

Two-level evolutionary approach to the survivable mesh-based transport network topological design

J. Sun · Q. Zhang · J. Li

Received: 15 August 2007 / Revised: 6 July 2009 / Accepted: 9 October 2009 /
Published online: 29 October 2009
© Springer Science+Business Media, LLC 2009

Abstract The complete topology design problem of survivable mesh-based transport networks is to address simultaneously design of network topology, working path routing, and spare capacity allocation based on span-restoration. Each constituent problem in the complete design problem could be formulated as an Integer Programming (IP) and is proved to be \mathcal{NP} -hard. Due to a large amount of decision variables and constraints involved in the IP formulation, to solve the problem directly by exact algorithms (e.g. branch-and-bound) would be impractical if not impossible. In this paper, we present a two-level evolutionary approach to address the complete topology design problem. In the low-level, two parameterized greedy heuristics are developed to jointly construct feasible solutions (i.e., closed graph topologies satisfying all the mesh-based network survivable constraints) of the complete problem. Unlike existing “zoom-in”-based heuristics in which subsets of the constraints are considered, the proposed heuristics take all constraints into account. An estimation of distribution algorithm works on the top of the heuristics to tune the control parameters. As a result, optimal solution to the considered problem is more likely to be constructed from the heuristics with the optimal control parameters. The proposed algorithm is evaluated experimentally in comparison with the latest heuristics based on the IP software CPLEX, and the “zoom-in”-based approach on 28 test networks problems.

J. Sun (✉)

CPIB, School of Bioscience, University of Nottingham, Sutton Bonington, LE12 5RD, UK
e-mail: j.sun@cpib.ac.uk

Q. Zhang

Department of Computer Science, University of Essex, Wivenhoe Park, Essex, CO4 3SQ, UK
e-mail: qzhang@essex.ac.uk

J. Li

Unilever Safety and Environmental Assurance Center, Colworth Park, Sharnbrook, Bedfordshire, MK44 1LQ, UK
e-mail: j.li@unilever.com

The experimental results demonstrate that the proposed algorithm is more effective in finding high-quality topologies than the IP-based heuristic algorithm in 21 out of 28 test instances with much less computational costs, and performs significantly better than the “zoom-in”-based approach in 19 instances with the same computational costs.

Keywords Survivable mesh-based network · Network topology design · Heuristics · Evolutionary algorithm · Estimation of distribution algorithm

1 Introduction

Mesh-based networks are widely viewed as the kernel network topology for the next generation networking based on DWDM technology (Grover and Doucette 2001; Mukherjee 2000). The mesh-survivable network topological design has attracted much attention since the advent of Sonet and DWDM mesh networks. The survivable network topological design for a mesh-based network mainly involves three tasks. Firstly, a sparse tree-like network topology should be designed for minimal link construction cost. Secondly, working traffic demands need to be routed on the network for minimal traffic routing cost. Finally, spare capacities should be allocated on a closed (bi-connected) topology for the single-failure protection purpose with minimal spare capacity allocation cost. Notably, each task (network topology design, working traffic routing and spare capacity allocation) could be modeled as an integer programming (IP) problem, which is known to be \mathcal{NP} -hard (Kershenbaum 1993; Sakauchi et al. 1990). As pointed out in Grover and Doucette (2001), whilst there are many studies focusing primarily on solution to only one of the problems (please see (Grover and Doucette 2001) for a detailed survey), there is little research that has been done to address the three tasks simultaneously. The issue to tackle all the tasks at the same time is known as *restorable mesh-based topological design problem* as named in Grover and Doucette (2001). Such an integrated complete problem almost always poses to be more difficult than any of its constituent task, because it has to make simultaneous decisions explicitly on network topology, its associated working path routing and spare capacity placement.

The IP formulation in Grover and Doucette (2001) involves a large amount of decision variables and constraints. The decision variables' size is in an order of $\mathcal{O}(M^4 + M^2T)$ where M is the size of the network, and T is the size of traffic demands. The size of the constraints is in an order of $\mathcal{O}(M^3 + T^2)$. It can be seen that even for a small-size network, the size of the complete design problem could be very big. It becomes impractical to apply directly the exact algorithms, e.g. branch and bound, on the network design problems with large sizes. Alternatively, thanks to the developing of evolutionary algorithms (EAs), we may apply EAs to solve the problem. However, it again would be impractical, if not impossible, to apply the EAs if the decision variables of the IP formulation are regarded as the solution's representation in the EAs due to the prohibitively large number of variables. In this paper, we propose a two-level algorithmic approach (Zhang et al. 2007) to handle the difficulty. In the lower level, two parameterized greedy heuristics, namely *the working traffic*

router and *the spare capacity allocator*, are developed to jointly construct a feasible solution (i.e., a closed-graph topology with routing for working traffic demands and spare capacity assignments). The working traffic router aims to find the network topology and the working traffic routing scheme. The spare capacity allocator refines the network topology and assigns spare capacities for survivability of the network. The constraints of the complete problem are taken into consideration in both the greedy heuristics. The two greedy heuristics work in a sequential order to build a feasible solution: the output of the working path router is the input of the spare capacity allocator. Since there is no explicit mathematical model to describe the relationship between the two heuristics, in the upper level, an estimation of distribution algorithm (EDA) is used to search for optimal parameters of the greedy heuristics. By using the optimal parameters found in the upper level, corresponding optimal solution to the complete problem is more likely to be found.

To test the performance of the proposed algorithm, we compare the proposed algorithm with the heuristic developed in Grover and Doucette (2001) (a heuristic based on the exact algorithms), and a zoom-in approach presented in Pickavet and Demeester (2000) on a set of 28 test network instances. The experimental results showed that the proposed algorithm can find better solutions in 21 out of the 28 test networks with significantly less computational cost than the heuristic in Grover and Doucette (2001), and in 19 test networks than the zoom-in approach with the same computational costs.

The rest of the paper is organized as follows. Section 2 presents the related work for the network optimization problems by using heuristics and meta-heuristics. Section 3 describes the considered complete topology design problem. The parameterized greedy heuristics for feasible solution construction to the complete network design problem are presented in Sect. 4. Section 5 describes the top-level estimation of distribution algorithm (EDA) and the proposed algorithm in a whole. The comparison results between the proposed algorithm and the latest heuristics proposed in Grover and Doucette (2001) and Pickavet and Demeester (2000) are given in Sect. 6. Section 7 concludes the paper.

2 Related work

Many algorithms, including exact methods, heuristics methods and stochastic global search methods, have been applied to the network topology design problems. They aim to ensure the designs to be survivable, restorable, and reliable. Previous heuristic approaches are mainly based on graph perturbation. These algorithms start with a feasible topology and perturb the topology with basic modifications (addition, deletion or exchange links) such as Branch Exchange method (BXC) (Kershenbaum 1993), Cut-Saturation Algorithm (CSA) (Chou et al. 1974; Boorstyn and Frank 1977), MENTOR (Kershenbaum et al. 1991), and so on. Exact methods include Lagrangian relaxation, gradient optimization procedure (Gavish 1992), and branch-and-bound algorithm (Jan 1993). However, both heuristics and exact methods aforementioned have their own disadvantages: the heuristic approaches are more likely to be trapped in local optima, whilst the exact approaches are only suitable to small or medium

size problems. Due to the weaknesses of both approaches, as well as strengths and ever increasing popularity of meta-heuristic approaches, recently, we have witnessed many meta-heuristics being applied to the survivable network optimization problems.

In Randall et al. (2002), a simulated annealing (SA) algorithm was applied to design both a computer communication network and a thrifty single failure protected network. A recent paper (Taheri and Zomaya 2007) presents an application of SA to the mobile network design problem. Tabu search (TS) (Glover and Laguna 1998) was applied to the computer network design (Shen et al. 2005), the LAN design (Pierre and Elgibaoui 1997), the IP networks designs (Wille et al. 2005), and many others. In Gil et al. (2006), TS, SA and evolutionary algorithms (EAs) are applied to the network partitioning problem, and compared against each other. Scatter search and path relinking (Glover et al. 2000) techniques were also applied to the network design problems (Álvarez et al. 2003). The greedy randomized adaptive search procedure (GRASP) has been applied to the network optimization problems, including a wide area network backbone design problem (Cancela et al. 2004; Faria et al. 2004) and a power transmission network design problem (Glover et al. 2000). A number of papers have been published on the network design problems by means of genetic algorithms (GAs), such as restorable networks (Al-Rumaih et al. 2000; Pickavet and Demeester 2000), IP networks (Buriol et al. 2007; Wille et al. 2005), wireless network design (Hsu et al. 2008), and so on. Memetic algorithms (Corne et al. 1999) that hybridize local search methods with evolutionary algorithms, have also been adopted to address the design of communication networks (Runggeratigul 2004), wireless sensor network (Ferentinos and Tsiligindis 2007), and so on. Ant colony optimization (ACO) (Dorigo et al. 1996), one of the probability-based algorithms, was mainly applied to routing problems (Anticona and Villegas 2007; Oubutra et al. 2005).

Among all these approaches for the network design problem, few of them dealt with the whole problem: to our best knowledge, the complete survivable network design problem is explicitly considered only in Cinkler et al. (2000), Grover and Doucette (2001), Pickavet and Demeester (2000). In Cinkler et al. (2000), randomized heuristic algorithms, including simulated annealing, tabu search and threshold accepting, were developed, in which the traffic demands are routed by shortest path algorithms with randomized alternative allocating and releasing resources for demands of node-pairs. One of the obvious drawbacks is that the proposed algorithms can be easily get stuck in local optima.

Pickavet and Demeester (2000) attempted to solve the complete problem using the so-called “zoom-in” approach. In the approach, the solution of the full problem is obtained through four phases gradually from rough to fine solution. In the first stage, a genetic algorithm is applied to find a rough network topology based on a simplified problem formulation. Some constraints are not considered in the simplified problem formulation. In the second phase, the network topology previously obtained is improved by a hill climbing algorithm, which is also based on the simplified problem formulation. In the third and fourth phases, simple heuristics are applied to locally improve the network topology (phase 3) and route the working and space capacity assignment (phase 4) based on exact problem formulation. The proposed algorithm can produce near-optimal solutions using modest requirements of time and memory

as claimed in Pickavet and Demeester (2000). However, since the finding and refining of the basic network topology are based on the simplified problem formulation where some detailed constraints are ignored, the overall solution quality may be deteriorated for some specific problem instances.

In Grover and Doucette (2001), the complete problem is named as *mesh topology, routing and spare capacity* (MTRS). A 0/1 IP formulation of the full MTRS is first set up. To solve the problem, a three-step approximate solution method was applied based on a specific hypothesis about the underlying structure of the MTRS problem. In the steps, unlike the zoom-in approach in which algorithmic search is applied, Grover et al. attempted to solve the decomposition of the topology, routing and sparing problems using exact algorithms which are available in the commercial software CPLEX. In the first step, step “W1”, the working-only fixed charge plus routing problem (FCR) is solved to find a rough topology graph. In the second step, step “S2”, an artificial problem, reserved network fixed charge plus spare capacity problem (RN-FCS), is solved to find some extra links for a restorable network topology. Finally, the restricted MTRS problem in which only links found in the first two steps are considered as the potential links, is solved in step “J3”. For details of the heuristic please refer to Grover and Doucette (2001). Since the exact algorithms are adopted, the heuristic is thus very time-consuming for large-size problem instances, and its applicability is limited to small and medium-size instances. But the heuristic may produce better solutions than that of the zoom-in approach (Pickavet and Demeester 2000) since it can guarantee exact solutions at each step.

In this paper, we will present a different approach to the zoom-in strategy. In our approach, a feasible solution to the complete MTRS problem is constructed in two steps, while the complete problem rather than the simplified subproblems in both the two steps is taken into account. Parameterized greedy heuristics are developed in the steps to construct a feasible solution. The estimation of distribution algorithm is used to tune the control parameters associated with the two greedy heuristics. In terms of solution quality, the proposed algorithm is of great potential to produce better solutions since we explore the whole search space in the steps rather than a subset of the whole search space.

3 Problem description

In this paper, the considered complete mesh¹-based transport network topology design problem can be described as follows:

Given: A weighted graph $\mathcal{G} = (\mathbb{V}, \mathbb{E}, \mathbb{D})$ that represents the network, where \mathbb{V} is the set of nodes with cardinality $M = |\mathbb{V}|$; \mathbb{E} is the set of potential links² among nodes where the cost of \mathbb{E}_{ij} is the Euclidean distance between nodes i and j ; matrix \mathbb{D} represents the set of traffic demands (or flows) on links where \mathbb{D}_{ij} gives the size

¹The term *mesh* here refers to networks where there maybe more than one possible route between node pairs (Grover and Doucette 2001).

²In the rest of the paper, we do not differentiate link, edge and span which is considered to be unidirectional. We also do not differentiate capacity and flow.

of traffic demands from a source node i to a destination node j . Moreover, Ω , the edge-to-capacity cost ratio which means that the fixed charge for establishment of a link (i, j) in the topology graph would be $\mathbb{E}_{ij}\Omega$, and the cost per unit of capacity added to an edge is \mathbb{E}_{ij} , is given.

Find: A closed (bi-connected) topological network with minimal overall network cost. The network cost includes the cost of establishing the links (fixed charge) and the cost of capacities assigned for routing the working traffic demands and spare capacities for restorability of the network with *single failure*.

If we denote ℓ_{ij} as the 0/1 decision variable indicating whether the link (i, j) in the graph exists in the designed network, w_{ij} as the capacities allocated for working demands from node i to j , p_{ij} as the spare capacities allocated on the link (i, j) for protection over *single* edge failure, then the objective of the complete MTRS problem is to minimize (cf. Grover and Doucette 2001):

$$\min \sum_{(i,j) \in \mathbb{E}} \{F_{ij}\ell_{ij} + c_{ij}(w_{ij} + p_{ij})\} \quad (1)$$

where F_{ij} is the fixed cost for establishing a link from node i to j , and c_{ij} is the incremental cost of adding one unit of capacity to link (i, j) . In the remaining of the paper, we assume $F_{ij} = \Omega\mathbb{E}_{ij}$ and $c_{ij} = \mathbb{E}_{ij}$ (as it has been done in Grover and Doucette (2001)). Then, the objective function is written as:

$$\min \sum_{(i,j) \in \mathbb{E}} \{\Omega\mathbb{E}_{ij}\ell_{ij} + \mathbb{E}_{ij}(w_{ij} + p_{ij})\} \quad (2)$$

subject to the following constraints:

1. Working flow balance constrains: for each demand pair, the total source flow equals the demand, the total sink flow also equals the demand, no net sourcing or sinking of flows for the given source-destination (S-D) pair occurs at any other node.
2. Spare capacity constrains: for each demand pair, the routed spare flow equals to the traffic flow; the spare flow for a source-destination pair does not pass through the working path of the pair. The total sink flow also equals to the spare demand, no net sourcing or sinking of flows for a given spare source-destination pair occurs at any other node.

Based on the above problem description, a feasible solution of the problem includes a closed mesh topology network with working and sparing capacities allocated on links satisfying all the constraints. Note that we do not have constraints on the maximum capacity that can be transported by each link.

Remark 3.1 Given a set of locations as the network nodes, the links among these locations will be considered as the potential links, if they can be established. Moreover, the above problem can be generalized to the design of a survivable legacy network: in a legacy network with a number of established links, the building costs of these established links can be set to zero.

Remark 3.2 In the considered problem, the survivability scheme is assumed to be *span-restoration*. That is, a link used in the found topology graph for working traffic routing are to be protected by a path.

4 Constructive heuristics for the complete MTRS problem

In this section, two parameterized greedy heuristics, namely the working traffic router and the spare capacity allocator, that are used to construct feasible solutions to the complete MTRS problem are presented. In the working traffic router, a network topology is established through routing the working traffic demands. Spare capacities are assigned in case of link failures and the network topology is refined in the spare capacity allocator. Both the two constructive heuristics must satisfy all the constraints. In brief, we construct solutions to the complete MTRS problem by the working traffic routing and spare capacity allocation, while the network topology is established during the routing process automatically.

4.1 Working traffic routing heuristic: the working traffic router

Before describing the heuristic, we shall define some notations.

π permutation of the node pairs with traffic demands (i.e. those non-zero elements in the traffic demand matrix \mathbb{D}), where π_i , $1 \leq i \leq n$ describes the source-destination node pair, and n is the number of node pairs with traffic demands.

\mathbb{M} an $m \times m$ matrix, called *cost matrix* in the following. The weighted graph $(\mathbb{V}, \mathbb{E}, \mathbb{M})$ is the basis of working traffic routing. The cost matrix \mathbb{M} is globally initialized (ahead of the routing procedure) to be the establishment cost of the potential links, i.e., for each link $(i, j) \in \mathbb{E}$, $\mathbb{M}_{ij} = \Omega \mathbb{E}_{ij}$.

\mathbb{L} a binary $m \times m$ matrix, called *link usage matrix*, which is the organization of ℓ_{ij} (cf. Sect. 3) for indicating which links are existing in the network topology. Its elements are initialized to be zero.

\mathbb{W} an $m \times m$ matrix \mathbb{W} , called *allocation assignment matrix*, which is the organization of w_{ij} as described in Sect. 3. It is used to record the number of working flows allocated on the links. The matrix \mathbb{W} is initialized to be zero for all elements.

The heuristic algorithm routes the traffic demands listed in the matrix \mathbb{D} one by one in order π of the node pairs. Assume that for traffic demands π_1, \dots, π_k , ($1 \leq k \leq n$), a set of working paths $\mathbf{WP}^k = \{\mathbf{WP}_{\pi_1}, \dots, \mathbf{WP}_{\pi_k}\}$ have been established. Suppose that the π_{k+1} -th node pair is (s, d) , and its working traffic demands are g , we do not allocate the whole g working demands once a time, but $g_1 < g$ at each time. We will iterate the routing process until all the g demands are allocated for the S-D node pair (s, d) . This is to make the traffic demands distributed in the network as diverse as possible, which can increase the possibility of the network survivability. As discussed in Grover and Doucette (2001), "... a mesh network should generally have a degree of 2.6 to 2.8 or higher because it is above this crossover point that the investment in spare capacity becomes more and more highly leveraged". In light of this view, in the execution of the heuristics, g_1 is set to take values from $I = \{2, 3, 4, 5\}$ (this principle is also adopted in the spare capacity allocator as to be presented in Sect. 4.2). The routing of the π_{k+1} -th working traffic demands is as follows.

- Set $ct := 1$;
- While $\mathbb{D}_{s,d} = g > 0$, do:

Step 1): Set the size of the traffic flows f to be routed: if $g > g_1$, set $f := g_1$; otherwise $f := g$. Update the flows to be routed $g := g - f$.

Step 2): Compute the temporary cost matrix \mathbb{M}' as follows:

- (1) Set $\mathbb{M}' := \mathbb{M}$.
- (2) For each link $(i, j) \in \mathbb{E}$, add the cost of the capacities to be routed, i.e. $\mathbb{M}'_{ij} := \mathbb{M}_{ij} + f\mathbb{E}_{ij}$;

Step 3): Find a shortest path based on the weight graph $(\mathbb{V}, \mathbb{E}, \mathbb{M}')$ by using the Dijkstra algorithm (Cook et al. 1998), the found path will be set as $\mathbf{WP}_{\pi_{k+1}}^{ct}$ for the f traffic demands of the π_{k+1} -th node pair.

Step 4): Update the cost matrix \mathbb{M} as follows:

- (1) If a link, e.g. (i, j) in the found shortest path, has **not** been used in the previous working paths \mathbf{WP}^k , set $\mathbb{L}_{ij} := 1$;
- (2) Add α_1 percent of the routed capacities' cost on the links of $\mathbf{WP}_{\pi_{k+1}}^{ct}$:

$$\mathbb{M}_{ij} := \mathbb{M}_{ij} + f\mathbb{E}_{ij}\alpha_1, \quad \forall (i, j) \in \mathbf{WP}_{\pi_{k+1}}^{ct}.$$

Step 5): Update \mathbb{W} : if a link $(i, j) \in \mathbf{WP}_{\pi_{k+1}}$, $\mathbb{W}_{ij} := \mathbb{W}_{ij} + f$.

Step 6): Set $ct := ct + 1$.

- Set $\mathbf{WP}_{\pi_{k+1}} = \bigcup_{i=1}^{ct} \mathbf{WP}_{\pi_{k+1}}^i$, and $\mathbf{WP}^{k+1} = \{\mathbf{WP}_{\pi_1}, \dots, \mathbf{WP}_{\pi_{k+1}}\}$.

In the routing procedure, we first set a counter ct to count the routing times of the traffic demands. At each routing, Step 1) decides the size of the flows to be routed, the cost to allocate these flows are added to current cost matrix in Step 2) to form a temporary cost matrix. The shortest path of the S-D pair is found in Step 3) based on the current weighted graph that induced from the temporary cost matrix. The cost matrix \mathbb{M} is updated with a control parameter α_1 in Step 4). The traffic flows are then assigned on the links of the found path. The used links and the assigned flows on these links are recorded in matrices \mathbb{L} and \mathbb{W} , respectively, in Step 5). Step 6) increases the number of routing counter. The steps from 1)–6) are iterated until all the c flows are routed. Once all the S-D node pair's traffic demands are routed, we obtain the working path. The whole procedure continues until all the node pairs are routed.

Remark 4.1 Along with the routing process for a S-D node pair, the links of the shortest path $\mathbf{WP}_{\pi_{k+1}}^{ct}$ tend to be more weighted. This will force the shortest path algorithm to find a different path to route the $(ct + 1)$ -th traffic demands, i.e. some or all the links in $\mathbf{WP}_{\pi_{k+1}}^{ct+1}$ will be different from the links in $\mathbf{WP}_{\pi_{k+1}}^{ct}$. Thus, the traffic demands of the π_{k+1} -th S-D node pair will be distributed on the network diversely. In our heuristic, the parameter α_1 adopted in Step 4) is used to control the degree of the diversity. If we route the $(ct + 1)$ -th traffic demands without updating the cost matrix (i.e. $\alpha_1 = 0$), the number of links to be used in the obtained network topology would be very small since the same shortest path will be used to route all the traffic demands of the π_{k+1} -th S-D node pair. On the other hand, if $\alpha_1 = 1$, the routing process will

intend to use more links. In both cases, the obtained network topology will be far away from optimal since the average degree of the network topology will be not in the range of 2.6 to 2.8 or higher, which is discussed to be heuristically optimal in Grover and Doucette (2001). The use of the control parameter α_1 in the range of $(0, 1)$ gives the greedy heuristic the freedom to control the degree of the network topology for a certain network problem.

Remark 4.2 Apparently, the order of the S-D node pairs will effect the obtained network topology. Different orders of the S-D node pairs will result in different network topologies. In summary, the control parameters π , g_1 and α_1 are regarded as the input of the working traffic routing heuristic. Given a set of control parameters π , g_1 and α_1 , the network topology will be constructed deterministically, while the matrices \mathbb{M} , \mathbb{W} and \mathbb{L} will be returned as the output.

4.2 Protective traffic routing heuristic: the spare capacity allocator

Before describing the protection heuristic, again we define some notations as follows:

\mathbb{M}^P an $m \times m$ matrix, called *protective cost matrix*. Similar to the \mathbb{M} used in the working traffic routing heuristic, the protective cost matrix is the basis of the protective routing heuristic. For a link $(i, j) \in \mathbb{E}$, $\mathbb{M}_{ij}^P = 0$ if the link (i, j) is already in the network obtained from the working traffic routing heuristic; otherwise $\mathbb{M}_{ij}^P = \mathbb{M}_{ij}$.

\mathbb{L}^P an $m \times m$ matrix, called *protective link usage matrix*. Similar to \mathbb{L} , it is applied to record the links used in the network topology for the restoration purpose. It is initialized to be zero.

\mathbb{C}^P an $m \times m$ matrix, called *protective capacity assignment matrix*, which is the organization of p_{ij} (cf. Sect. 3). It is used to record the spare capacities assigned on the links, and globally initialized to be zero.

According to the problem description, the assigned capacities for working traffic demands recorded in the matrix \mathbb{W} are to be routed for the purpose of survivability. Suppose that the number of non-zero elements in \mathbb{W} is q (this number could be different for different \mathbb{W}), we denote the order of these non-zero elements as $\sigma = (\sigma_1, \dots, \sigma_q)$. If we assume the coordinates of the σ_i -th element are (u, v) , then we need to find a path from the source node u to the destination node v without passing through the link (u, v) . We call these (u, v) 's as the protective S-D node pairs. The same as the working traffic routing heuristic, the protective traffic routing heuristic routes the assigned capacities one by one in the order of σ . Other inputs of the heuristic include the cost matrix \mathbb{M} , the link usage matrix \mathbb{L} and the capacity allocation matrix \mathbb{W} derived from the working traffic router.

Assume that the spare capacities have already been assigned for the first k protective S-D node pairs in $(\sigma_1, \dots, \sigma_k)$, and a set of protection paths $\mathbf{BP}^k = \{\mathbf{BP}_{\sigma_1}, \dots, \mathbf{BP}_{\sigma_k}\}$ have been established. Suppose that the $(k + 1)$ -th protective S-D node pair w.r.t. σ_{k+1} is (u, v) , and the size of the capacities allocated on the link (u, v) is $\mathbb{W}_{uv} = p$, the protective traffic routing heuristic works as follows:

- Set a temporary capacity allocation matrix $\mathbb{Z} := 0$ and a counter $pt := 1$
- While $p \neq 0$, do

Step 1): Set the size of the traffics to be routed as f : if $p > g_2$, $f := g_2$; otherwise $f := p$; Update $p := p - f$;

Step 2): Compute a temporary protective cost matrix \mathbb{M}' as follows:

- (1) Set the link associated with σ_{k+1} to be infinity, i.e. $\mathbb{M}'_{uv} := \infty$, which means that the link (u, v) is not usable.
- (2) For a link (i, j) which is different from (u, v) , add the cost of the capacities to be routed, i.e. $\mathbb{M}'_{ij} := \mathbb{M}_{ij} + f\mathbb{E}_{ij}$;

Step 3): Find a shortest path by using the Dijkstra algorithm based on the weighted graph $(\mathbb{V}, \mathbb{E}, \mathbb{M}')$. The found path is set as $\mathbf{BP}_{\sigma_{k+1}}^{pt}$, the protective routing path for the f traffic demands of the σ_{k+1} -th protective S-D node pair.

Step 4): Update the protective cost matrix \mathbb{M}^p and \mathbb{L}^p as follows:

- (1) If a link e.g. (i, j) in the found shortest path has **not** been used in the previous paths \mathbf{BP}^k , set $\mathbb{L}_{ij}^p := 1$.
- (2) Add the cost of the allocated spare capacities with parameter α_2 :

$$\mathbb{M}_{ij}^p := \mathbb{M}_{ij} + f\mathbb{E}_{ij}\alpha_2, \quad \forall (i, j) \in \mathbf{BP}_{\sigma_{k+1}}^{pt}.$$

Step 5): Update \mathbb{Z} : $\mathbb{Z}_{ij} := \mathbb{Z}_{ij} + f, \forall (i, j) \in \mathbf{BP}_{\sigma_{k+1}}^{pt}$.

Step 6): Set $pt := pt + 1$.

- Set $\mathbf{BP}_{\sigma_{k+1}} = \bigcup_{i=1}^{pt} \mathbf{BP}_{\sigma_{k+1}}^i$, and $\mathbf{BP}^{k+1} = \{\mathbf{BP}_{\sigma_1}, \dots, \mathbf{BP}_{\sigma_{k+1}}\}$.
- Set $\mathbb{C}^p = \max(\mathbb{C}^p, \mathbb{Z})$.

The same as in the working traffic routing heuristic, we protect g_2 working flows at each time and iterate until all the traffic flows are routed. The size of flows to be protected is set in Step 1). The shortest path is found by the Dijkstra algorithm in Step 3) based on the weighted graph induced from the temporary cost matrix \mathbb{M}' as computed in Step 2). The temporary cost matrix \mathbb{M}' is computed by adding the cost of the flows to be protected on all potential links, and forbidding the use of the link to be protected. After routing, we update the protective cost matrix to record the cost already assigned on the links for the consideration of further routing, and the protective link usage matrix to record the links that have been used (Step 4).

To make the computation of the spare capacities assigned on the links clear, we first give an example. Suppose that we need p_1 traffic flows for a protective S-D node pair on a link (i, j) , if there are p_2 spare flows already assigned on the link, and $p_1 < p_2$, then no more flows on the link (i, j) are required; else if $p_1 > p_2$, we only need to place $p_1 - p_2$ more spare flows on the link. From the example, we see that only the maximal number of flows need to be allocated on a link as the spare capacity. To record the maximum value of the spare capacities assigned on the links, in our heuristic, the matrix \mathbb{Z} is adopted to record the spare capacity allocation for the protective S-D node pairs. The matrix \mathbb{C}^p is then built by taking the maximal value among all the \mathbb{Z} 's, that is, for each link (i, j) , $\mathbb{C}_{ij}^p = \max(\mathbb{C}_{ij}^p, \mathbb{Z}_{ij})$.

To optimally protect the working network topology, the spare capacities should be shared as much as possible to minimize the costs. To this purpose, in our heuristic, a parameter $0 < \alpha_2 < 1$ is applied in (2) of Step 4). In case $\alpha_2 = 0$, no new links will be added in the current network topology. This may result in unfeasible solutions, i.e.

the network topology is not closed. On the other hand, in case $\alpha_2 = 1$, more links will be required which will increase the establishment cost of the links. If we adjust the control parameter α_2 in the range $(0, 1)$, not only the usage of more links can be limited, but also the sharing of the spare capacities will be encouraged.

In the protective routing heuristic, the control parameters are g_2 , α_2 and σ . Given a specific control parameters, the protective link usage matrix \mathbb{L}^P and the protective capacity assignment matrix \mathbb{C}^P , will be deterministically obtained.

4.3 Feasible solution construction and evaluation

As we can see from the routing construction process, a solution can be constructed by using the proposed working and spare traffic routing heuristics given the control parameters π , σ , g_1 , g_2 , α_1 and α_2 . After routing, the working capacities assigned on the links are recorded in the matrix \mathbb{W} , the spare capacities are recorded in the matrix \mathbb{C}^P , and the network topology can be induced from the link usage matrix \mathbb{L} , and the protective link usage matrix \mathbb{L}^P . In the constructed solution, the working flow balance constraints and the spare capacity constraints are automatically satisfied. That is, for any S-D node pair (s, d) , firstly the size of the flows allocated on the links is exactly $\mathbb{D}_{s,d}$, and there is no net flows assigned on the nodes; secondly the protective routing assigns exactly the right traffic flows to protect the working path of the pair (s, d) : there are no net flows, and the protective routing does not pass the working paths as we forbid the use of these links.

The network cost of the constructed feasible solution can then be computed according to these matrices by using Eq. 2. For the sake of convenience, we write the working traffic router as:

$$(\mathbb{W}, \mathbb{L}, \mathbb{M}) = \text{wrouter}(\pi, g_1, \alpha_1),$$

the spare capacity allocator as:

$$(\mathbb{L}^P, \mathbb{C}^P) = \text{srouter}(\sigma, g_2, \alpha_2, \mathbb{W}, \mathbb{L}, \mathbb{M}),$$

and the evaluation of the network as:

$$\text{cost} = \text{evaluate}(\mathbb{W}, \mathbb{L}, \mathbb{L}^P, \mathbb{C}^P).$$

If we ignore the intermediate construction process, we may obtain the network cost given the control parameters π , σ , g_1 , g_2 , α_1 and α_2 as follows:

$$\text{cost} = \mathbf{HEU}(\pi, \sigma, g_1, g_2, \alpha_1, \alpha_2).$$

From the description of the heuristics, we see that the spare capacity allocation depends on the network topology obtained in the working traffic routing heuristic. Since it is not easy to develop a mathematical model for the dependence between these two heuristics, we propose to apply an evolutionary algorithm to tune the control parameters of the heuristics. The network cost for a set of given control parameters can be used as the fitness function.

5 The algorithm

In this section, we first present the estimation of distribution algorithm (EDA) (Larrañaga and Lozano 2002) adopted for tuning the control parameters of the constructive heuristics. Then the proposed algorithm for the complete network design problem, called probabilistic evolutionary algorithm (PEA), is described in details.

5.1 Estimation of distribution algorithm

EDA (Larrañaga and Lozano 2002) is a relatively new type of evolutionary algorithms (EAs). A typical EDA mainly consists operations of initialization, selection, modeling, sampling and replacement. The main characteristics of EDAs, which differentiate it from the other evolutionary approaches, lie on the mechanism of new offspring generation: EDAs sample new solutions from a probability model which is constructed from visited promising solutions, rather than using recombination operators like crossover and mutation in genetic algorithm. Generally speaking, EDAs can be summarized as follows:

1. **Initialization.** A set of S feasible solutions consists of the initial population $x(0)$; Evaluate solutions in the population. Set $t := 0$;
2. **While (not met stop criteria), do:**
3. **Selection.** Select some promising solutions to constitute the parent set $Q(t)$;
4. **Modeling.** Construct a probabilistic model $p(X; t)$ according to $Q(t)$;
5. **Sampling.** Sample some offspring from $p(X; t)$ and evaluate the offspring;
6. **Replacement.** Replace partially or fully the current population with the offspring to constitute a new population $x(t + 1)$; set $t := t + 1$.

In line 1, a set of randomly generated solutions (called “population” in the context of EAs), which are represented as a permutation of the S-D node pairs in this paper, to the optimization problem is initialized. Then the search procedure iterates from line 3 to line 6 until stop criteria have been met. The population evolves through modeling (line 4) and sampling (line 5) based on the extraction of stochastic information from promising solutions (**Selection** in line 3). In the following subsections, we describe the EDA components used for tuning the control parameters in the defined heuristics one by one.

5.1.1 Probability matrix

Without loss of generality, we assume that $x = (x_1, \dots, x_n)$ is a permutation of $U = \{1, 2, \dots, n\}$, where n is the dimension of x . The size of U is n in the working traffic router, and q in the spare capacity allocator. The corresponding probability model $p(X; t)$ at generation t can be organized as an $n \times n$ probability matrix $\mathbf{H}(t)$, where $\mathbf{H}_{ij}(t)$ denotes the probability that the i -th element of a permutation \mathbf{x} is j . The probability matrix \mathbf{H} is initialized to be the same values, that is $\mathbf{H}_{ij}(0) = 1/n$.

5.1.2 Selection and update of probability matrix

The selection operation used in our algorithm is the truncation selection where half the best individuals are selected from the population. At generation t , the selected individuals are used to update the probability matrix $\mathbf{H}(t)$. The simple population-based incremental learning (PBIL) method (Baluja 1994) is applied, that is:

$$\mathbf{H}_{ij}(t) = (1 - \beta) \cdot \frac{2}{S} \sum_{k=1}^{S/2} I_{ij}(Q^k) + \beta \cdot \mathbf{H}_{ij}(t - 1), \quad 1 \leq i, j \leq n; \quad (3)$$

where $Q(t) = \{Q^1, \dots, Q^{S/2}\}$ is the selected population at generation t , $I_{ij}(Q^k)$ is the indicator function and defined as follows:

$$I_{ij}(Q^k) = \begin{cases} 1, & \text{if } Q^k(i) = j; \\ 0, & \text{otherwise;} \end{cases}$$

and $0 \leq \beta \leq 1$ is the learning rate, which controls the contribution of the present promising solutions to the probability matrix $\mathbf{H}(t)$.

5.1.3 Sampling and replacement

An offspring \mathbf{x} (a permutation) is constructed iteratively. Suppose that the first k elements of \mathbf{x} have been filled ($k = 0$ means that we need to create \mathbf{x} from scratch), we denote this partial solution as $\mathbf{x} = (x_1, \dots, x_k)$. The $(k + 1)$ -th element x_{k+1} is to be filled as follows:

- Reset the $(k + 1)$ -th column of the probability matrix $\mathbf{H}(t)$ by setting $\mathbf{H}_{x_i, k+1}(t)$, $1 \leq i \leq k$ as zero, since these elements are forbidden to be x_{k+1} for constructing a feasible permutation.
- Sum the $(k + 1)$ -th column of the probability matrix $\mathbf{H}(t)$,

$$h = \sum_{i=1}^n \mathbf{H}_{i, k+1}$$

set $\mathbf{h}_{k+1}(t) = (\frac{\mathbf{H}_{i1}(t)}{h}, \dots, \frac{\mathbf{H}_{in}(t)}{h})^T$;

- Select an element, e.g. ν , using the roulette wheel method based on $\mathbf{h}_{k+1}(t)$, set $x_{k+1} := \nu$.

The iteration continues until a full solution is constructed. In our algorithm, we sample S offsprings from the probability matrix $\mathbf{H}(t)$.

The replacement is performed after sampling. To perform the replacement, the generated offspring are partially replaced with solutions in the current population. The best S individuals are selected from the combined set of the current population and the sampled offspring. Note that in order to preserve the diversity of the population, duplicates are not allowed in the new population.

5.2 Tuning the control parameters of the constructive heuristics by the PEA

The proposed PEA is a three-phase approach. In the first phase, the control parameters, α_1 and α_2 are decided in a greedy way. The optimal orders of the working and protective S-D node pairs are tuned, respectively, in the following two phases, while the control parameters g_1 and g_2 are adapted in the phases.

I. Tuning the parameters α_1 and α_2 .

- One hundred permutations $\{\pi_1, \dots, \pi_{100}\}$ for the working traffic routing are first randomly generated. For each $\pi_i, 1 \leq i \leq 100$, to construct a feasible solution, an order of the protective S-D node pairs σ_i is randomly created during the spare routing process. Here we assume $\alpha_1, \alpha_2 \in \{0.1, \dots, 0.9\}$ and $g_1 = g_2 = 2$.
- Select the optimal α_1 and α_2 as follows:

$$\{\alpha_1^*, \alpha_2^*\} = \arg \min_{\alpha_1, \alpha_2 \in \{0.1, \dots, 0.9\}} \frac{1}{100} \sum_{j=1}^{100} \mathbf{HEU}(\pi_i, \sigma_i, g_1, g_2, \alpha_1, \alpha_2) \quad (4)$$

II. Tuning the control parameters π .

- Randomly generate S permutations to constitute $x(0) = (\mathbf{x}_1^0, \dots, \mathbf{x}_N^0)$. To evaluate these permutations, a set of S permutations $\sigma_i^0, 1 \leq i \leq S$ of the protective S-D node pairs, $\mathbf{c}_1 = (g_1^1, \dots, g_1^S)$ and $\mathbf{c}_2 = (g_2^1, \dots, g_2^S)$ where each $g_1^i, g_2^i \in I, 1 \leq i \leq S$, are randomly created. The fitness of the initial population $x(0)$ is set as $\mathbf{HEU}(\mathbf{x}_i^0, \sigma_i^0, g_1^i, g_2^i, \alpha_1^*, \alpha_2^*)$.
- Set $t := 0, \text{SucGen} := 0$ and initialize the probability matrix $\mathbf{H}(0)$ as described in Sect. 5.1.1; Select

$$(\pi^*, g_1^*, g_2^*) = \arg \min_{\mathbf{x}_i^0, \sigma_i^0, g_1^i, g_2^i, 1 \leq i \leq S} \mathbf{HEU}(\mathbf{x}_i^0, \sigma_i^0, g_1^i, g_2^i, \alpha_1^*, \alpha_2^*)$$

where π^* is the permutation with the smallest cost in $x(0)$.

- While ($\text{SucGen} < 20$ or $t < 100$), do
 - * Select the best $S/2$ permutations from $x(t)$;
 - * Update the probability matrix $\mathbf{H}(t)$ according to Eq. 3;
 - * Sample a set of permutations $y(t) = \{y_1, \dots, y_S\}$ from $\mathbf{H}(t)$ (see Sect. 5.1.3). In evaluating the sampled permutations, randomly generated $\sigma_i^t, 1 \leq i \leq S$, g_1^* and g_2^* are applied.
 - * Select the best S solutions from $y(t)$ and $x(t)$ to constitute the next generation $x(t + 1)$;
 - * Update g_1^*, g_2^* and π^* as

$$(\pi^*, g_1^*, g_2^*) = \arg \min_{1 \leq i \leq N} \mathbf{HEU}(\mathbf{x}_i^t, \sigma_i^t, g_1^*, g_2^*, \alpha_1^*, \alpha_2^*)$$

- * If a new best permutation π^* is found, set $\text{SucGen} := 0$; otherwise $\text{SucGen} := \text{SucGen} + 1$;
- * Set $t := t + 1$.

III. Tuning the parameter σ .

- Fix the network topology obtained in Phase II and $g_1^*, \alpha_1^*, \alpha_2^*$. Note that if we fixed the network topology, the protective S-D node pairs are uniquely defined.
- Perform the described EDA for searching the order σ of the protective traffic demands with the following fitness function:

$$\mathbf{HEU}(\pi^*, \sigma, g_1^*, g_2, \alpha_1^*, \alpha_2^*)$$

- while g_2 is updated during the evolution procedure similarly to that in Phase II.
- Set σ^* and g_2^* to be the best settings found by the described EDA.

The best solution is then the one generated by the heuristics with the parameter settings $\pi^*, \sigma^*, g_1^*, g_2^*, \alpha_1^*, \alpha_2^*$. Note that we can iterate Phases I and II many times in order to further improve the qualities of the control parameters. However, due to the consideration of computational cost, we do not carry out the iteration in our experiments.

6 Computational results

To evaluate the performance of the proposed algorithm, we compared the proposed algorithm with the branch-and-bound-based heuristic developed by Grover and Doucette (2001) on a set of 28 problems available at <http://www.ee.ualberta.ca/~grover/>. The heuristic was reported comparably with the branch-and-bound algorithm (an exact algorithm for IP implemented in CPLEX) for the full MTRS problem on small-size problem instances, and better than the exact algorithm for medium and large-size instances where the global optimal solutions cannot be produced by CPLEX in a reasonable time (6–18 hours). The zoom-in approach by Pickavet and Demeester (2000) was developed about the same time as Grover and Doucette (2001). We also compare our algorithm with the zoom-in approach.

The proposed algorithm and the zoom-in approach were implemented in C++ and all experiments reported were carried out on identical PCs (AMD Athlon 2400 MHZ) running Linux. To determine appropriate N and β , we carried out the PEA on the test problem “9n36s1” for $\beta = 0.0, 0.1, \dots, 1.0$ and $N = 20, 40, 60, 80, 100, 120, 150, 200$, respectively. We found that the parameter settings $N = 100$ and $\beta = 0.2$ are the best for the PEA. We adopted these parameters for all the test problems although it does not mean that they are always appropriate.

To perform a fair comparison, the proposed PEA and the zoom-in approach (called ZGA in the table) were carried out on each test instances for 10 times. The results are listed in Table 1. In the table, *instance* is the name of the test networks. The test problems used by Grover et al. (Grover and Doucette 2001) are named following some basic characteristics of the problems. For example, a test network named “9n36s1” means that the test network has 9 nodes and 36 links, “s1” denotes the structure of the node locations. Moreover, a certain test network is tested with different edge-to-capacity cost ratios as shown in Column Ω to simulate different situations in practice.

In the table, w_{\min} , w_{\max} and w_{avg} are used to denote the minimal, maximal, average cost found by the PGA and the ZGA over 10 runs, respectively. t_{avg} is the average

running time of the PEA in seconds. To make a fair comparison with the ZGA, the average time t_{avg} used by the PGA is used as the stop criterion for the ZGA. Moreover, the best average network costs found by the compared algorithms are typeset in bold.

The numbers in column $MTRS_{cplex}$ are the results listed in Grover and Doucette (2001) by using CPLEX, where w is the network cost obtained (for some problems, different experiments were carried out, the best results obtained are listed here; for some problems, their heuristic cannot find reasonable solutions within the given time and marked as “–” in the table), $t_{W1+S2+J3}$ is the whole time (in seconds) used in Grover and Doucette (2001) including the three-step (W1, S2, and J3) (note that the problem was solved in Grover and Doucette (2001) by CPLEX 6 MIP solver on a four × 250 MHZ Sun Enterprise processor running the Sun Solaris Operating System 2.6 with 892 MB of RAM).

In the table, $imp.\%$ is defined as $(w_{avg} - w)/c * 100$. That is, $imp.\%$ indicates the improvement of the average network costs found by the PEA over 10 runs against the network cost obtained by the $MTRS_{cplex}$. Notice that some $imp\%$ values are not shown since the CPLEX software cannot solve the corresponding test problems in predefined very-long time (8 hours).

From the table, we can see that in 21 out of 28 instances the PEA finds smaller costs than those of the $MTRS_{cplex}$. For the other 7 instances, the PEA cannot find better solutions, but the deterioration is no larger than 3.5%. Through closer observation, we notice that in case the edge-to-capacity ratio Ω value is large (especially when the ratio is larger than 100), the performance of the developed algorithm is much better than the $MTRS_{cplex}$: the improvement could be up to 17.38%. However, if the ratio value is small, the performance of the PEA could be worse than the CPLEX-based heuristic: in case $\Omega = 5$, the solution qualities found by the PEA are not as good as those of the $MTRS_{cplex}$. One of the possible reasons is that in the test network instances with small Ω 's, the order of the S-D node pairs may have complex dependency relationships, while the simple EDA modeling used in the PEA cannot perform well. However, on average, we can claim the proposed algorithm is very competitive in solving the complete network design problem.

Since the $MTRS_{cplex}$ and the PEA have been carried out on different computers, the comparison of the computational time (shown in Table 1) is not fair. However, it should not be a surprise to see that the proposed algorithm needs significantly less time than that of the $MTRS_{cplex}$.

Considering the time complexity of a feasible solution construction in the developed greedy heuristic, we see that a feasible solution can be constructed in an order of

$$O\left(M^2 \sum_i \sum_j \left[\left\lceil \frac{D_{ij}}{g_1} \right\rceil + \left\lceil \frac{D_{ij}}{g_2} \right\rceil \right] \right)$$

where $\lceil x \rceil$ is the smallest integer than larger than x . Notice that

$$\sum_i \sum_j \left[\left\lceil \frac{D_{ij}}{g_1} \right\rceil + \left\lceil \frac{D_{ij}}{g_2} \right\rceil \right] \leq \left\lceil \frac{T}{g_1} \right\rceil + \left\lceil \frac{T}{g_2} \right\rceil$$

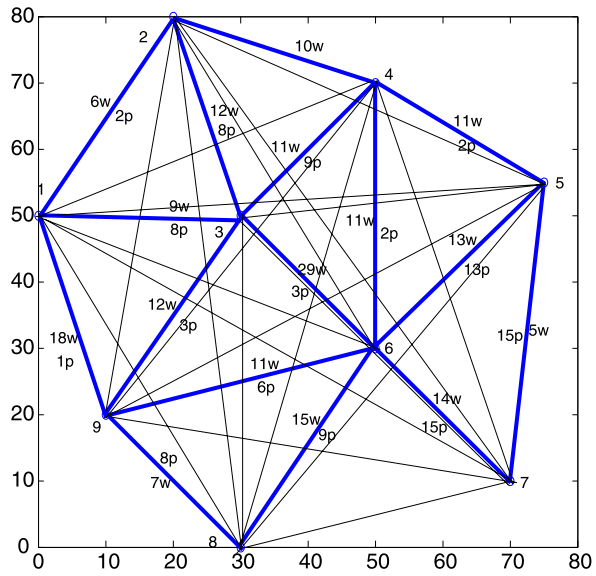
Table 1 The comparison results of PEA and the heuristic algorithm proposed in Grover and Doucette (2001) and the zoom-in approach (Pickavet and Demeester 2000)

Instance	Ω	PEA			ZGA			MTRS _{optex}		Imp.	
		w_{min}	w_{max}	w_{avg}	$t_{avg}(s)$	w_{min}	w_{max}	w_{avg}	$t_{W1+S2+J3}$ (%)		
9n36s1	15	17977	18766	18420.8	16.10	17810	18856	18531.4	19056	160.93	3.33
9n36s2	15	17879	19359	18714.2	20.61	17882	19404	18824.2	18809	193.68	0.50
9n36s3	15	17396	18913	17899.4	16.01	17461	19000	17912.4	17956	76.54	0.32
9n36s4	15	19125	19974	19518.5	12.45	19305	20010	19723.7	20560	312.37	5.07
10n45s1	15	21161	23734	22402.3	24.34	21511	24043	23001.1	22964	2639.94	2.45
10n45s2	15	22173	23534	22767.3	20.93	22301	23627	22815.9	23300	1638	2.29
10n45s3	15	20885	22685	21722.7	28.87	20959	22990	21982.5	21160	2014.87	-0.26
10n45s4	15	21576	22434	22018.4	22.64	22001	22589	23003.2	22850	1195.07	3.64
15n28s1	20	25368	27000	26212.3	50.54	25381	27031	26235.3	27841	511.56	5.85
15n56s1	20	22084	24645	22904.1	176.99	22149	24690	22912.3	0.103	1152.61	-3.06
15n56s1	40	28950	30098	29491.5	94.70	28998	30284	29871.9	0.017	29005	-1.68
19n74s1	5	182701	214532	185670	156.65	192109	228952	201067	0.000	180951	-2.61
19n74s1	20	330240	341598	334865.0	139.70	331407	345998	339861.3	0.000	370712	9.67
19n74s1	50	454906	473102	463016.0	226.29	454687	472910	462818.5	0.002	550897	15.95
19n74s1	100	678083	697536	690075.0	183.61	678093	697528	690101.9	0.203	812501	15.07
19n74s1	200	1072980	1094461	1077681	128.35	10718538	1093917	1067616	0.003	1304441	17.38

Table 1 (continued)

Instance	Ω	PEA			t_{avg} (s)			ZGA			MTRS _{complex}		Imp.
		w_{min}	w_{max}	w_{avg}	w_{min}	w_{max}	w_{avg}	w_{min}	w_{max}	w_{avg}	$t_{W1+S2+J3}$	(%)	
20n80s1	5	118457	123224	120849.0	162.02	120139	143852	132838.6	0.000	117958	5727	-2.45	
20n80s1	20	149109	152905	151191	230.16	150011	153296	151298	0.004	161824	7926	6.57	
20n80s1	50	182490	193899	189865	268.32	183078	199853	190657	0.001	211814	19620	10.36	
20n80s1	100	249210	265335	258881	225.12	249258	265490	259841	0.101	291701	2609	11.25	
23n92s1	5	134557	143881	139422	327.41	140578	150381	147829	0.000	138860	5708	-0.40	
23n92s1	20	171708	190441	178568	281.77	172390	193245	179326	0.003	179559	5624	0.55	
23n92s1	50	247418	263019	254250	375.49	247520	263424	254278	0.150	245551	5923	-3.54	
23n92s1	100	311133	324079	316680	424.73	311129	324065	316631	0.501	342366	5923	7.50	
26n104s1	5	202129	210341	206288	447.47	202487	210873	207012	0.000	-	28800	-	
26n104s1	20	263810	274083	267841	446.07	264285	274975	268098	0.005	-	28800	-	
26n104s1	50	309946	331701	316774	622.17	309876	331547	316668	0.001	329184	12897	3.77	
26n104s1	100	385889	396860	392114	621.30	385832	396546	392091	0.019	460137	12897	14.78	

Fig. 1 The network topology found for 9n36s1 with $\Omega = 15$ and cost 18308



where M is the size of the network, $T = \sum_i \sum_j \mathbb{D}_{ij}$ is the size of the traffic demands. Hence the time complexity to generate a solution in the PEA is in the order of $2.5M^2T$ since both g_1 and g_2 are less than five. As a comparison, we see that the number of variables and constraints in a direct solution of $MTRS_{cplex}$ is:

$$\mathcal{O}(M^4 + M^2T) + \mathcal{O}(M^3 + T^2) \tag{5}$$

Obviously, the time complexity of the exact algorithms, which highly depends on the problem size, used in Grover and Doucette (2001) is for sure significantly larger than that of the developed greedy heuristics. Therefore, we can say that the speedup is mostly due to the algorithmic improvement rather than the computers.

In the comparison between the PEA and the ZGA, the two-tailed t-test is applied on the experimental results. The ‘p-value’ column in Table 1 lists the p-values of the t-test. The p-value smaller than 0.05 implies that the difference between the results of the PEA and the ZGA is significant. From the table, we see that in 19 out of the 28 test networks, the PEA performs significantly better than the ZGA in terms of solution quality. In 3 of the test networks, the ZGA is significantly better than the PEA, while in the rest of the test networks, they perform statistically similar.

From the results, we see that in most cases, the better performance of the ZGA is achieved when the ratio Ω is bigger than 100. Moreover, the ZGA achieves a similar worse performance as the PEA against the $MTRS_{cplex}$ when the ratio is small. We can then conjecture that the networks with big ratio Ω have local structures that are easy to be exploited, which enables the success of the local search in the zoom-in approach. As a conclusion, we can claim that the PEA performs better than the ZGA on average.

In Figs. 1 and 2, two examples of the network topologies found by the PEA with smaller cost than those of the $MTRS_{cplex}$ are shown with capacities assigned on the

Fig. 2 The network topology found for 9n36s4 with $\Omega = 15$ and cost 18765

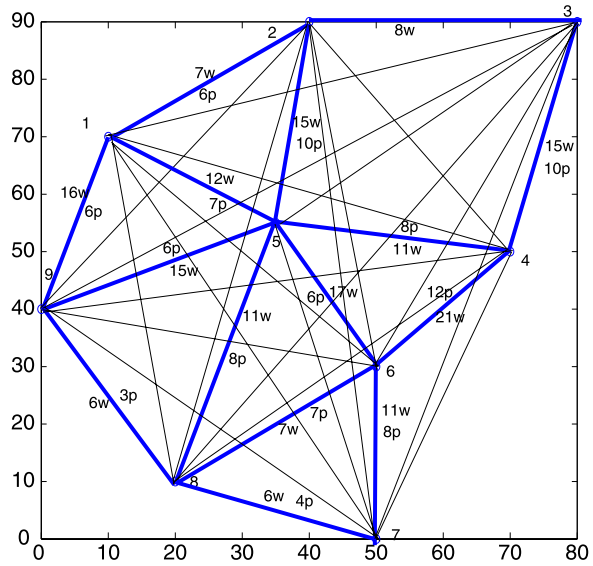
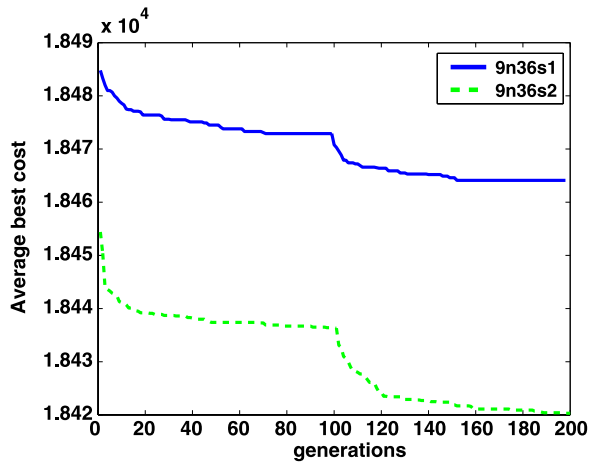


Fig. 3 Evolution of the average of the lowest network costs against generations for test network 9n36s1 and 9n36s2



links (for example, 12w5p means that 12 units of capacities and 5 units of spare capacities are assigned on the link). Bold links indicate the found optimal network topologies. Figure 3 demonstrates the evolution procedure of the PEA in searching for two network instances 9n36s1 and 9n36s2, respectively. From Fig. 3, we see that there are significant decreases during the evolution between the phases for tuning π and σ .

7 Conclusion

The problem of simultaneously optimizing mesh-restorable topology, routing, and sparing in a network design involves a large number of decision variables and con-

straints and is proved to be \mathcal{NP} -hard. For this reason, most existing heuristics in literature adopted a zoom-in approach: the full problem is split into several simplified subproblems, and the search is from rough to fine solutions. This paper presents a different approach in which the routing and sparing are optimized separately. All the constraints must be satisfied in both processes. A two-level evolutionary approach is applied to search for the optimal solutions. The proposed algorithm was compared with the latest heuristic based on CPLEX by Grover and Doucette (2001), and a zoom-in approach by Pickavet and Demeester (2000). The computational results show that the proposed algorithm outperforms the heuristic based on CPLEX in 21 test problems out of 28 in terms of solution quality with significantly less computational time, and outperforms the zoom-in approach in 19 test problems.

In the future, the two-level algorithmic structure shall be evaluated further. In particular, we shall investigate the effectiveness of the proposed two-level algorithmic structure to other hard optimization problems, especially those optimization problems that classical evolutionary algorithms may not handle well due to a large number of decision variables involved. Moreover, as we observed in the experiments that simple EDA cannot perform well for network instances with complex dependencies among the order of the S-D node pairs, we will investigate the application of high-order EDAs and/or ant colony optimization methods to this problem in the future.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful and constructive comments.

References

- Al-Rumaih, A., Tipper, D., Liu, Y., Norman, B.A.: Spare capacity planning for survivable mesh networks. In: Proceedings of the IFIP-TC6 / European Commission International Conference on Broadband Communications, High Performance Networking, and Performance of Communication Networks. Lecture Notes in Computer Science, pp. 957–968. Springer, Berlin (2000)
- Álvarez, A.M., González, J.L., De-Alba, K.: Scatter search for a network design problem. Technical Report PISIS-2003-02, Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica, División de Posgrado en Ingeniería de Sistemas, Mexico (2003)
- Anticona, M.T., Villegas, C.E.: Design of ant colony-based ant route for solve the OSPF problem. CERMA, pp. 386–394 (2007)
- Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report, Carnegie Mellon University (1994)
- Buriol, L.S., Resende, M.C.G., Thorup, M.: Survivable IP network design with OSPF routing. *Networks* **49**(1), 51–64 (2007)
- Boorstyn, R., Frank, H.: Large-scale network topological optimization. *IEEE Trans. Commun.* **COM-25** **1**, 29–47 (1977)
- Cancela, H., Robledo, F., Rubino, G.: A GRASP algorithm with tree based local search for designing a survivable Wide area network backbone. *J. Comput. Sci. Technol.* **4**(1), 52–58 (2004)
- Chou, W., Gerla, M., Frank, H., Eckl, J.: A cut saturation algorithm for topological design of packet switched networks. In: Proc. IEEE National Telcom. Conf., pp. 1074–1085 (1974)
- Cinkler, T., Henk, T., Gordos, G.: Stochastic algorithms for thrifty single-failure-protected networks. In: Proceedings of Design of Reliable Communication Networks, Munich, Germany, pp. 299–303 (2000)
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial Optimization. Wiley, New York (1998)
- Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw–Hill, New York (1999)
- Dorigo, M., Maniezzo, V., Colnari, A.: The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern., Part B* **26**(1), 29–41 (1996)

- Faria, Jr.H., Binato, S., Resende, M.G.C., Falcao, D.M.: Power transmission network design by greedy randomized adaptive path relinking. *IEEE Trans. Power Syst.* (2004)
- Ferentinos, K.P., Tsiligindis, T.A.: A memetic algorithm for dynamic design of wireless sensor network. *CEC 2007*, pp. 2774–2781
- Gavish, B.: Topological design of computer communication networks—the overall design problem. *Eur. J. Oper. Res.* **58**, 149–172 (1992)
- Gil, C., Banos, R., Montoya, M.G., Gomez, J.: Performance of simulated annealing, tabu search and evolutionary algorithms for multi-objective network partitioning. **1**, 55–64 (2006)
- Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic, Dordrecht (1998)
- Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. *Control Cybern.* **39**(3), 653–684 (2000)
- Grover, W.D., Doucette, J.: Topological design of survivable mesh-based transport networks. *Ann. Oper. Res.* **106**, 79–125 (2001)
- Hsu, C.-Y., Wu, J., Wang, S., Hong, C.: Survivable and delay-guaranteed backbone wireless mesh network design. *J. Parallel Distrib. Comput.* **68**, 306–320 (2008)
- Jan, R.H.: Design of reliable networks. *Comput. Oper. Res.* **20**(1), 25–34 (1993)
- Kershenbaum, A.: *Telecommunications Network Design Algorithms*. McGraw–Hill, New York (1993)
- Kershenbaum, A., Kermani, P., Grover, G.A.: MENTOR: An algorithm for mesh network topological optimization and routing. *IEEE Trans. Commun.* **39**(3), 503–513 (1991)
- Larrañaga, P., Lozano, J.A.: *Estimation of Distribution Algorithms: a New Tool for Evolutionary Computation*. Kluwer Academic, Dordrecht (2002)
- Mukherjee, B.: WDM optical communication networks: progress and challenges. *IEEE J. Sel. Areas Commun.* **18**(10), 1810–1824 (2000)
- Ooubutra, M., Avemprayoon, P., Wnrktein, P.: Topological communication network design using ant colony optimization. In: *Proceedings of the 7th Conference on Advanced Communication Technology*, vol. 2, pp. 1147–1151 (2005)
- Pickavet, M., Demeester, P.: A zoom-in approach to design SDH mesh-restorable networks. *J. Heur.* **6**(1), 103–126 (2000)
- Pierre, S., Elgibaoui, A.: A tabu-search approach for designing computer-network topologies with unreliable components. *IEEE Trans. Reliab.* **46**(3), 350–359 (1997)
- Randall, M., McMahon, G., Sugden, S.: A simulated annealing approach to communication network design. *J. Comb. Optim.* **6**, 55–65 (2002)
- Runggeratigul, S.: A memetic algorithm for communication network design taking into consideration an existing network. *Appl. Optim.* 615–626 (2004)
- Sakauchi, H., Nishimura, Y., Hasegawa, S.: A self-healing network with an economical spare-channel assignment. In: *Proc. IEEE Globecom*, pp. 438–444 (1990)
- Shen, J., Xu, F., Zheng, P.: A tabu search algorithm for the routing and capacity assignment problem in computer network. **32**(11), 2785–2800 (2005)
- Taheri, J., Zomaya, A.Y.: A simulated annealing approach for mobile location management. **30**(4), 714–730 (2007)
- Wille, E.C.G., Mellia, M., Leonardi, E., Marsan, M.A.: Topological design of survivable IP networks using meta-heuristic approaches. In: *The 3rd International Workshop on QoS in Multiservice IP Networks* (2005)
- Zhang, Q., Sun, J., Xiao, G., Tsang, E.: Evolutionary algorithm refining a heuristic: a hybrid method for shared path protection in WDM networks under SRLG constraints. *IEEE Trans. Syst. Man Cybern., Part B* **37**(1), 51–61 (2007)