Memetic Comp.
DOI 10.1007/s12293-015-0177-7

REGULAR RESEARCH PAPER

A survivable design of last mile communication networks using multi-objective genetic algorithms

Lam Thu Bui[1] · **Huynh Thi Thanh Binh**[2]

Received: 15 April 2015 / Accepted: 11 December 2015

Abstract In this paper, we are interested in the survivable network design problem (SNDP) for last mile communication networks called (L-SNDP). Given a connected, weighted, undirected graph G = (V, E); a set of infrastructure nodes and a set of customers C including two customer types where customers in the subset C1 require a single connection (type-1) and customers in the subset C2 need to be redundantly connected (type-2). The aim is to seek a subgraph of G with the smallest weight in which all customers are connected to infrastructure nodes and the connections are protected against failures. This is a NP-hard problem and it has been solved only with the objective of minimizing the network cost. In this paper, we introduce a new multi-objective approach to solve L-SNDP called ML-SNDP. These objectives are to minimize the network cost (total cost) and to minimize the maximal amount of sharing links between connections. Results of computational experiments reported show the efficiency of our proposal.

Keywords Survivable network design · Multi-objective genetic algorithm

1 Introduction

In the recent years, the increasing of communication demands requires more extended networks. Moreover, the standard quality of services must be also higher than ever. In other words, today, the customers such as companies, commercial plazas, stock exchanges, IT centers, etc. require not only fast but also reliable connections. The word "reliable" has many meanings, but one of the most important meanings is the survival ability having at least one back-up connection. It means that if the main connection is down, the network still works fine.

Due to its advantages such as large capacity, small size, light-weighted, security, optic cables have been used gradually widely instead of coaxial one to satisfy customers' requirements for fast connections. Besides, survivable network design problem (SNDP) was also considered to increase reliability for connections. Up to now, SNDP has gained more and more interests and been solved for many different network models.

In this paper, we consider the survivable design of the last mile communication network (L-SNDP), which was stated as a single objective problem by Nguyen et al. [20]. This objective is to minimize the network cost. The authors utilized available links for solving this problem. However, the more links are utilized, the more connections are affected if these link are failed. It causes a non-trivial risk in protecting the network against failures. A proposed method to overcome it will be presented in this paper.

This paper proposes a formulation of the SNDP in the model of fault-tolerant multi-objectivity: minimizing network cost; minimizing the maximal sharing coefficient. The sharing coefficient is a parameter representing for the required number of simultaneous connections. An edge with a high sharing coefficient will have a higher failure risk. If this edge fails, it will be a major influence on the whole network. So, making a fault-tolerant network design has to optimize the construction cost while ensuring the minimizing the sharing coefficient of all links. It is the goal which

✉ Huynh Thi Thanh Binh
binhht@soict.hust.edu.vn

Lam Thu Bui
lam.bui07@gmail.com

[1] Le Quy Don Technical University, Hanoi, Vietnam

[2] Hanoi University of Science and Technology, Hanoi, Vietnam

Published online: 20 January 2016

is interested by this research. In other words, we will solve the survivable design of the last mile communication network with two objectives that are minimizing the network cost and minimizing the maximum sharing coefficient. However, it is impossible to satisfy these two objectives simultaneously because they oppose against each other. The more sharing edges be used in network, the more cost we can save and reduce the total cost of the design. While sharing edges help us save the total cost, they might contain potential problems because if they fail, many clients who are using these edges are influenced. Therefore, there is no completely optimal solution for this multi-objective problem. We propose to find acceptable solutions by applying a multi-objective method.

Firstly, L-SNDP is formulated as a multi-objective design problem (ML-SNDP). Then, we propose a multi-objective approach to solve it. We introduce two new versions of multi-objective genetic algorithms based on NSGA-II. The first one is a multi-objective genetic algorithm with a new individual encoding (complete path encoding—CPE) scheme, called CPE-NSGA-II. The second one is a multi-objective genetic algorithm with the edge list encoding (ELE) scheme proposed by Huynh et al. [25], called ELE-NSGA-II. We carried out experiments on real-world instances, and then compared the results with the single objective (cost) genetic algorithm (GA-EDP) [25], and ELE-NSGA-II (cost value); CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II. To strengthen the design, we also employ a memetic procedure during ecolution cycles. Our experimental results showed the efficiency of the multi-objective approach for solving ML-SNDP.

The rest of this paper is organized as follows. In Sect. 2, we introduce problem formulation for ML-SNDP. Related works is presented in Sect. 3. Section 4 describes the proposed algorithms for solving ML-SNDP. The details of our experiments and the computational results are given in Sect. 5. Section 6 presents conclusion and future works.

2 Problem formulation

We consider the problem of augmenting an existing infrastructure network by additional links (and switches) in order to connect potential customer nodes. There are two types of customers. In type-1, a standard, single link connection is sufficient, while type-2 customers require more reliable connections, ensuring connectivity even when a single link or routing node fails.

Before the ML-SNDP problem can be formally stated, we need some definitions relating to connection constraints.

The infrastructure network includes infrastructure nodes and links. All links belong to the infrastructure network can be used without any costs. Customer nodes represent the customers in our network. The set of customer nodes C is partitioned into two subsets C_1 and C_2, ($C_1 \cup C_2 = C$ and $C_1 \cap C_2 = \varnothing$), the following conditions specify how customer nodes are connected:

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

Definition of the sharing coefficient:
Given graph $G = (V, E)$, connection r is a path from a source node to a destination node, for any e $\in$ E, having:

$$f_e(r) = \begin{cases} 1 & if \quad e \in r \\ 0 & if \quad e \notin r \end{cases} \tag{1}$$

The sharing coefficient of edge e is defined as s_e and calculated by:

$$s_e = \sum_{i=1}^{N} f_e(r_i) \tag{2}$$

where N is the number of connection requires satisfying ML-SNDP ($N = |C|$).
ML-SNDP can be formulated as following:
Input

- A connected, weighted, undirected graph $G = (V, E)$ in which V is a set of vertex representing network nodes, the set of edges $E = \{e|e = (u, v)\}$ with $u, v \in V$.
- Graph $G_1\ (V_1, E_1) \subset G$ represents the infrastructure network with $w\ (e) = 0, \forall e \in E_1$ in which w(e) is weight of edge e.
- A set of customer nodes C ($C \subset V$ and $C \cap V_1 = \varnothing$) is partitioned into subsets C_1, C_2 ($C_1 \cup C_2 = C$ and $C_1 \cap C_2 = \varnothing$) representing sets of type-1 and type-2 customer nodes.

Constraints

- All customer nodes are connected to the infrastructure network G_1.
- Each customer node in C_2 has two disjoint-edge paths to G_1

Objectives:

$$f_1 = totalCost\left(G'\right) \rightarrow \min$$
$$f_2 = \max\left(s_{e_i}\right) \rightarrow \min$$

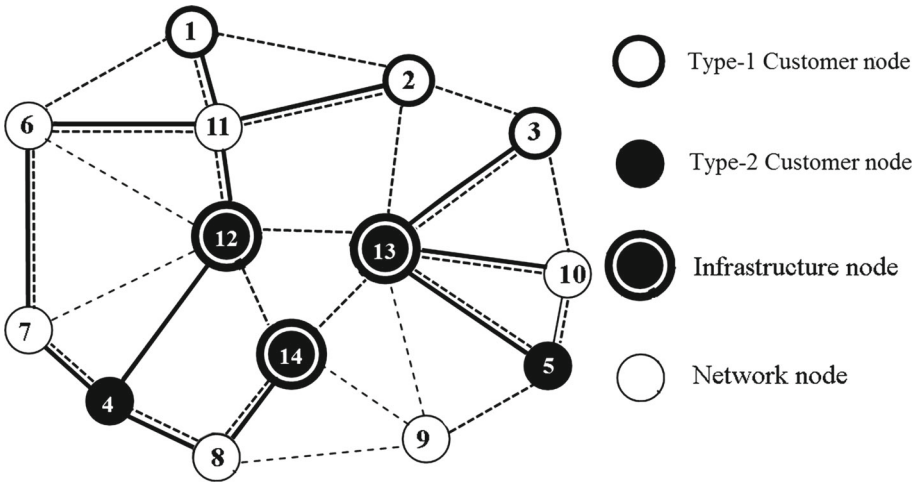

Fig. 1 An example of acceptable solution for ML-SNDP

with

$$G'(V', E') \subset G$$
$$totalCost(G') = \sum_{e \in E'} w(e)$$

where w(e) is weight of edge e. In which s_{ei} is the sharing coefficient of edge

Output

– A sub-graph G′ of G in which all customer nodes are connected to the infra-structure network.

Figure 1 illustrates an example of a survivable design of the last mile communication network. This network is represented by the set V of vertices labeled 1 to 14 and the set E of dash edges, in which the infrastructure network G1 includes 3 nodes (12, 13, 14) and the links among them. The node 1, a type-1 customer, can connect to G1 by the path: $1 \rightarrow 11 \rightarrow 12$. Similarly, the other type-1 customer nodes (node 2, node 3) connect to G1 by the path: $2 \rightarrow 11 \rightarrow 12$ and $3 \rightarrow 13$ respectively. Each of the type-2 customers (node 4, node 5) connects to G1 by two disjoint-edge paths: $\{4 \rightarrow 8 \rightarrow 14, 4 \rightarrow 7 \rightarrow 6 \rightarrow 11 \rightarrow 12\}$ for node 4 and $\{5 \rightarrow 13, 5 \rightarrow 10 \rightarrow 13\}$ for node 5. The infrastructure network G1 along with the set of nodes $\{V - \{9\}\}$ and the set of solid edges, construct a sub-graph of the graph (V, E) which is an acceptable solution. In this solution, the link between node 1 and node 11 is used three times so its sharing coefficient is 3.

This problem is known to be NP-hard for $|C1| > 0$ and $|C2| > 0$ [1]. Therefore, with its multi-objective version, ML-SNDP, we propose a multi-objective genetic method to solve.

3 Related work

Over the years, many research works have been published for SNDP. We can find an overview of SNDP as well as methods for solving it in [1]. According to this survey, SNDP can be approached by two ways: exact methods and heuristic methods.

In general, exact approaches for solving L-SNDP are based on mixed linear integer programming. Wagner et al. [2] modeled this problem as an integer linear program (ILP) by means of an extended multi-commodity network flow (MCF) formulation. Then they used ILP-solver CPLEX [6] (http://www.ilog.com) to give optimal solutions for their model. The largest in-stance that they could solve has 190 total nodes, 377 edges but only 6 customer nodes. With instances up to 2804 nodes, 3082 edges and 12 customer nodes, their approach could solve with a final gap of about 7 %. However, it is unsuitable to use this method for larger instances and/or in particular instances with larger number of customer nodes.

Also with exact approach, Wagner et al. [7] gave a different formulation for L-SNDP that based on directed connectivity constraints. By using a branch-and-cut algorithm, their method could find proven optimal solutions for instances with only 190 nodes, 377 edges, and 13 customer nodes.

Liubíc et al. [9] presented an exact method for the PCSTP (price-collecting Steiner tree problem) based on directed connection cuts. Other successful mathematical programming solutions were based approaches including a relax-and-cut by Da Cunha et al. [10] and a cutting plane method by Lucena et al. [11]. However, being deterministic and exhaustive in nature, these approaches could only be used to solve small problem instances (e.g. sparse graphs with number of customers less than 15).

With ambition to solve larger instances, many researchers proposed heuristic algorithms. Busics and Raidl [8]. used meta-heuristic approaches such as local search and simu-

lated annealing, variable neighborhood descent and variable neighborhood search. This approach has solved and obtained some significant improvements for some problem instances. Leitner (2008) formulated it as an abstract integer linear program and applied Lagrangian decomposition to obtain relatively tight lower bounds as well as feasible solutions. Furthermore, hybrids of a variable neighborhood search and a GRASP are applicable to larger problem instances. Canuto et al. [12] 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. Uchoa [14] has described experiments to reduce the number of nodes and edges that need to be considered in an instance of the PCSTP. Chapovska and Punnen [13] discussed complexity of methods for several variants of the PCSTP. Leitner and Raidl [3] 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 mediums sized instances within reasonable time.

The most recently, Nguyen et al. [20] solved SNDP in the design of the last mile communication network by using heuristic algorithms and gave better results than the other and can apply for larger problem.

Other related problems are the various variants of the SNDP. They can be found in [1,15,16].

Up to now, there have not any research solving this problem with a multi-objective approach, though this is a potential approach in general network design as indicated in our earlier work [25] Note that using MOEAs for design problems has been a popular topic in [26,27].

This paper proposes SNDP in the model of fault-tolerant multi-objectivity: minimize network cost; minimize the maximal sharing coefficient. This is a NP-hard problem. We use a multi-objective genetic algorithm in order to solve it. The proposed approach is expected to give good results that satisfy both objectives.

4 Proposed methodology

We propose a genetic-based approach to solve ML-SNDP. We applied the non-dominated sorting mechanism of NSGA-II to rank solutions. As a result, there are different versions of MOEAs proposed:

CPE-NSGA-II: genetic algorithm which uses CPE.

ELE-NSGA-II: genetic algorithm which uses ELE.

In order to enhance performance of the algorithms, we employ memetic procedure in two versions.

CPEL-NSGA-II: genetic algorithm which uses CPE and Local Search.

ELEL-NSGA-II: genetic algorithm which uses ELE and Local Search.

4.1 Genetic algorithm with complete path encoding

4.1.1 Individual representation

In CPE, an individual consists of |C| chromosomes. Chromosome i represents a solution for a customer ci's connection demand by a list of nodes in the path from ci to the infrastructure network.

Each customer in C1 requires only one path to the infrastructure network, so the corresponding chromosome has only a gene. In addition, each chromosome corresponding to a customer in C2 has two genes that represent two disjoint-edge paths, one of which is the working path and the other is the backup path.

Figure 2 depicts an individual representation with |C| = 5, labeled from 1 to 5, in which number 1, 2 and 3 represent C1 = {c11, c12, c13}; number 4 and 5 represent C2 = {c21, c22}. The bold numbers 12, 13 and 14 represent nodes in the infrastructure network. Customer c-21 belongs to C2. It has two genes. The first one is {4, 7, 6, 11, 12} representing the path from c21 to the infrastructure as follows: $4 \rightarrow 7 \rightarrow 6 \rightarrow 11 \rightarrow 12$, called working path, denoted by Wc21; the other one is {4, 8, 14} giving us the path $4 \rightarrow 8 \rightarrow 14$, we call Bc21. These two paths are disjoint-edge. It is the same as the type-2 customer type-2 c22.

4.2 Initial population

We propose a procedure to find a solution randomly.

Algorithm 1: findRandSolution

Input:
Undirected graph G = (V, E)
A set of infrastructure nodes J
A set of client nodes C which is partitioned into 2 subsets C_1, C_2 corresponding to each client type.
Output:
A set of paths S which satisfies all client requirements.

```
Begin
1. while (|C| > 0)
2.     c = randomSelect(C);
3.     if (c ∈ C1)
4.     p = findPathDijkstra(c, G);
5.     S = S ∪ p
6.     reweight(G, p, cost);
7.     if (c ∈ C2)
8.         <p1, p2> = findPathType2(c, G);
9.     S = S ∪ {p1, p2}
10.        reweight(G, p1, c);
11.        reweight(G, p2, c);
12. endwhile
13. reset(G);
14. return S;
End
```

c11	1	11	12		
c12	2	11	12		
c13	3	13			
c21	4	7	6	11	12
	4	8	14		
c22	5	13			
	5	10	13		

Fig. 2 An example of individual represented by CPE

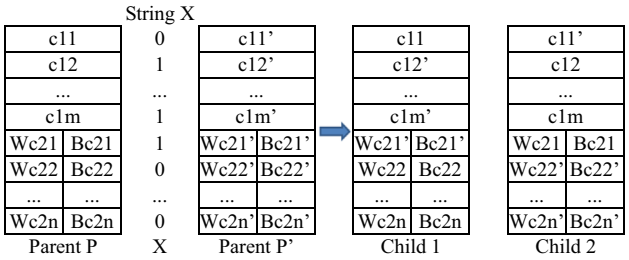

Fig. 3 Path crossover operator

randomSelect(C) randomly selects a client from C and also removes it from C.
findPathDijkstra(c, G) find the shortest path from C to infrastructure nodes of G by Dijkstra algorithm.
reweight(G, p, cost) reweight the cost for all edges that appear in the path p. There are two ways to reweight the cost:

Set it to be 0 (encourages next clients use this path that leads to reduce the total cost)

Set its value to be equal to the edge which has maximal cost (avoid using these edges multiple times to reduce sharing edges).
findPathType2(c, G) find the two disjoint-edge paths for client type 2c by two methods:
Method 1: finds the first path p1 by *findPathDijkstra(c, G)*, removes all edge on the found path and reuse *findPathDijkstra(c, G)* again to find the second path p2.
Method 2: finds the disjoint-edge shortest pair by applying Suurballe algorithm [23].

By combining the ways to *randomSelect, reweight* and *findPathType2* we can reach multiple ways to find a random solution. To build the initial population we use the *findRandSolution* method to find unique solutions.

4.2.1 Crossover operator

Use path crossover to create new individuals. The main idea is that: choose two individuals in the current population randomly and initialize a random binary string X having |C| bits. If Xi is 0, copy chromosome ci of the parent 1 and the parent 2 to child 1 and child 2 respectively. If Xi is 1, do inversely. After the children are created, check if it is unique or not.

Figure 3 depicts path crossover operator. Parent P copies its chromosome C11, C22 and C2n to child 1 as the corresponding bits on the string X are 0. The remaining chromosomes of parent P are copied to child 2. Inversely, parent P′ copies its chromosome C11′, C22′ and C2n′ to child 2, its remaining chromosomes are copied to child 1.

4.2.2 Mutation operator

Choose an individual and its chromosome randomly, remove all edges in the corresponding path from the input graph, then find a new path and replace the chosen chromosome.

For type-1 customers, we find a new shortest path to the terminal nodes. For type-2 customers, find a new pair of paths to the terminal nodes using edge disjoint shortest pair algorithm.

Mutate method : mutate
Input:
Undirected graph G = (V, E)
A solution S
Output: another solution X
begin
1. c = **randomSelectClient**(S);
2. **removeEdgesRandomly**(G, c);
3. X = **findRandSolution**(G)
4. **reset**(G)
5. **return** X
end

4.2.3 Local search operator

Mutate method using localsearch : mutatels
Input:
Undirected graph G = (V, E)
A solution S
Output: another solution X
begin
1. X = S
2. T = **mutate**(G, S)
3. ncount = 0
3. **while** (ncount < 10)
4. **if** T.cost < X.cost
5. X = T
6. ncount = 0
6. **else**
7. ncount++
8. T = **mutate**(G, S)
9. **reset**(G);
10. **return** X;
end

4.3 Genetic algorithm with edge list encoding

Edge list encoding (ELE) is the method proposed by Huynh and Duong [22]. In this encoding scheme, an individual is represented by an edge list which contains all edges used by connections satisfying all customers' requirements.

Initial population: use findRandSolution procedure to generate the initial population. This procedure creates a graph G′ using all edges and vertexes of G. Then we set a random

Table 1 Problem instances

Data set	|V|	|E|	|C1|	|C2|
ClgSExtra-I2-08.ist	190	377	8	7
ClgSExtra-I2-11.ist	190	377	11	6
ClgSExtra-I2-12.ist	190	377	9	7
ClgSExtra-I3-07.ist	190	377	6	6
ClgMExtra-08.ist	1757	3877	5	3
ClgMExtra-I2-01.ist	1523	3290	8	4
ClgMExtra-I2-04.ist	1523	3290	10	4
ClgMExtra-I2-11.ist	1523	3290	10	4
ClgN1ExtraI1-07.ist	3867	8477	5	8
ClgN1ExtraI1-10.ist	3867	8477	3	9
ClgN1ExtraI1-12.ist	3867	8477	4	8
ClgN1ExtraI1-19.ist	3867	8477	4	8

weight for each edge of G′. Finally, apply H-EDP algorithm on G′, we find one new solution.

Crossover operator: from two individuals, a new graph is created containing two parent's edges, which we call merged graph. Then we find two possible solutions from this graph. The first one is found by applying H-EDP algorithm on merged graph. We get the other individual by using Find-RandSolution procedure as present above on the merged graph.

Mutation operator: randomly remove some edges which are used by selected solution from the inputted graph, and use H-EDP to find new solution.

5 Computational results

5.1 Problem instances

The problem instances used in our experiments are the real-world instances which can be found at https://www.ads.tuwien.ac.at/people/mleitner/sndp/sndpinstances.tar.gz. The real-world instances are data from a German city and are used in [3–5], [18–20]. These instances also used by Huynh and Duong [22]. Their sizes are reported in Table 1.

5.2 System setting and parameters

We experimented four multi-objective genetic algorithms for solving ML-SNDP: CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II. The results found by these algorithms are compared with each other and with the single objective (cost) genetic algorithm (GA-EDP) [22].

In the experiments, the system was run ten times with different random seeds for each problem instance with parameters: number of individuals: 500, number of generations: 100, crossover probability: 50 % mutate probability: 20 %.

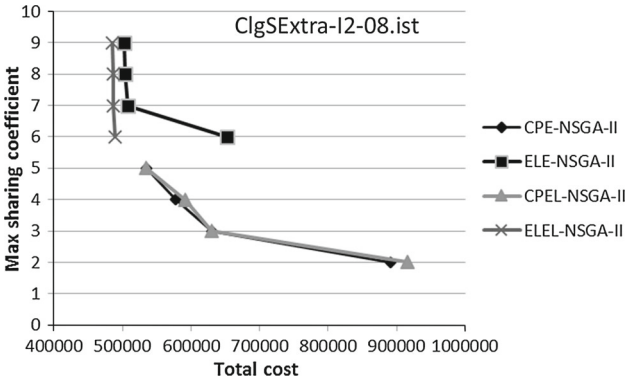

Fig. 4 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgSExtra-I2-08.ist

All the programs were ran on a machine with Intel Core i5 2500 K @4.5 GHz, 8 GB RAM @2133 MHz, Windows 7 Ultimate. CPE-NSGA-II implemented in C++ language, ELE-NSGA-II implemented in Java language.

5.3 Computational results

5.3.1 The effect of representation schemes

Considering the effect of the different encoding schemes, we compare the result of CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II in the same computing environment. In order to do this, we recorded the comparison-solutions found by each algorithm and make comparision on these solutions.

The figures below represent the final optimal Pareto boundaries for test sets. The vertical axis represents sharing coefficient, and horizontal axis represents total cost value.

Figures 4, 5, 6 and 7 shows objective values of the two-objective of individual found by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II after 100 generations on four test sets: ClgSExtra-I2-08, ClgSExtra-I2-11, ClgSExtra-I2-12 and ClgSExtra-I3-07. The spread of pareto set indicate that we cannot optimize both two objectives. If we use less sharing edge on design, we must pay more extra cost, and reversely.

In comparing two algorithms we used, ELE-NSGA-II showed a bit more effective on total cost objective while CPE-NSGA-II showed more effective on sharing coefficient. However, with the same sharing coefficient, we pay less cost for CPE-NSGA-II than ELE-NSGA-II, the result of two sets ClgSExtra-I2-11 and ClgSExtra-I3-07 is a clear proof. Algorithms which intensified by localsearch, show more efficient results than others.

Figures 8, 9, 10 and 11 show the objective values of the two-objective of individual found by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II after 100

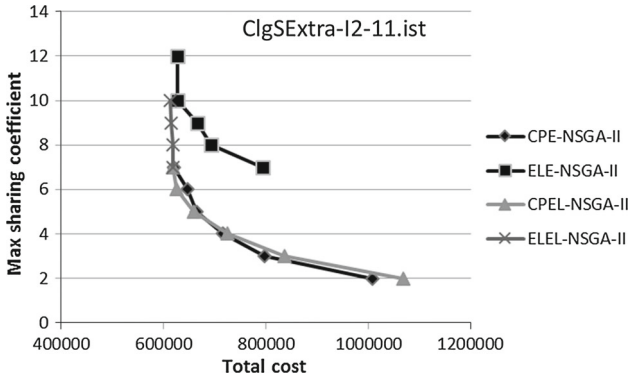

Fig. 5 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgSExtra-I2-11.ist

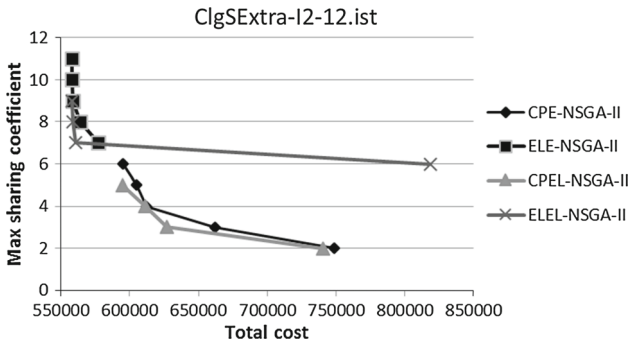

Fig. 6 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgSExtra-I2-12.ist

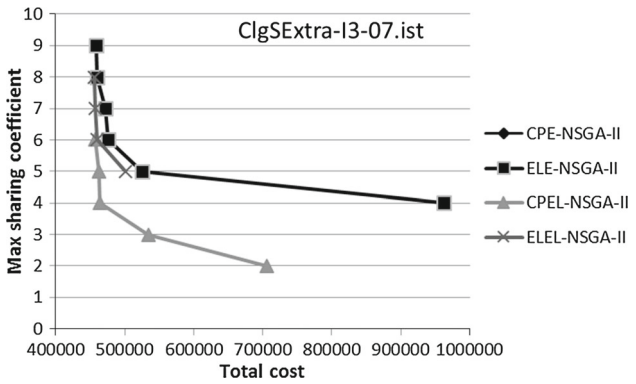

Fig. 7 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgSExtra-I3-07.ist

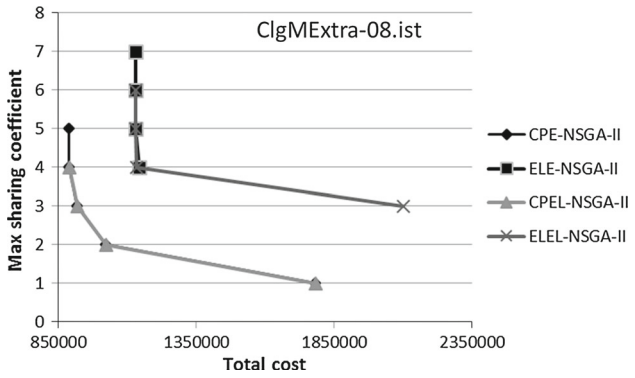

Fig. 8 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgMExtra-08.ist

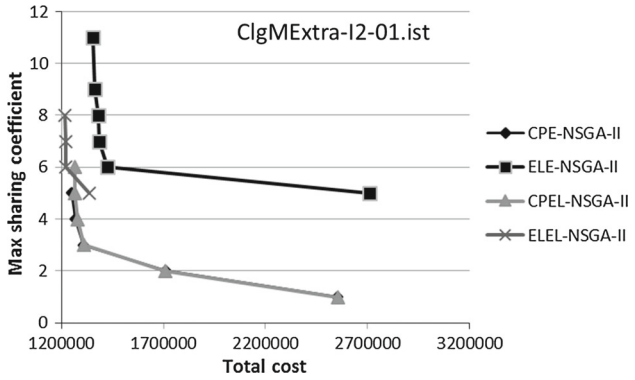

Fig. 9 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgMExtra-I2-01.ist

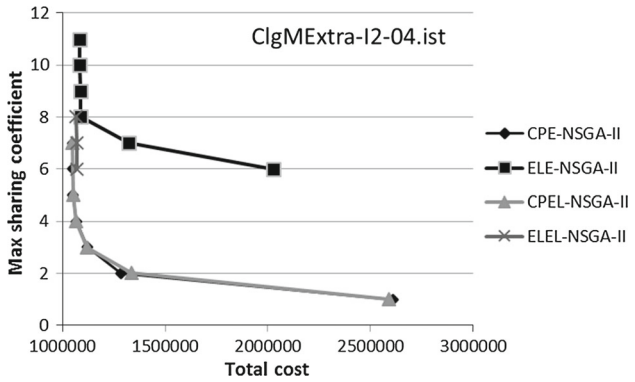

Fig. 10 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgMExtra-I2-04.ist

generations on four test sets: ClgMExtra-08, ClgMExtra-I2-01, ClgMExtra-I2-04 and ClgMExtra-I2-11.

CPE-NSGA-II showed more effective than ELE-NSGA-II for both objective, but in the last test, the set ELE found less cost solution than CPE-SNGA2. In all four testing sets, total cost of CPE-NSGA-II show very far better than ELE-NSGA-II, especially when lower sharing coefficient (Figs. 12, 13, 14, 15).

To have a better view about the effect of two approaches, we use another metric to compare the result of CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II. Two set coverage (SC) is an effective way to estimate how much a population better than another [24]. This metric is easy to calculate and provides a relative comparison based upon dominance numbers between populations. The SC

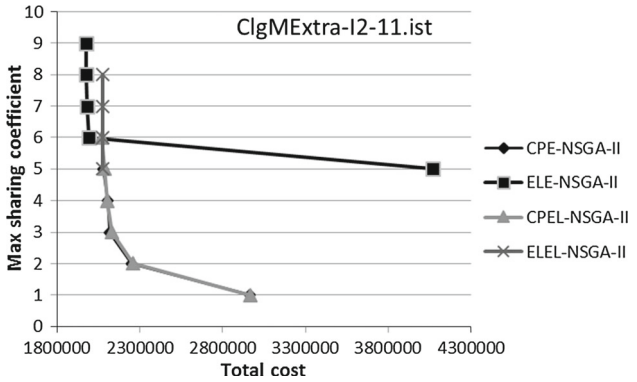

Fig. 11 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgMExtra-I2-11.ist

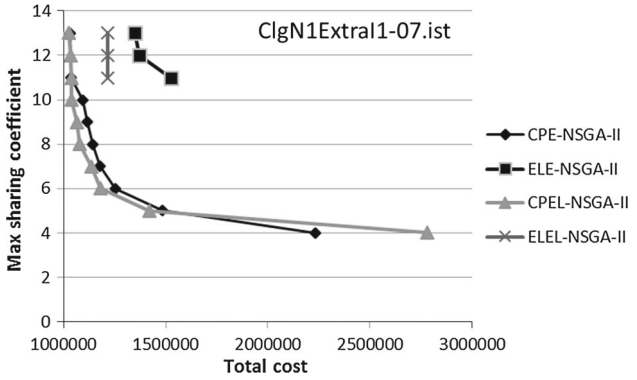

Fig. 12 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgN1ExtraI1-07.ist

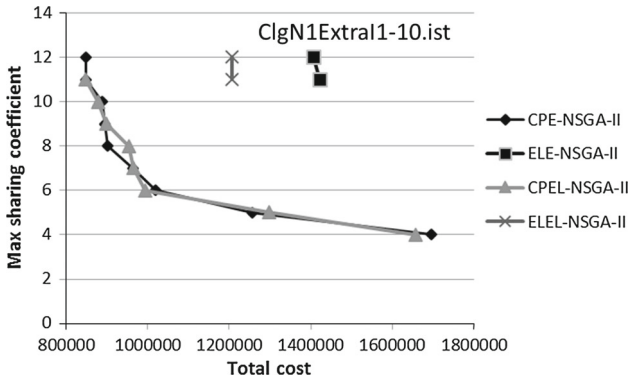

Fig. 13 The non-dominated solutions obtained by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgN1ExtraI1-10.ist

between the last populations of CPE-NSGA-II (PCPE) and ELE-NSGA-II (PELE) are shown in Table 2.

Except set ClgSExtra-I2-12, almost individuals of CPE-NSGA-II can dominate an individual of ELE-NSGA-II. CPE-NSGA-II can give us better solutions than ELE-NSGA-II.

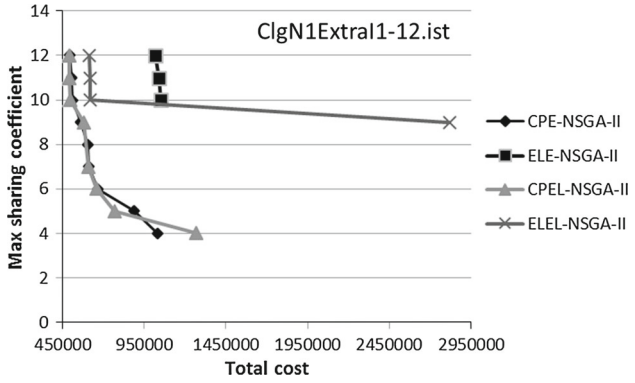

Fig. 14 The non-dominated solutions compared by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgN1ExtraI1-12.ist

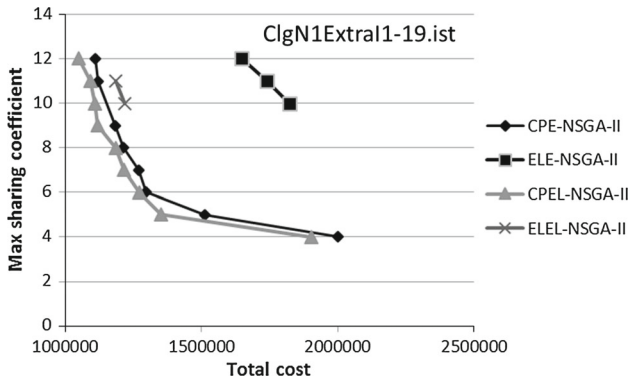

Fig. 15 The non-dominated solutions compared by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II on test set ClgN1ExtraI1-19.ist

Table 2 Set coverage results

Set	SC (P_{CPE}/P_{ELE})	SC (P_{ELE}/P_{CPE})
ClgSExtra-I2-08.ist	0.878	0.000
ClgSExtra-I2-11.ist	0.956	0.070
ClgSExtra-I2-12.ist	0.170	0.070
ClgSExtra-I3-07.ist	1.000	0.000
ClgMExtra-08.ist	1.000	0.000
ClgMExtra-I2-01.ist	1.000	0.000
ClgMExtra-I2-04.ist	1.000	0.000
ClgMExtra-I2-11.ist	1.000	0.000
ClgN1ExtraI1-07.ist	1.000	0.000
ClgN1ExtraI1-10.ist	1.000	0.000
ClgN1ExtraI1-12.ist	1.000	0.000
ClgN1ExtraI1-19.ist	1.000	0.000

5.3.2 The evolution of CPE-NSGA-II

On this experiment, we focus on the evolution of CPE-NSGA-II algorithm, especially how the populations evolve

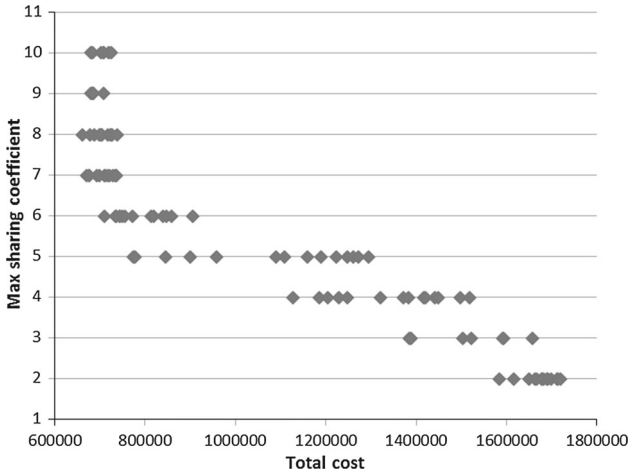

Fig. 16 The initial population of ClgSExtra-I2-11.ist

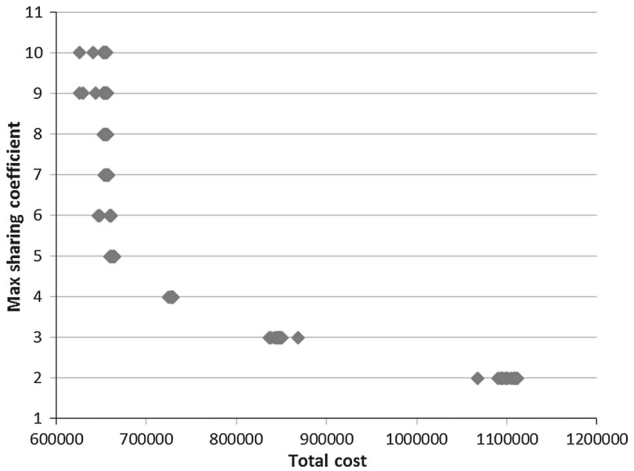

Fig. 17 The final population of ClgSExtra-I2-11.ist

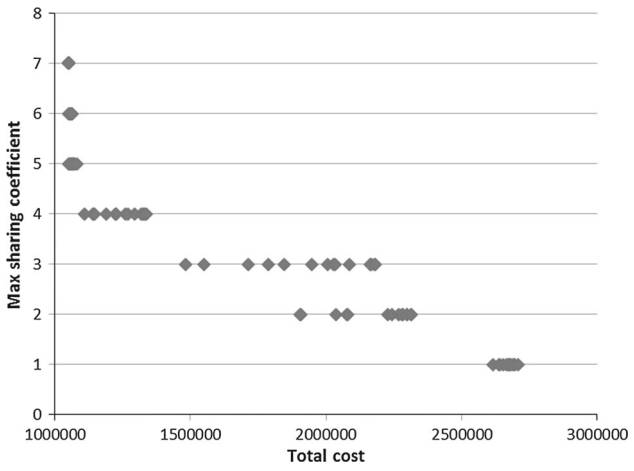

Fig. 18 The initial population of ClgMExtra-I2-04.ist

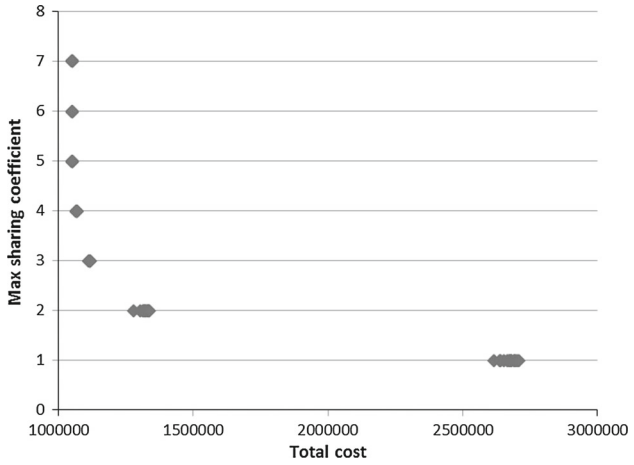

Fig. 19 The final population of ClgMExtra-I2-04.ist

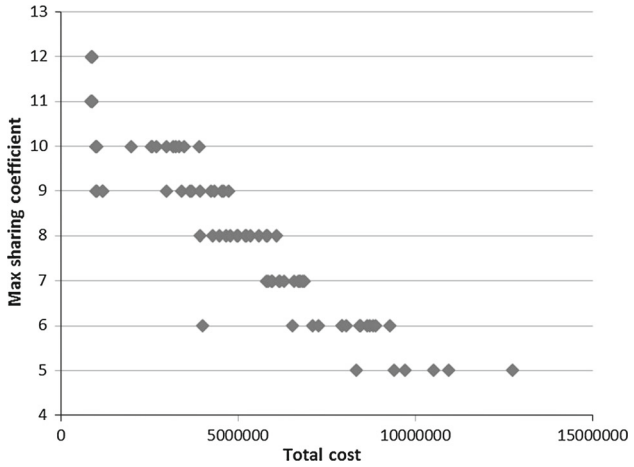

Fig. 20 The initial population of ClgN1ExtraI1-10.ist

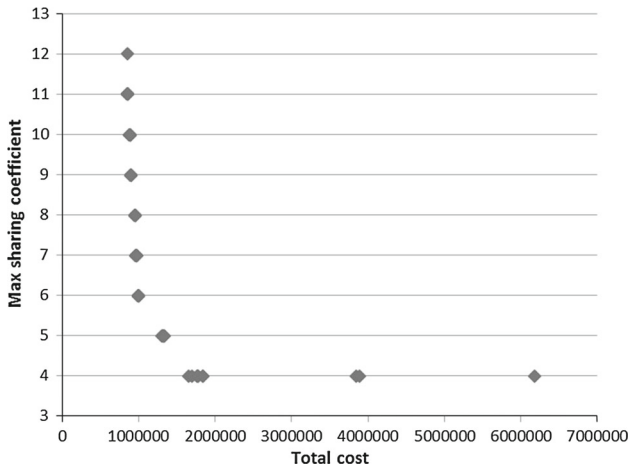

Fig. 21 The final population of ClgN1ExtraI1-10.ist

throughout generations and how quickly they reach the stable state?

Average runtime:

On small data sets: 414 s
On medium data sets: 10,526 s
On large data sets: 39,794 s

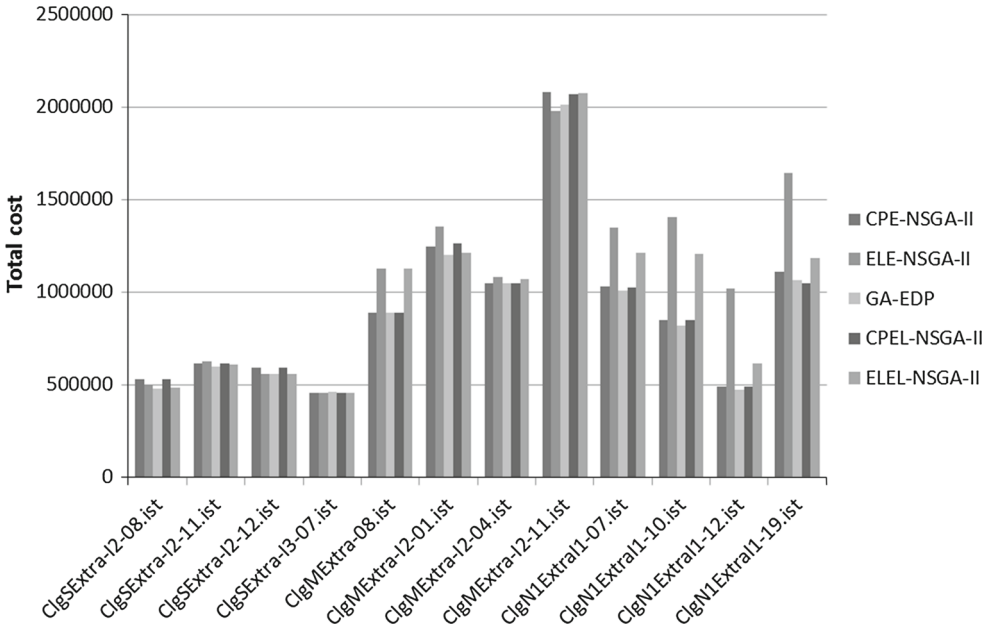

Fig. 22 Comparisons between the best costs found by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II, ELEL-NSGA-II and GA-EDP

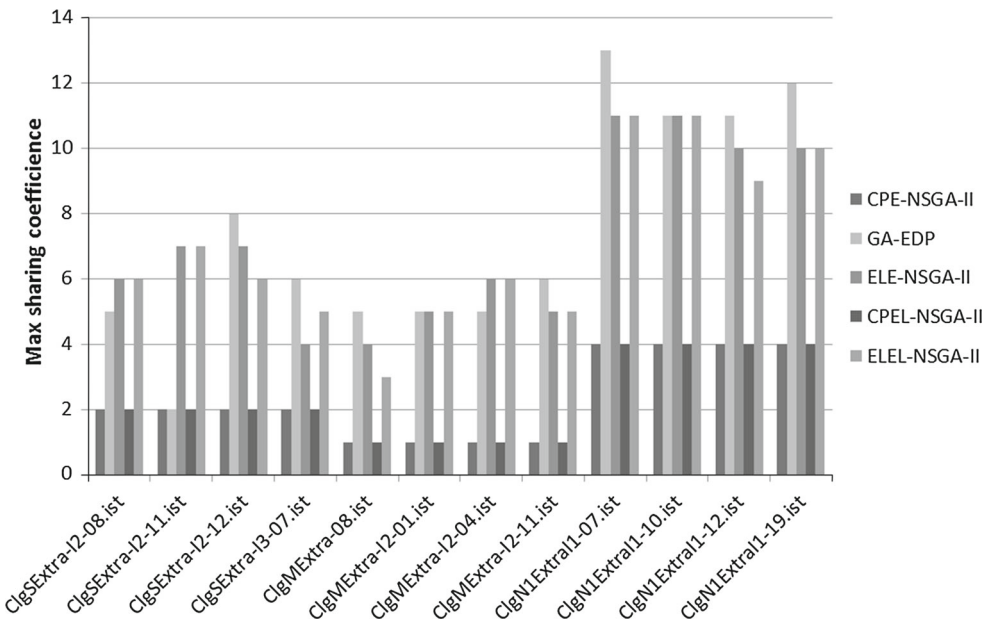

Fig. 23 Comparisons between the best max sharing coefficient found by CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II, ELEL-NSGA-II and GA-EDP

The evolution process:
On small data sets, the population evolves quickly to stable state after about 60 generations (Figs. 16, 17).

On medium data sets, the population evolves quickly to stable state after about 80 generations (Figs. 18, 19).

On large data sets, the population evolves quickly to stable state after about 170 generations (Figs. 20, 21).

5.3.3 The effect of multi-objectivity

The motivation for us is that, network design problem is popular in addressing a single objective, but in fact, we usually have to deal with two objectives. So, in any case, we have to take into account multi-objectivity when designing a network. So the focus is more on problem formulation

and adjusting a multi-objective evolutionary algorithm for it. Hence there is a need to compare the results between this and the single objective approach.

Figure 22 shows the comparison between the best costs found by CPE-NSGA-II, ELE-NSGA-II and GA-EDP [22].

The cost found by GA-EDP is the best on almost problem instances compare to CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II. The reason that CPE-NSGA-II, ELE-NSGA-II, CPEL-NSGA-II and ELEL-NSGA-II have to chase two objectives at the same time, they could not focus on total cost more than GA-EDP. In the small test sets like ClgSExtra-series, ELE-NSGA-II shows the better cost than CPE-NSGA-II. When we increased the size of test sets, ELE-NSGA-II lose its cost advantage compare with CPE-NSGA-II. The total costs of CPE-NSGA-II are better and better than ELE-NSGA-II when the sets become larger. However, in the set ClgMExtra-I2-11, ELE-NSGA-II shows the best result, better than GA-EDP. This can be explained by the factor of mutation in the genetic algorithm (Fig. 23).

Although ELE-NSGA2-II based on GA-EDP which is the most cost-effective, it shows in effective on large test sets with the worst cost and longest run time. CPE-NSGA-II shows the best stability algorithm for both the two objectives and run time. Compares with the result of GA-EDP, CPE-NSGA-II's best cost result is slightly different in some requires but ELE-NSGA-II' best cost result is very different.

6 Conclusion

In this paper, we proposed the multi-objective genetic algorithm for solving ML-SNDP. We experimented on 12 real world instances. With each data set, we run 10 times to take the best solutions. The results show that our proposed approaches are effective. We found that a good network design has not only has minimal cost, but also to be able to minimize risks such as failure on edges (links), on nodes. The failure on nodes rarely occurs because each node usually has many servers, when one is failed, the others can do instead. The failure on edges (links) is more complicated and difficult to control as well as guarantee reliable. When failures occur, it is difficult to determine the failed point to repair in the short time.

In the future, we will add a new objective, constructing a model allowing quantifying survivable coefficient of edges, nodes that can help to find better network design solutions.

Acknowledgments This work was supported by the project "Advanced methods in Evolutionary Computation in approximating solutions for combinatorial optimization problems", Grant no. 102.01-2015.12, funded by National Foundation for Science and Technology Development, Vietnam.

References

1. Kerivin H, Mahjoub AR (2005) Design of survivable networks: a survey. Networks 46(1):1–21
2. Wagner D, Raidl GR, Pferschy U, Mutzel P, Bachhiesl P (2007) A multi-commodity flow approach for the design of the last mile in real-world fiber optic networks. In: Waldmann KH, Stocker UM (eds) Operations research proceedings 2006, pp 197–202
3. 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. (eds) Hybrid metaheuristics, vol 5296. Springer, Berlin, Heidelberg, pp 158–174
4. Leitner M, Raidl GR (2010) Branch-and-cut and price for capacitated connected facility location. Technical report TR-186-1-10-01, Vienna University of Technology, Vienna, Austria
5. Leitner M, Raidl GR (2010) Strong lower bounds for a survivable network design problem. In: International symposium on combinatorial optimization, Hammamet, Tunisia, pp 295–302
6. IBM ILOG (2006) CPLEX optimizer performance benchmarks 10.0. http://www.ilog.com
7. 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
8. Bucsics T, Raidl G (2007) Metaheuristic approaches for designing survivable fiber-optic networks. In: Institute for computer graphics and algorithms of the Vienna University of Technology
9. Ljubic 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
10. 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
11. Lucena A, Resende MGC (2004) Strong lower bounds for the prize collecting Steiner problem in graphs. Discrete Appl Math 141(1–3):277–294
12. Canuto SA, Resende MGC, Ribeiro CC (2001) Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks 38:50–58
13. Chapovska O, Punnen AP (2006) Variations of the prize-collecting Steiner tree problem. Networks 47(4):199–205
14. Uchoa E (2006) Reduction tests for the prize-collecting Steiner problem. Op Res Lett 34(4):437–444
15. Fortz B, Labbe M (2006) Polyhedral approaches to the design of survivable networks. In: Resende MGC, Pardolas PM (eds) Handbook of optimization in telecommunications. Springer, Berlin, pp 367–389
16. Stoer M (1992) Design of survivable networks. LNCS 1531. Springer, Heidelberg
17. 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
18. 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 Institute, Department of Telematics and Network Engineering, Klagenfurt, Austria
19. 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
20. Nguyen MT, Vo KT, Huynh BTT (2012) Heuristic Algorithms for Solving the Survivable Problem in the Design of Last Mile Com-

munication Networks. In: Proceedings of the 9th IEEE—RIVF international conference on computing and communication technologies, Ho Chi Minh, Vietnam, pp 219–224
21. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
22. Huynh Thi Thanh Binh, Nguyen Thai Duong (2015) Heuristic and genetic algorithms for solving survivability problem in the design of last mile communication networks. Soft Computing 19:2619–2632
23. Suurballe JW, Tarjan RE (1984) A quick method for finding shortest pairs of disjoint paths. Networks 14:325–336
24. Coello CAC, Sierra MR (2003) A multi-objective evolutionary algorithm based on coevolutionary concepts. In: The 2003 congress on evolutionary computation, vol 1, pp 482–489
25. Binh HTT, Bui LT, Ha NST, Ishibuchi H (2014) A multi-objective approach for solving the survivable network design problem with simultaneous unicast and anycast flows. Appl Soft Comput 24:1145–1154
26. Farnsworth M, Benkhelifa E, Tiwari A, Zhu M, Moniri M (2011) An efficient evolutionary multi-objective framework for MEMS design optimisation: validation, comparison and analysis. Memet Comput 3(3):175–197
27. Joshi R, Deshpande B (2014) Empirical and analytical study of many-objective optimization problems: analysing distribution of nondominated solutions and population size for scalability of randomized heuristics. Memet Comput 6(2):133–145