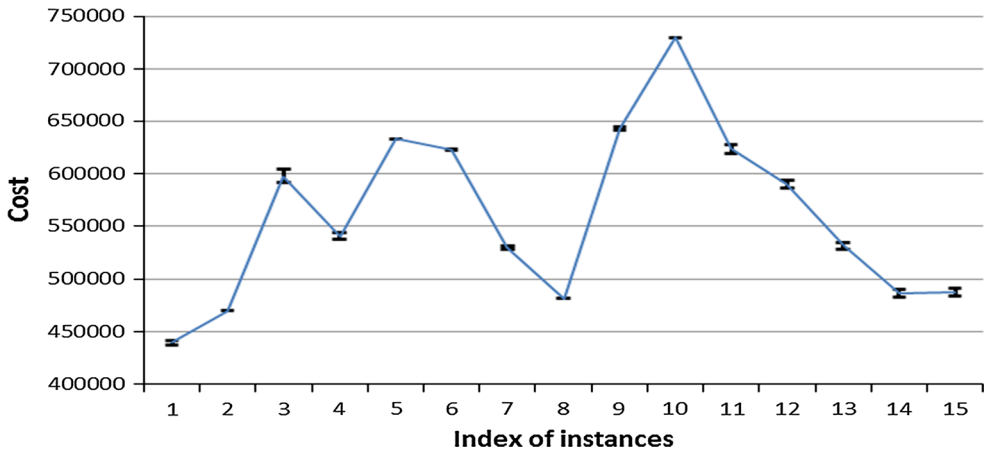

Fig. 12 The average results found by GA-EDP on ClgSExtra-I2 instances and 95 % confidence intervals

5.4 Results of computational experiments on random instances

Figure 18 represents the best results in 20 running times of the compared algorithms on the 98 random instances. The results show that on the random problem instances, the proposed genetic algorithm GA-EDP is very effective. In almost cases, the best results found by this algorithm are better than that found by the other algorithms. Figure 19 represents the average results found by the genetic

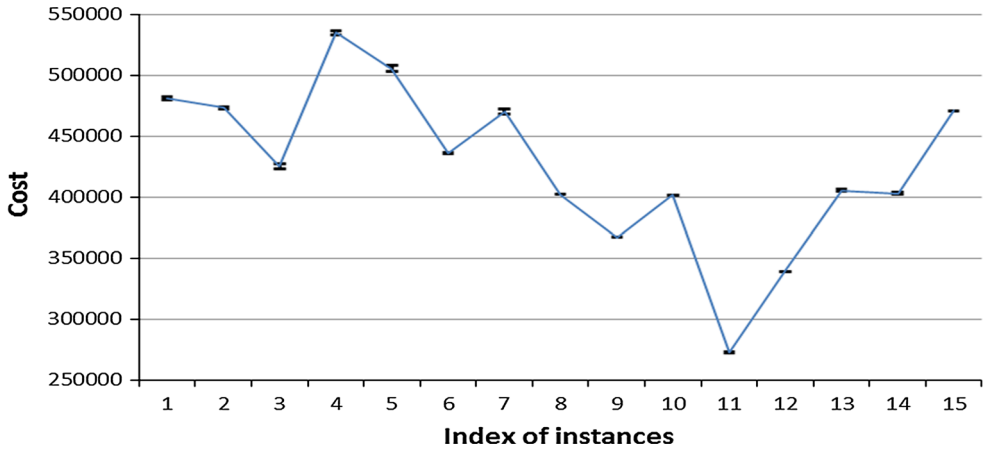

Fig. 13 The average results found by GA-EDP on ClgSExtra-I3 instances and 95 % confidence intervals

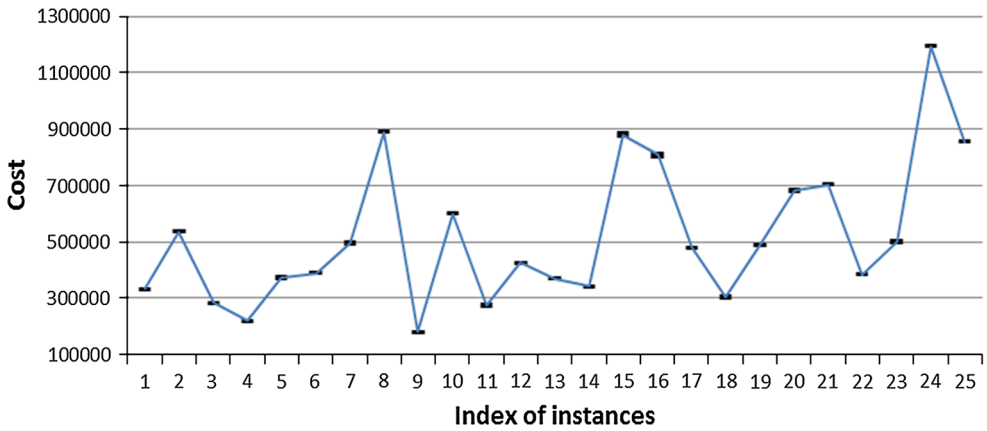

Fig. 14 The average results found by GA-EDP on ClgMExtra instances and 95 % confidence intervals

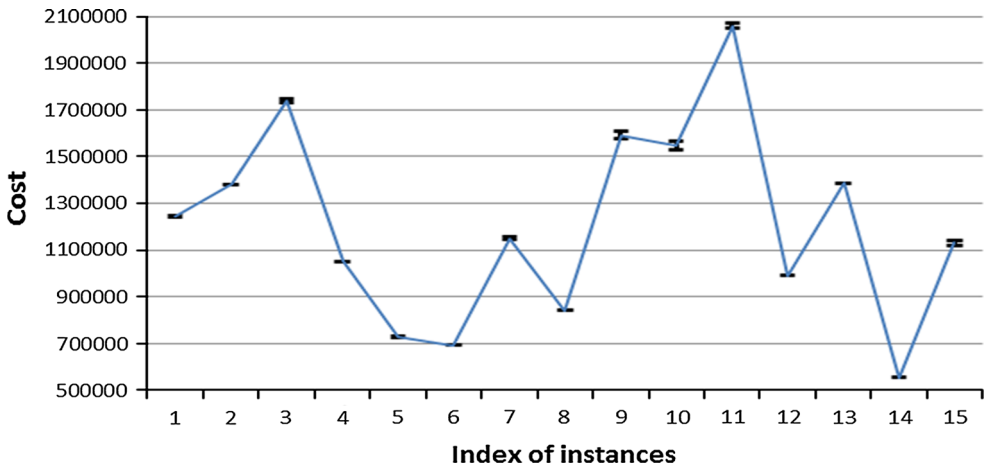

Fig. 15 The average results found by GA-EDP on ClgMExtra-I2 instances and values and 95 % confidence intervals

algorithm in 20 running times with 95 % confidence intervals. These confidence intervals are very small compared to the average results. Table 3 represents the average running time of the compared algorithms on the random instances. The genetic algorithm has the largest running time, about 2 min for each random instance. But the running time is still acceptable for the NP-hard problem. With the small computational complexity, the proposed heuristic

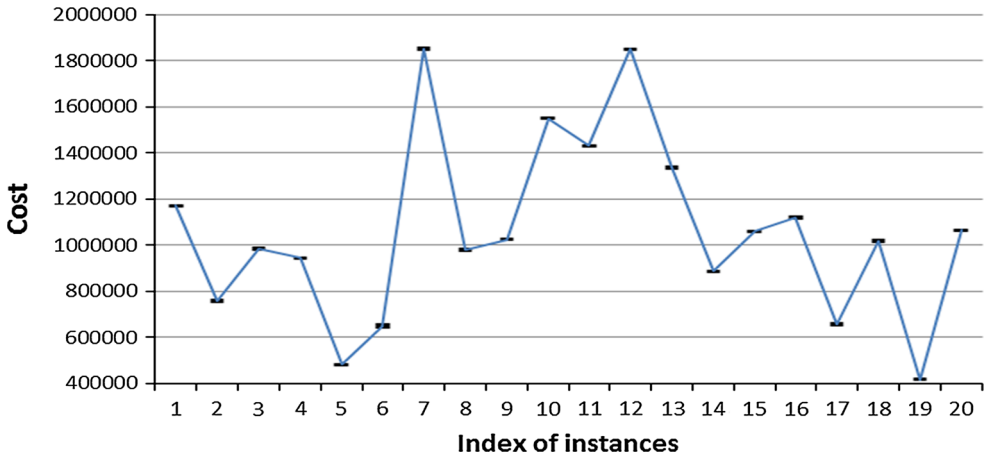

Fig. 16 The average results found by GA-EDP on ClgN1BoundI1 instances and values and 95 % confidence intervals

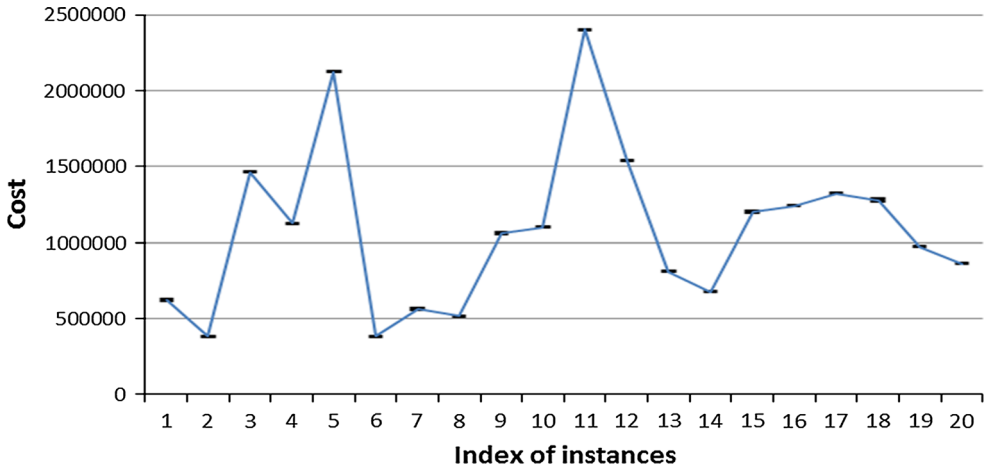

Fig. 17 The average results found by GA-EDP on ClgN1BoundI2 instances and values and 95 % confidence intervals

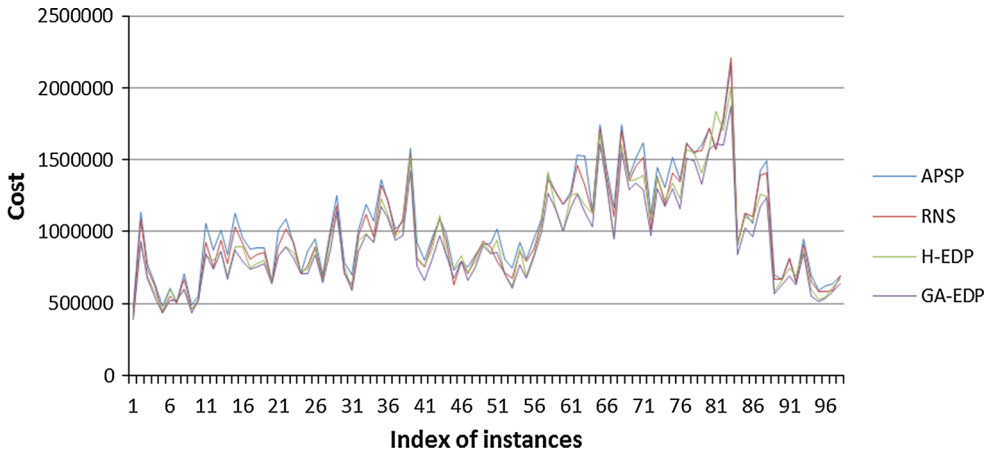

Fig. 18 The best results found by APSP, RNS, H-EDP and GA-EDP on random instances

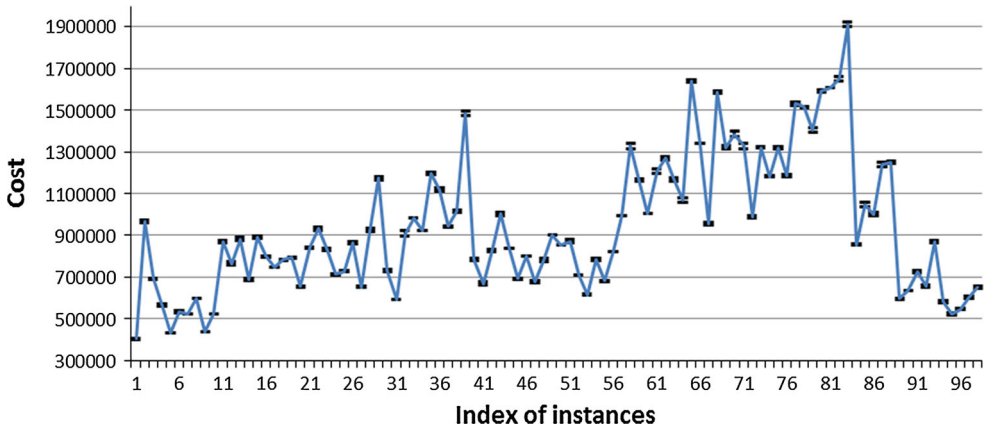

Fig. 19 The average results found by GA-EDP on Random instances and values and 95 % confidence intervals

Table 3 The average running time of APSP, RNS, H-EDP and GA-EDP on the sets of instances (in second)

Set	APSP	RNS	H-EDP	GA-EDP
ClgSExtra	0.02	1.64	0.00	3.61
ClgSExtra-I2	0.18	4.37	0.00	6.78
ClgSExtra-I3	0.05	2.83	0.00	5.71
ClgMExtra	1.11	43.07	0.05	48.66
ClgMExtra-I2	3.03	52.12	0.04	40.70
ClgN1BoundI1	1.37	80.22	0.13	79.36
ClgN1BoundI2	1.98	103.04	0.19	99.24
Random	0.56	13.54	0.05	116.21

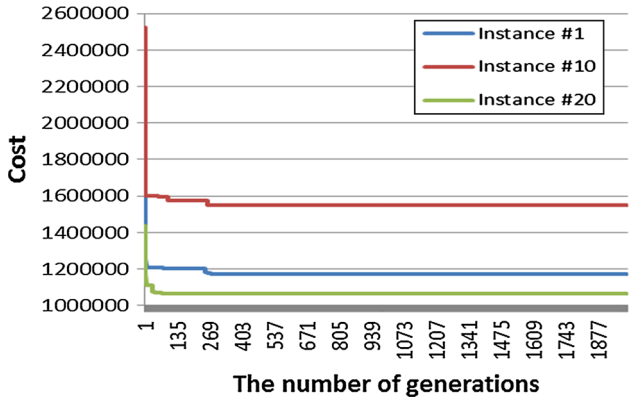

Fig. 20 Results of GA-EDP depend on generations (on the instances of ClgN1BoundI1 set)

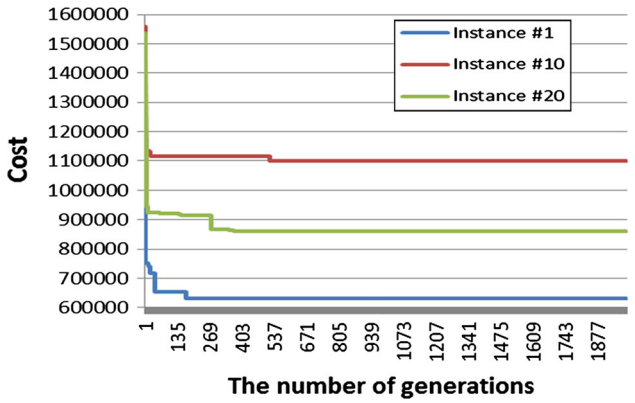

Fig. 21 Results of GA-EDP depend on generations (on the instances of ClgN1BoundI2 set)

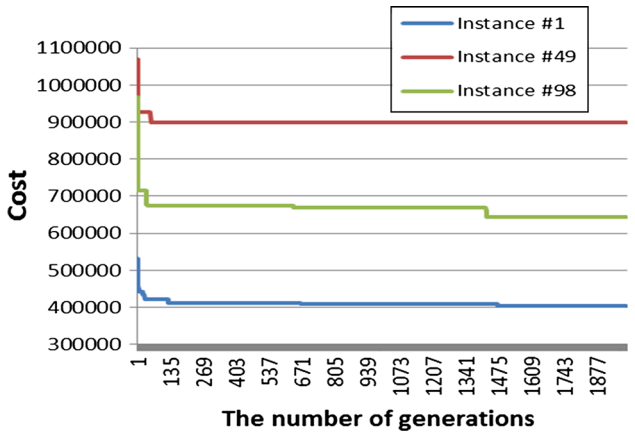

Fig. 22 Results of GA-EDP depend on generations (on the instances of Random set)

algorithm H-EDP is still the fastest algorithm on the random instances.

Figures 20, 21 and 22 show the dependence of results of the genetic algorithm on the number of generations in instances of large instance sets. These figures show that the genetic algorithm suffers stagnation around 100–1,500 generations.

6 Conclusion

In this paper, we propose a new heuristic called H-EDP and a new genetic algorithm called GA-EDP for solving SNDP. We experimented on 135 real-world instances derived from

a library of test instances for Survivable fixed telecommunication Network Design (SNDLib) and 98 random instances. On the real-world instances and random instances, H-EDP is efficient in terms of running time. The network costs found by GA-EDP algorithm are better than that found by the other algorithms in almost the instances. The results show that genetic algorithms can be attractive for solving SNDP.

In the future, we are planning to improve the algorithms for solving SNDP with multi-objective. Moreover, a direction for further research could be a study of the other survivable models such as multicast, any-cast models.

Acknowledgments This work was supported by the project "Models for the next generation of a robust Internet," Grant Number 12/2012/HD-NDT, funded by the Ministry of Science and Technology, Vietnam.

References

Bachhiesl P (2005) The OPT- and the SST-problems for real world access network design basic definitions and test instances. Working report 01/2005, Carinthia Tech Institue, Department of Telematics and Network Engineering, Klagenfurt, Austria

Bhandari R (1999) Survivable networks: algorithms for diverse routing. Springer Int Ser Eng Comput Sci 477:46

Bucsics T (2007) Metaheuristic approaches for designing survivable fiber-optic networks. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria

Canuto SA, Resende MGC, Ribeiro CC (2001) Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks 38:50–58

Chapovska O, Punnen AP (2006) Variations of the prize-collecting Steiner tree problem. Networks 47(4):199–205

Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms, 3rd edn. The MIT Press, Cambridge

Da Cunha AS, Lucena A, Maculan N, Resende MGC (2009) A relax-and-cut algorithm for the prize-collecting Steiner problem in graph. Discrete Appl Math 157(6):1198–1217

IBM (2006) ILOG CPLEX optimizer performance benchmarks 10.0. http://www.cs.cornell.edu/w8/iisi/ilog/cplex100/index.html. Accessed 15 Aug 2014

Kerivin H, Mahjoub AR (2005) Design of survivable networks: a survey. Networks 46:1–21

Leitner M (2010) Solving two network design problems by mixed integer programming and hybrid optimization methods. Ph.D. thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria

Leitner M, Raidl GR (2008) Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks. In: Blesa MJ et al (ed) Hybrid Metaheuristics, LNCS, vol 5296. Springer, Berlin, Heidelberg, pp 158–174

Leitner M, Raidl GR (2010) Strong lower bounds for a survivable network design problem. International symposium on combinatorial optimization. Hammamet, Tunisia, pp 295–302

Ljubić I, Weiskircher R, Pferschy U, Klau G, Mutzel P, Fischetti M (2006) An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. Math Program Ser B 105(2–3):427–449

Lucena A, Resende MGC (2004) Strong lower bounds for the prize collecting Steiner problem in graphs. Discrete Appl Math 141(1–3):277–294

Nakayama MK (2002) Simulation output analysis. In: Proceedings of the winter simulation conference, pp 23–34

Nguyen MT, Vo TK, Huynh BTT (2012) Heuristic algorithms for solving the survivable problem in the design of last mile communication networks. In: Proceedings of the 9th IEEE-RIVF international conference on computing and communication technologies, Ho Chi Minh, Vietnam, pp 1–6

Uchoa E (2006) Reduction tests for the prize-collecting Steiner problem. Oper Res Lett 34(4):437–444

Vo TK, Nguyen MT, Huynh BTT (2012) Heuristic algorithms for solving survivability problem in the design of last mile communication network. In: Proceedings of the 4th Asian conference on intelligent information and database systems, Kaohsiung, Taiwan, pp 519–528

Wagner D, Pferschy U, Mutzel P, Raidl GR, Bachhiesl P (2007) A directed cut model for the design of the last mile in real-world fiber optic networks. In: Proceedings of the international network optimization conference 2007, Spa, Belgium, pp 1–6

Wagner D, Raidl GR, Pferschy U, Mutzel P, Bachhiesl P (2007) A multicommodity flow approach for the design of the last mile in real-world fiber optic networks. In: Waldmann KH, Stocker UM (ed) Operations research proceedings, vol 2006, pp 197–202