Omega xxx (xxxx) xxx

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

A matheuristic for a telecommunication network design problem with traffic grooming☆

Xinyun Wu[a], Zhipeng Lü[b,*], Fred Glover[b]

[a] SMART, School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan 430074, PR China
[b] ECEE - College of Engineering & Applied Science, University of Colorado, Boulder, Colorado 80309, USA

ARTICLE INFO

Article history:
Received 7 June 2017
Accepted 11 November 2018
Available online xxx

Keywords:
Network design problem
Traffic grooming and routing
Matheuristic
Mixed integer programming
Tabu search

ABSTRACT

This paper addresses a network design and traffic grooming problem arising in optical telecommunication networks that are based on wavelength division multiplexing. Given a set of nodes and a set of traffic demands between these nodes, the network design and traffic grooming problem (NDGP) consists of installing a minimum number of lightpaths between the nodes and of routing the demand on the lightpaths while respecting capacity constraints. We introduce a new mathematical formulation of the NDGP as well as a hybrid algorithm capable of finding high quality solutions in short computing times. The proposed algorithm uses linear and mixed integer programming as slave methods and embeds them within a tabu search procedure. Computational results and comparisons with an existing method from the literature show the effectiveness of the proposed algorithm. Further analyses also show the efficiency of the neighborhood structure and of its evaluation technique.

1. Introduction

The network design and traffic grooming problem (NDGP) arises in the design of optical telecommunication networks. One of the main costs in a wavelength division multiplexing (WDM) network is that of the transceivers installed, which provide optical connections (edges) between pairs of nodes in the network [21]. The optical connections in the network are called lightpaths and they are responsible for transferring traffic demands. In practice, the capacity of lightpaths is usually much larger than the bandwidth of one traffic demand. Assigning an exclusive lightpath to each demand is thus costly and unjustified. Since one lightpath can usually be shared by several demands, effective optimization algorithms can be developed to reduce the cost of the required transceivers.

In the optical network design field, there exist two main phases: one is the planning phase, which focuses on constructing the lightpath network with the minimum cost; the other is the operating phase, which considers how to use the existing lightpath network most efficiently. In the operating phase, traffic grooming, routing, and wavelength assignment problems are usually considered in an independent or integrated manner based on a predefined lightpath network, while in the planning phase similar problems (i.e., traffic grooming and routing) are studied too, but the main concern is to design the lightpath network with the minimum cost. In the literature, most existing papers study the operating phase problems instead of the planning phase ones, which motivates us to study the planning phase problem in this paper. The importance of the planning phase was also raised by our industrial partner, and hence we do not focus attention on the wavelength assignment problem, as is usually done, since the latter is restricted to operating phases. The effect known as attenuation is another important concern which should be considered in the planning phase of optical network design. A lightpath is an unsubstantial connection between two sites, which may traverse several sites and fibers before reaching its destination. The lightpath may not be possible to create if the route is too long. To make the problem more general, we do not consider the attenuation issue in this study. However, the proposed algorithm can be adapted to consider this issue naturally as described in Section 3.2.2.

The NDGP tackled in this paper was recently introduced by Wu et al. [32] and can be stated as follows. Given a set of nodes and a set of traffic demands, the objective is to design an optical network with the smallest number of lightpaths to satisfy all the traffic demands. Wu et al. proposed a two-level iterated local search (TL-ILS) algorithm, which consists of two local search procedures in a nested structure. The main local search handles the transformation of the topology (to decrease the number of lightpaths) while the inner search validates the feasibility of the current solution (i.e. the grooming subproblem). The inner algorithm, which is implemented

☆ This manuscript was processed by Associate Editor Sterna.

* Corresponding author.
E-mail addresses: xavier@hust.edu.cn (X. Wu), zhipeng.lv@hust.edu.cn (Z. Lü), fredwglover@yahoo.com (F. Glover).

https://doi.org/10.1016/j.omega.2018.11.012
0305-0483/

through a tree-search based neighborhood, is the most important part of the TL-ILS. This algorithm uses a breadth-first search strategy and a cutoff mechanism when there is no hope to find a better route. These components reduce redundant calculations when the candidate routes for a given traffic demand share common subpaths.

In this paper, we propose a matheuristic, i.e., an algorithm that combines local search with mathematical programming techniques, for the NDGP. The proposed algorithm implements tabu search as the master algorithm and embeds two slave algorithms, which are linear programming and mixed integer programming. This new algorithm has found improved solutions to 21 of 22 instances in a benchmark set.

During the last decades, optimization problems arising in WDM networks have received extensive attention due to their economic significance. One of these problems is the traffic grooming problem which consists of routing each traffic demand in the network while respecting the capacity of the lightpaths. This problem was proven to be NP-complete by Wang and Gu [29]. Chen et al. proposed an effective and efficient hierarchical traffic grooming framework for WDM networks [8], while Saleh and Kamal addressed the problem of designing and provisioning WDM networks to support many-to-many traffic grooming in order to minimize the overall network cost [26]. They also introduced two novel approximation algorithms for the many-to-many traffic grooming problem [27]. A mathematical formulation of the traffic grooming problem in WDM mesh networks was presented by Zhu and Mukherjee [40]. Zhang et al. [38] proposed a multi-layer auxiliary graph to jointly solve the electrical-layer routing and optical-layer routing and spectrum assignment problems. This work was then improved by Zhang et al. [37], who added a "sub-transponder layer" in the auxiliary graph and proposed two spectrum reservation schemes for multi-flow transponders to improve the transponder's utilization.

Liu et al. [19] investigated the survivable traffic grooming problem for elastic optical networks with flexible spectrum grid employing new transmission technologies. Rubio et al. [25] proposed two evolutionary algorithms, which are multiobjective variants of the standard differential evolution and variable neighborhood search, for solving the traffic grooming problem. Wu et al. [34] studied a very similar problem as this paper but with an additional simple physical path constraint for each traffic demand. Dutta et al. [10] articulated how the traditional optical networking research area of traffic grooming may be combined with recent advances in Internet architecture, specifically the proposed ChoiceNet Future Internet architecture, to create an agile system capable of reflecting both provider and customer interests on an ongoing basis as network conditions change. Yazar et al. [36] investigated the green field design and the copper field re-design problems originating from one of the largest Turkish Internet service providers. Their computational results show the efficacy of the proposed models for moderate dimension scenarios.

The traffic grooming problem is often combined with the routing and wavelength assignment problem, which is denoted as the traffic grooming routing and wavelength assignment (GRWA) problem [4,7,16,18,39]. Vignac et al. [28] proposed a mathematical formulation and an exact method for solving the GRWA problem. Their approach is highly effective for realistic instances. Rubio-Largo et al. [3] presented a tri-objective optimization algorithm for solving the traffic grooming and wavelength assignment problem with objectives of very different scales. They applied their proposed algorithm to a real-world telecommunication problem and obtained very promising results. The routing and wavelength assignment problem is itself an important problem aimed at reducing the usage of wavelengths (see, e.g., [6,22,24,33]).

The NDGP is also related to the classical multi-commodity capacitated network design problem (MCND) [9,11]. However, as has been noted in [32], the NDGP is different from the MCND and its variants [1,2,5,15], not only because it requires the flow variables to be integer, but also because the network to be designed can be a multi-graph in the case of the NDGP. These features make the NDGP particularly challenging to solve. There are several similar problems (especially the non-bifurcated network design problem (NBNDP) [5]) studied in the literature but most consider a given set of edges that should either be open or closed. This contrasts with the NDGP in which only a set of nodes is given and the network, which may form a multi-graph, has to be designed from scratch.

The remainder of the paper is organized as follows. A new mathematical formulation of the problem is presented in Section 2. The framework of the proposed matheuristic is then presented in Section 3, while the formulation used in the slave algorithms is introduced in Section 3.1. The neighborhood structure of the master problem is described in Section 3.2, and the proposed algorithm is detailed in Section 3.3. Experimental results and the analysis are described in Section 4, before concluding the paper in Section 5.

2. Problem definition and mathematical formulation

2.1. Problem description

The network design and traffic grooming problem is defined as follows. We are given a node set $V = \{1, \ldots, n\}$ and a set of m traffic demands Γ. The elements of Γ are tuples (s_t, d_t, b_t) representing the source and sink nodes as well as the bandwidth of each traffic demand t. The objective is to design an optical network, denoted as a multi-graph $G = (V, L)$, and groom all the demands in set Γ with the minimum number of lightpaths. Here, L represents the set of lightpaths used in the network. Note that set L is not defined *a priori* but is actually the output of the optimization model.

The traffic demands and the lightpaths are undirected. The network to be designed is a multi-graph, i.e., there may be several lightpaths in the graph with the same source and sink nodes. What makes the problem difficult is that the traffic demands are unsplittable in the sense that every traffic demand has to be assigned to a unique sequence of lightpaths forming a path from the origin to the destination of the demand. Thus, we cannot consider the lightpath capacity to be additive on each edge.

Fig. 1 depicts an example: We are given a node set V with 4 nodes and a traffic set T with 6 traffic demands of bandwidth from 1 to 2, assuming that the capacity of each lightpath is 3. The straightforward solution to this problem is to construct a network with 6 lightpaths, each carrying one traffic demand, as shown in Fig. 1(a). The lightpaths are represented by the gray bold lines while the traffic demands are represented by the colored lines. As an alternative, we can groom traffic (A, C, 1) to one of the lightpaths AB and lightpath BC, and groom traffic demand (D, A, 1) to another of the lightpaths AB, lightpath BC and CD, as shown in Fig. 1(b). Thus, only 4 lightpaths are needed to satisfy all the 6 traffic demands, saving 2 lightpaths compared to the previous solution. The network with 4 lightpaths is the optimal solution for this example. However, if only one lightpath is permitted between each pair of nodes, the optimal solution would be 5 lightpaths, as shown in Fig. 1(c).

The model introduced in [32] assumes that when several lightpaths are installed between a pair of nodes, their capacity can be shared, which may lead to infeasible solutions. Suppose there are three traffic demands with bandwidth of 2 and the capacity of a lightpath is 3. If these demands all travel through the same two nodes, there should be at least three lightpaths between the pair of nodes to carry them all. However, according to the model described in [32] two lightpaths are enough, which does not reflect

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

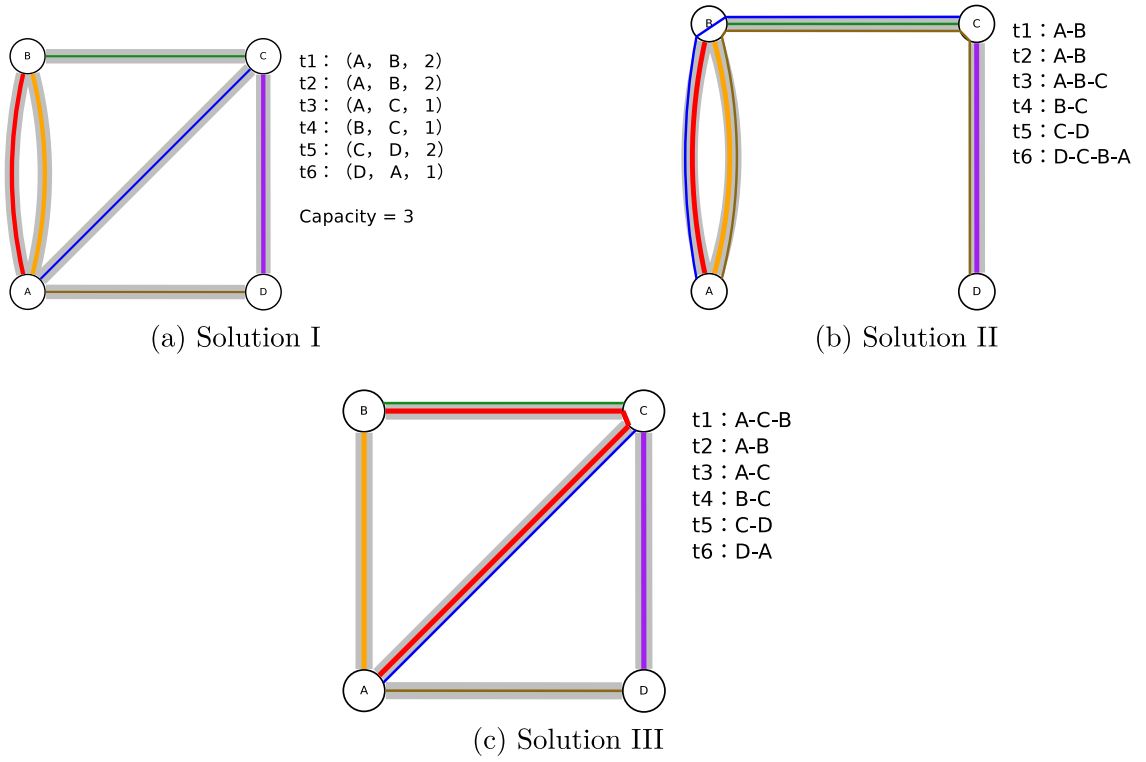


Fig. 1. An example for the NDG.

reality. The reason lies in the fact that each traffic should be assigned to a specific lightpath when there are multiple traffic demands between a pair of nodes. By respecting this constraint, the linear programming model is insufficient to describe the problem exactly.

In the following section, we introduce a new mathematical formulation that properly models the case where traffic demands are unsplittable.

2.2. Problem formulation

Let C denote the capacity of each lightpath. We define T_i^t to equal 1 if traffic t originates from i, -1 if traffic t is destined to i, and 0 otherwise. K represents the maximum number of lightpaths that can be installed in the network. Since in the worst case each traffic demand will be assigned to a different lightpath, the number of lightpaths will not exceed the number of traffic demands. Thus, an upper bound on the number of lightpaths is m (number of traffic demands). To make the model feasible, we can set $K = m$ or another value less than m if a better upper bound is already known. For each potential lightpath l (with $1 \le l \le m$), we also define the following binary decision variables:

- x_l equals 1 if lightpath l is in use, and 0 otherwise;
- s_i^l equals 1 if the origin of lightpath l is node i, and 0 otherwise;
- d_i^l equals 1 if the destination of lightpath l is node i, and 0 otherwise;
- v_l^t equals 1 if demand t uses l in the forward direction, and 0 otherwise;
- w_l^t equals 1 if demand t uses l in the reverse direction, and 0 otherwise.

Although the graph $G = (V, L)$ to be designed is undirected, we treat $l \in L$ as a pair of two directed arcs (denoted as v_l^t and w_l^t) to make the formulation clear.

Given this notation, we can formulate the NDGP as follows.

Minimize

$$\sum_{l=1}^{K} x_l \tag{1}$$

subject to:

$$\sum_{i=1}^{n} s_i^l = x_l \quad l = 1, 2, \ldots, K \tag{2}$$

$$\sum_{i=1}^{n} d_i^l = x_l \quad l = 1, 2, \ldots, K \tag{3}$$

$$x_l - v_l^t - w_l^t \ge 0 \quad t = 1, 2, \ldots, m \;\; l = 1, 2, \ldots, K \tag{4}$$

$$\sum_{l=1}^{K} v_l^t \left(s_i^l - d_i^l\right) + \sum_{l=1}^{K} w_l^t \left(d_i^l - s_i^l\right) = T_i^t \quad t = 1, 2, \ldots, m \;\; i = 1, 2, \ldots, n \tag{5}$$

$$\sum_{t=1}^{m} b_t \left(v_l^t + w_l^t\right) \le C x_l \quad l = 1, 2, \ldots, K \tag{6}$$

$$x_l, s_i^l, d_i^l, v_l^t, w_l^t \in \{0, 1\} \quad t = 1, 2, \ldots, m \;\; l = 1, 2, \ldots, K \;\; i = 1, 2, \ldots, n. \tag{7}$$

Constraints (2) and (3) ensure that each lightpath has one source and one sink if the corresponding lightpath is used. Constraints (4) ensure that every traffic demand only uses feasible lightpaths, and one lightpath cannot be used by the same traffic demand both forward and backward. Constraints (5) ensure that each traffic demand has a feasible path from its origin to destination. Specifically, for each node on the path of each traffic demand,

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

the difference between the out-degree and in-degree is equal to 1, -1 or 0 depending on whether the node is the origin, destination or an intermediate node for the traffic, respectively. Since the flow variables v_l^t and w_l^t are binary variables, they specify whether the traffic demand uses the lightpath or not. This ensures that traffic demands will not be split. Constraints (6) ensure that capacity constraints are satisfied.

The network of lightpaths to be designed is undirected. However, there is a direction in describing the flow of the traffics. Thus, variables v and w are respectively used to denote the forward and reverse directions when a lightpath is used by a traffic demand. The two variables are summed when calculating the bandwidth in constraints (6).

Since the above formulation contains the quadratic constraints (5), we propose the following linearization model.

2.3. Linearization model

Our linearization model succeeds in removing the quadratic constraints (5) to yield a formulation that is tractable for a commercial general solver, like CPLEX. We first change the constraints (5) into the following form:

$$\sum_{l=1}^{K}\left(v_l^t s_i^l - v_l^t d_i^l + w_l^t d_i^l - w_l^t s_i^l\right) = T_i^t \quad t=1,2,\ldots,m \;\; i=1,2,\ldots,n \tag{8}$$

Next, we add the following constraints, enabling the quadratic elements in the constraints of (8) to be replaced by four sets of variables (α, β, γ, ζ).

$$\alpha_{li}^t \le v_l^t \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{9}$$

$$\alpha_{li}^t \le s_i^l \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{10}$$

$$\alpha_{li}^t \ge v_l^t + s_i^l - 1 \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{11}$$

$$\beta_{li}^t \le v_l^t \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{12}$$

$$\beta_{li}^t \le d_i^l \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{13}$$

$$\beta_{li}^t \ge v_l^t + d_i^l - 1 \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{14}$$

$$\gamma_{li}^t \le w_l^t \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{15}$$

$$\gamma_{li}^t \le d_i^l \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{16}$$

$$\gamma_{li}^t \ge w_l^t + d_i^l - 1 \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{17}$$

$$\zeta_{li}^t \le w_l^t \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{18}$$

$$\zeta_{li}^t \le s_i^l \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{19}$$

$$\zeta_{li}^t \ge w_l^t + s_i^l - 1 \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n \tag{20}$$

$$\alpha_{li}^t, \beta_{li}^t, \gamma_{li}^t, \zeta_{li}^t \ge 0 \quad t=1,2,\ldots,m \;\; l=1,2,\ldots,K \;\; i=1,2,\ldots,n. \tag{21}$$

Given the above constraints, the constraints (8) can then be changed into:

$$\sum_{l=1}^{K}(\alpha_{li}^t - \beta_{li}^t + \gamma_{li}^t - \zeta_{li}^t) = T_i^t \quad t=1,2,\ldots,m \;\; i=1,2,\ldots,n \tag{22}$$

Therefore, the problem can now be expressed in an integer linear programming model as follows.

Minimize

$$\sum_{l=1}^{K} x_l$$

subject to : (2)–(4), (6), (7), (9)–(22)

In spite of the improvements represented by the preceding formulation, our experiments demonstrate that this linearized model is still intractable for an ILP solver for moderate and large scale instances. Therefore, we have developed a specially tailored matheuristic to solve the NDG problem.

3. Solution method

The matheuristic algorithm that we propose to solve the NDGP is a matheuristic that relies on a local search to design the network and on linear programing to solve the grooming problem. For an overview of matheuristics, we refer the interested reader to Maniezzo et al. [20]. Tabu search is a natural procedure to use as a master algorithm for a matheuristic, since it was proposed and implemented together with exact algorithms, and included the design of mixed integer programming methods using linear programming [12,13], long before the "matheuristic" term was invented.

The pseudocode of our algorithm is given in Algorithm 1, where X_{best} records the best solution, i.e., the network with the fewest lightpaths found so far. At the beginning, X_{best} is initialized by a greedy algorithm. The initialization method is the same as in [32] and can be described as follows: from an empty solution, assign each traffic demand sequentially in the current network, introducing a new lightpath whenever the assignment of a demand is infeasible. In each local search iteration, a randomly selected lightpath is removed from X_{best}. The resulting network is then passed to the TS_κNDG procedure, whose role is to try to find a feasible network with a number of lightpaths $k = |X_{best}| - 1$. The solution with the smallest number of lightpaths found by the algorithm is returned at the end of the *nbIter* local search iterations, where *nbIter* is a parameter of the algorithm, and *iterTS* represents the number of iterations used by procedure TS_κNDG.

In the above description, the stopping criterion is the total number of iterations (*nbIter*), but it is easy to adapt the algorithm

Algorithm 1 Algorithm for the NDGP.

```
1: X_best ← Initialization()
2: iter ← 0
3: while iter < nbIter do
4:     X ← REMOVE_LIGHTPATH(X_best)
5:     (succ, X, iterTS) ← TS_κNDG(X, nbIter − iter)
6:     iter ← iter + iterTS
7:     if succ = true then
8:         X_best ← X
9:     else
10:        return X_best
11:    end if
12: end while
13: return X_best
```

to use an alternative stopping criterion, such as the total computing time.

It can be seen that our approach is based on a reduction to the decision version of the problem, the k-NDGP. In order to solve a NDGP instance, i.e., to design a network $G = (V, L)$ with the fewest lightpaths, our algorithm tries repeatedly to find a network G with k lightpaths, i.e., it tackles a series of k-NDGP instances with decreasing values of k. This is one of the main differences between the proposed algorithm and that of [32]. Therefore, in the following sections we mainly discuss the k-NDGP as well as the detail of the procedure TS_kNDG.

3.1. A MIP approach to the grooming subproblem

In this section we introduce an exact method to solve the grooming subproblem of the NDGP which we define as follows. Given a fixed optical network and a set of traffic demands, the goal is to determine the routes for all the traffic demands ensuring that there is no overloaded lightpath in the network. From the previous definition of NDGP, it is clear that the grooming subproblem is exactly the NDGP with fixed lightpath decision variables (x_l, s_i^l and d_i^l). For the grooming subproblem, if the integrality constraints on the flow variables (v_l^t and w_l^t) are relaxed, the problem becomes a multi-commodity flow problem with constraints but without an objective function. This multi-commodity flow problem with fractional flow variables can easily be solved by linear programming. However, this recourse to an easy constraint satisfaction formulation does not help very much. In order to embed the exact method within the local search heuristic, it is desirable to understand the causes of infeasibility when the instance is infeasible. Therefore, we change this decision problem into another optimization problem. For each lightpath l, we define a new constant Λ_i^l equal to 1 if the lightpath originates from i, -1 if it terminates at i, and 0 otherwise. We also define variables δ_l representing the overload for each lightpath l. The model for the grooming subproblem can then be given as follows.

Minimize

$$F = \sum_{l=1}^{|L|} \delta_l \tag{23}$$

subject to

$$\sum_{l=1}^{|L|} \left(\Lambda_i^l v_l^t - \Lambda_i^l w_l^t\right) = T_i^t \quad t = 1, 2, \ldots, m \quad i = 1, 2, \ldots, n \tag{24}$$

$$\sum_{t=1}^{m} b_t \left(v_l^t + w_l^t\right) - \delta_l \le C \quad l = 1, 2, \ldots, |L| \tag{25}$$

$$\delta_l \ge 0 \quad l = 1, 2, \ldots, |L| \tag{26}$$

$$v_l^t, w_l^t \in \{0, 1\} \quad t, l = 1, 2, \ldots, m. \tag{27}$$

Constraints (24) ensure the flows for each traffic demand. They are almost identical to constraints (5) in the original model. Constraints (25) guarantee that the flow on lightpath l minus δ_l should be less than or equal to the capacity of one lightpath. The objective function (23) is the total overload of the network. It is clear that the solution represents a feasible grooming configuration if and only if $F = 0$. Moreover, the solution identifies the overloaded lightpaths when the grooming configuration is infeasible.

This mixed integer programming problem can be solved by a MIP solver such as CPLEX in reasonable time, because the number of variables is not excessive. Another good feature of this formulation is that it has a small integrality gap, a characteristic that proves valuable when the model is used to evaluate the neighborhood structure.

3.2. A matheuristic for the k-NDGP

In this section, we explain how the exact algorithm for the grooming subproblem is embedded within a local search algorithm. The local search algorithm is considered as the master algorithm while the MIP and LP procedures are used as slave procedures integrated into the local search.

In the following, we first present the local search approach by defining the search space, the cost function, and the neighborhood. The procedure to tackle the k-NDGP is described in Section 3.3.

3.2.1. Search space and cost function

The search space (set of configurations) explored by our local search procedure is denoted by $\mathcal{S}$. In the proposed local search approach, a configuration X is any network of k edges (lightpaths) with an associated assignment for each traffic demand.

The cost $f(X)$ is defined as the overload of the network:

$$f(X) = \sum_{l=1}^{m} \left[\max \left(\sum_{t=1}^{m} b_t (v_l^t + w_l^t) - Cx_l, 0 \right) \right]. \tag{28}$$

We note that a configuration X represents a feasible solution of the k-NDG if and only if $f(X) = 0$. In our algorithm, $f(X)$ is evaluated by solving the grooming subproblem which is described in Section 3.1.

3.2.2. Neighborhood definition

Given a configuration $X \in \mathcal{S}$, we denote by $< x, y >$ the operation, named a *twist* operator, by which a lightpath x is removed from G, and a new lightpath y is added to G, with the restriction that x and y must share a common vertex, and the traffic is re-groomed on G. In addition, $X \oplus < x, y >$ denotes the configuration obtained by applying the twist operator $< x, y >$ to X.

The twist operation is illustrated in Fig. 2. The bold gray lines represent the lightpaths and the thin black lines represent the traffic demands, where $\Gamma = \{t_{AB}, t_{CB}, t_{BD}\}$ and $L = \{l_{AC}, l_{AB}, l_{AD}\}$. We denote by $X' = X \oplus < l_{AD}, l_{BD} >$ the configuration obtained by applying move $< l_{AD}, l_{BD} >$ to configuration X. This gives X' with $L = \{l_{AC}, l_{AB}, l_{BD}\}$ and the assignment for t_{BD} changes from $\{l_{AB}, l_{AD}\}$ to $\{l_{BD}\}$.

The time complexity is $O(2|L|(|V| - 1))$ if all the configurations in the neighborhood are checked. However, the calculation to check all configurations is excessive. In most cases, there is no advantage to applying a move to a lightpath which lies in a part of the graph with no binding capacity restrictions. Instead, we only apply moves to lightpaths which are adjacent to an overloaded lightpath. More specifically, assuming there is an overloaded lightpath l_{ij} with endpoints i and j, the lightpaths l_{iv} (l_{jv}) with one endpoint i (j) will be evaluated to see if it would improve the cost function to twist it to l_{jv} (l_{iv}), which is the move $< l_{iv}, l_{jv} >$ ($< l_{jv}, l_{iv} >$), in the hope that some traffic demands on l_{ij} will change their route to avoid using l_{ij} so that the overload can be decreased. For example, in Fig. 2, if lightpath l_{AB} is overloaded, the two moves that should be evaluated are $< l_{AD}, l_{BD} >$ and $< l_{AC}, l_{CB} >$. In this way, the calculation can be reduced to $O(2\mathcal{O}\mathcal{D})$, where $\mathcal{O}$ represents the number of overloaded lightpaths and $\mathcal{D}$ represents the average degree of the vertices in the network.

At each iteration, the best move, which leads to the minimum $f(X)$, is chosen and applied to the current configuration. In other words, the best-improvement strategy is used.

In practice, the issue of attenuation should be considered too. A constraint to handle this can be naturally added into the proposed neighborhood. If site j is too far away from site v we can simply prohibit the move $< l_{iv}, l_{jv} >$. Thus, the signal strength can be guaranteed. However, to make the problem more general, we do not consider the attenuation issue in this study.

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

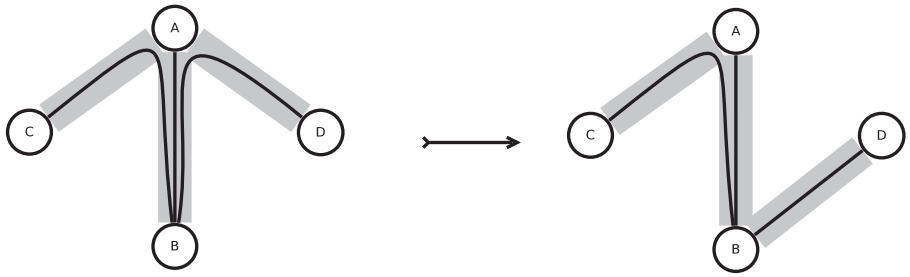


Fig. 2. Illustration of the twist operator $< l_{AD}, l_{BD} >$.

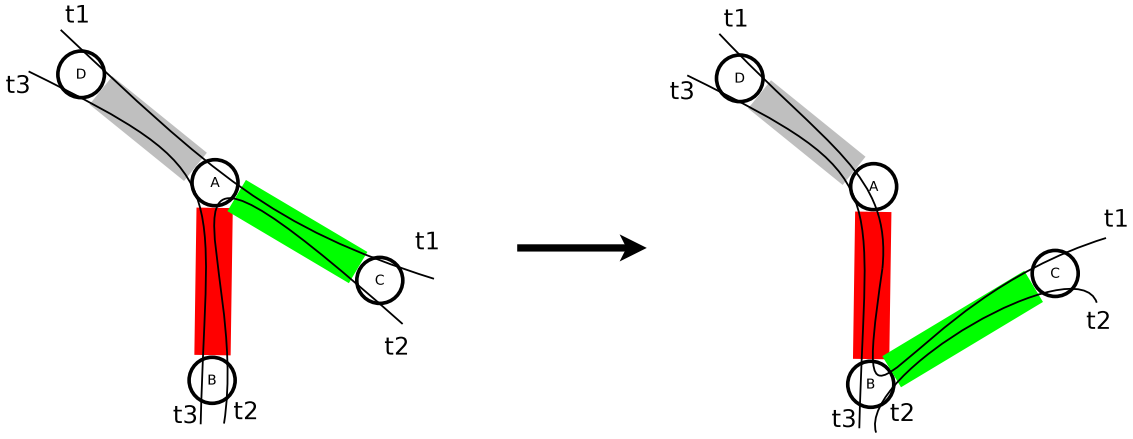


Fig. 3. Illustration of the twist operator with three lightpaths.

3.2.3. The essence of the twist neighborhood

In this section we explain the essence of the proposed twist neighborhood. For most of the configurations which are infeasible solutions, there are usually only a few overloaded lightpaths. Thus, it makes sense to only do operations related to the overloaded lightpaths to save time.

The basic idea is to keep the overloaded lightpaths unchanged, but to make changes to those that are neighbors of the overloaded ones, in the hope that some traffic demands on the overloaded lightpath will change their routes and reduce the overload. For example, Fig. 3 illustrates a part of one configuration. To make things clearer, we only present three lightpaths and three traffic demands here to represent three different kinds of lightpaths and traffic demands involved during a twist operation. Suppose lightpath AB is overloaded and lightpath AC is the one to be moved, i.e., the move operator is $< AC, BC >$. There are three kinds of traffic demands which are as follows:

- $t1$ The traffic demand only passes through the lightpath (AC) to be moved but not through the overloaded lightpath (AB);
- $t2$ The traffic demand passes through both the overloaded lightpath (AB) and the lightpath (AC) to be moved;
- $t3$ The traffic demand only passes through the overloaded lightpath (AB) but not the lightpath (AC) to be moved.

After the move operation, $t2$ is highly likely to change its route to avoid traversing AB, and in this case the overload will decrease. But $t1$ is also likely to change its route to traverse AB, and in this case the overload may be worse. Usually $t3$ is not affected by the move. Therefore, if there are more traffic demands of type $t2$ than $t1$ affected by the move, this move may reduce the overload of the corresponding lighpath.

The proposed twist neighborhood is essentially different from the two-level neighborhood (TLN) described in [32]. First, the two neighborhoods use totally different search spaces. The twist neighborhood tries to find a feasible solution for the k-NDGP problem by exploring infeasible solutions and decreasing k gradually. By contrast, the TLN directly optimizes the number of lightpaths and only accepts feasible solutions during the search. Second, the twist neighborhood gathers more heuristic information about the configuration of the current solution, which can guide the search more effectively, while the topology transformation layer in TLN is apparently "blinded" by adding or removing lightpaths randomly.

3.2.4. Completeness of the twist move

Proposition 1 states that the twist operator is sufficient to explore the whole solution space of the k-NDGP.

Proposition 1. *If X is a feasible solution for k-NDGP, then any configuration X' can be altered to yield X by performing a finite number of twist moves.*

Proof. We first define a basic move operator (p, q) which consists of deleting lightpath p from G and adding a new lightpath q to G without other restrictions. Assuming that X is a feasible solution, it is obvious that any configuration X' can be altered to yield X by performing a finite number of basic moves (p, q), only if we require that $p \in X'$ and $q \in X$. For each (p, q), if p and q share a common vertex then it is equivalent to the twist operator $< p, q >$; otherwise, it can be equal to a sequence of two twist operators $[< p, r >, < r, q >]$ only if r shares common vertices both with p and q. The lightpath r can be determined by choosing any endpoint of p and q and making these two the endpoints of r. Thus a basic operator can be replaced by one or a sequence of two twist operators, and any configuration X' can be altered to yield the feasible solution X by performing a finite number of twist moves. □

3.2.5. Evaluation of the neighborhood

Let $M(\mathcal{S})$ denote the set of the candidate moves of configuration $\mathcal{S}$. It is important to be able to evaluate the effect of a move m in $\mathcal{S}$ in an efficient way. If we simply calculate for each move m the cost function $f(X \oplus m)$ of the resulting solution to see

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

Algorithm 2 Evaluate the neighborhood.

```
1: procedure EVALUATE(X)
2:     bestMv ← NULL
3:     for each overload lightpath l_o do
4:         u = source node of l_o
5:         v = sink node of l_o
6:         for each lightpath l_u that connected to u do
7:             if F(X⊕ < l_o, l_u >) < F(X ⊕ bestMv) then
8:                 bestMv ←< l_o, l_u >
9:             end if
10:        end for
11:        for each lightpath l_v that connected to v do
12:            if F(X⊕ < l_o, l_v >) < F(X ⊕ bestMv) then
13:                bestMv ←< l_o, l_v >
14:            end if
15:        end for
16:    end for
17:    return bestMv
18: end procedure
```

whether the move is good or not, the computing time would become prohibitive. Different mechanisms have been proposed in the literature for neighborhood evaluation in a local search algorithm, among which the fast incremental evaluation has become the most used (see, e.g., [17,30,31,33,35]). This evaluation calculates the score of m, i.e. the amount by which m increases the cost of the configuration. Then the move with the least score will be considered to be the best move in the case of a minimization problem. However, the fast incremental evaluation is usually effective only for problems with simple solution structures. In NDGP, the solution structure consists of the topology of the network and the routes for each traffic. It is hard to design a fast evaluation mechanism for the neighborhood introduced above.

In this paper, an estimation method is used to evaluate each move. Here, the cost function of the configurations is evaluated rather than the score of the moves. In other words, we estimate $f(X \oplus m)$ to check whether m is good or not. The grooming subproblem formulation described in Section 3.1 is used to evaluate each neighboring configuration. However, instead of solving the MIP model we solve the linear programming relaxation of this model for each configuration in the neighborhood. The configuration with minimum $\mathcal{F}$ is chosen to be the next solution. The pseudo-code of the evaluation procedure is given in Algorithm 2, where function $\mathcal{F}(X)$ calculates the optimal linear programming objective value of the grooming problem with the network of configuration X. Here, we define $\mathcal{F}(X \oplus NULL) = \infty$. As mentioned before, we only consider the lightpaths that are connected to an overload lightpath.

3.2.6. Applying the move

The best move chosen is applied to the current configuration, which involves making changes to the network and reassigning the routes for some traffic demands. As explained in Section 3.1, the grooming solution is obtained by solving the mixed integer programming problem. Although this MIP model is tractable, it is still time consuming to solve it in each iteration.

However, during the local search procedure, only the information of the overload on each lightpath is needed for the evaluation. The execution of traffic grooming only make sense when there is no overload on any lightpath. Fortunately, the gap between the continuous grooming model and the model with binary variables is rather small (as we will see in Section 4.4). Thus it is possible to solve the linear programming relaxation at each iteration to get the estimated overload information when $\mathcal{F}$ is greater than zero, and solve the integer program when $\mathcal{F}$ equals zero to verify whether the network can satisfy all the traffic demands.

The running time can be further reduced if we terminate the MIP process when the MIP solver finds a best bound greater than 0, since a positive best bound indicates that there is impossibility to get a non-overloaded solution. Thus, the detail of the routes for each traffic demand is not necessary at this stage. Since the desired objective for the MIP process is 0, there are 4 cases for the gap value obtained by the solver: (i) no gap value available (feasible solution not found), (ii) 100% (feasible solution found with best bound of 0), (iii) less than 100% (feasible solution found with positive best bound) and (iv) un-defined value (solved optimally with objective value 0). We can set a cutoff gap to the solver with a value slightly less than 100%, for example 99.9%. In this way, the MIP solver will stop once it finds a positive best bound, reducing the total running time.

3.3. The matheuristic

In this section, we detail the tabu search procedure of the proposed algorithm. Tabu search (TS) was first proposed by Glover [12–14] and has since been applied to numerous combinatorial optimization problems (see [23,33]). A TS algorithm typically incorporates a tabu list as a recency-based memory structure to guarantee that solutions visited within a certain span of iterations will not be revisited and to introduce robustness into the search. The algorithm then restricts its attention to moves not forbidden by the tabu list. Additional techniques having their origins in tabu search and which are frequently used in metaheuristic optimization are diversification and intensification, which typically involve additional forms of memory such as frequency memory. In particular, the goal of diversification is to drive the method to explore new regions of the search space, while intensification seeks to focus the search in regions previously evaluated to be promising, and is often carried out by a local search algorithm. A detailed description of these strategies and their interactions is provided in [14].

3.3.1. Tabu search procedure

The TS_kNDG procedure has two input parameters: the initial configuration X_0 and the number of iterations *nbIter*. We denote by k the number of lightpaths.

The goal of the procedure is to explore the set of configurations by applying a series of moves to discover a configuration of minimum cost. As already mentioned in Section 3.2, a configuration X corresponds to any network with k lightpaths. The cost of this configuration is the total overload on the network and the set of moves applicable to it is defined by the twist neighborhood.

In order to escape from local optima, the TS_kNDG procedure employs a tabu search approach based on a simple tabu mechanism of forbidding a newly added lightpath to be deleted for a short period. A lightpath belonging to the tabu list is said to be tabu. Also, any move $< x, y >$ such that y belongs to the tabu list is declared tabu. After a move $mv =< x_{mv}, y_{mv} >$ is performed, the lightpath y_{mv} is inserted into the tabu list for tt iterations, where tt (the so-called tabu tenure) is determined by two parameters named *ttMin* and *ttMax*.

In addition, the tabu status of a move $< x, y >$ can be overridden by an aspiration criterion. We employ the simplest (commonly used) instance of such a criterion which disregards tabu status if applying this move leads to a configuration whose cost is smaller than the cost *fBest* of the best configuration previously found. Finally, we define a candidate *move* to be a move that is non-tabu or that satisfies the aspiration criterion. In each iteration, the tabu algorithm selects the best candidate move, i.e., the one having the lowest score.

The TS_kNDG procedure also employs a rudimentary type of diversification technique, which implements several random moves to give more diversity to the algorithm. The pseudocode of the TS_kNDG is given in Algorithm 3.

In Algorithm 3, X represents the current configuration of the tabu search procedure. X is first set to the initial configuration X_0, which is a parameter of the procedure. In each iteration, the best

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

Algorithm 3 Tabu search procedure for the k-NDGP.

```
1: INIT()
2: for iter = 1 ... nbIter do
3:     < x, y > ← SELECT_MOVE(X)
4:     APPLY_MOVE(< x, y >)
5:     tt ← RAND(ttMin, ttMax)
6:     RENDER_MOVE_TABU(< x, y >, tt)
7:     if f(X) = 0 then
8:         return (true, X, iter)
9:     end if
10:    APPLY_DIVERSIFICATION()
11: end for
12: return (false, X, iter)
```

candidate move $< x, y >$ applicable to X is selected by breaking ties randomly among those moves with the lowest score. Then, move $< x, y >$ is applied to configuration X by removing x from X and adding y to X, whereon the linear programming or mixed integer programming problem is solved depending on the current $\mathcal{F}$ (see Section 3.2). Finally, lightpath y is made tabu for tt iterations, where tt is an integer randomly chosen from the interval [*ttMin, ttMax*], and *ttMin* and *ttMax* are two parameters of the algorithm. The procedure stops when a zero-cost configuration is discovered or *nbIter* iterations have elapsed.

3.3.2. Diversification technique

In our tabu diversification procedure, we perform a simple perturbation consisting of a series of random moves whose strength is measured by the number of random moves to be performed. Therefore, the strength of the perturbation may be interpreted as the "amplitude" of a jump in the search space.

The diversification mechanism is governed by two parameters: *strength* and *perturbPeriod*. Parameter *strength* is used to set the value of the perturbation strength, while parameter *perturbPeriod* is used to determine when the perturbation is triggered. Also, the algorithm manipulates two values named X_{best} and *iterStagnation*. X_{best} denotes the best configuration; i.e., the last improved configuration visited by the procedure. Therefore, X_{best} is updated whenever $f(X) \leq f(X_{best})$. *iterStagnation* represents the number of iterations elapsed since the last improvement of the best configuration or since the last perturbation was triggered.

In our diversification approach, a perturbation is triggered when *perturbPeriod* iterations have been performed without improving the best configuration. In this case, a perturbation whose strength equals *strength* is applied to X_{best}. The pseudo-code of the diversification procedure is given in Algorithm 4. It should be noted that variables *iterStagnation, X* and X_{best} are global and are maintained through the tabu search procedure.

Algorithm 4 Procedure used to apply the diversification mechanism on each iteration.

```
1: procedure APPLY_DIVERSIFICATION()
2:     if f(X) < f(X_best) then
3:         iterStagnation ← 0
4:         X_best ← X
5:     else if f(X) = f(X_best) then
6:         iterStagnation ← iterStagnation + 1
7:         X_best ← X
8:     else
9:         iterStagnation ← iterStagnation + 1
10:    end if
11:    if iterStagnation ≥ perturbPeriod then
12:        X ← PERTURBATION(X_best, strength)
13:        iterStagnation ← 0
14:    end if
15: end procedure
```

3.3.3. Parameters of the matheuristic

In addition to the input traffic demands, the input of the matheuristic algorithm consists of the following parameters:

- the parameters *ttMin* and *ttMax* used to set the tabu list;
- the parameter *strength* used to set the value of perturbation strength;
- the parameter *perturbPeriod* used to determine when a perturbation is triggered;
- the parameter *nbIter* used for the stopping criterion.

Parameter *nbIter* depends on the available computing time. Tests performed in order to set the other parameters are presented in Section 4.2.

4. Computational results

In this section, we present experiments performed in order to evaluate the performance of our algorithm. The matheuristic algorithm was programmed in the Java 8 language. All experiments have been executed on a Windows 7 PC with an intel i5 2.7 GHz CPU and 8 GB of RAM. CPLEX 12.61 was used to solve the linear programming and the integer programming subproblems inside the algorithm.

4.1. Benchmark and algorithms used for comparison

For our tests, we have selected the benchmark proposed in [32], consisting of 22 large instances with up to 100 nodes and 500 traffic demands. Three small scale instances are added in order to make comparisons with CPLEX. For the group NDG20_t400, the bandwidth of traffic demands ranges from 1 to 16. For the remaining instances, half of the demands have a bandwidth of 1, and the remaining demands have a bandwidth of 2. For all instances, the lightpath capacity is set to be 32.These instances were randomly generated according to real world scenarios from one of the biggest telecommunication companies in China. The first number in each instance name indicates the size of the node set, and the second number indicates the size of the demand set.

We also provide computational tests comparing out proposed matheuristic with CPLEX, the two-level iterated local search (TL-ILS) algorithm proposed in [32] and the RTS algorithm for NBNDP [5]. The TL-ILS and RTS algorithms were re-programmed in JAVA and run on the same platform as the Matheuristic algorithm proposed in this paper.

4.2. Calibration of the parameters of the tabu search algorithm

In this section, we present preliminary experiments conducted to set the values of the key parameters of the matheuristic:

- Parameters *ttMin* and *ttMax* determine the range of the tabu tenure (tt). We have tested two possible values for these parameter pairs: [1, 5] and [5,10].
- Parameter *strength* is used for setting the strength of the perturbation. We have tested two possible values for this parameter: $0.1|L|$ and $0.2|L|$.
- Parameter *perturbPeriod* corresponds to the period used to apply a perturbation. We have tested two values for this parameter: $0.5|L|$ and $|L|$.

For this experiment, a subset of seven instances was used: NDG20_t100.1, NDG20_t200.1, NDG20_t300.1, NDG40_t200.1, NDG40_t200.5, NDG40_t400 and NDG100_t500. The algorithm was run 10 times on each instance and for each parameter setting. The time limit was set to 4 hours for each run. The results of these experiments are presented in Table 1. Each row in the table corresponds to a particular combination of the parameters. The first

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

Table 1
Calibration experiments for parameter setting.

Parameters			Average	
tt	*perturbPeriod*	*strength*	gap (%)	time (s)
[5,10]	$\|L\|$	0.1$\|L\|$	21.26	3662
[5,10]	$\|L\|$	0.2$\|L\|$	21.99	3084
[5,10]	0.5$\|L\|$	0.1$\|L\|$	21.38	4530
[5,10]	0.5$\|L\|$	0.2$\|L\|$	21.86	3377
[1,5]	$\|L\|$	0.1$\|L\|$	21.81	3692
[1,5]	$\|L\|$	0.2$\|L\|$	21.97	3368
[1,5]	0.5$\|L\|$	0.1$\|L\|$	22.34	3251
[1,5]	0.5$\|L\|$	0.2$\|L\|$	21.68	3223

columns indicate the combinations of the parameters. The last two columns display, for the considered combination, the gap between the lower bound (calculated as in [32]) and the average score of the solutions returned by the algorithm over the 10 runs, and the average running time to reach the best solution.

Table 1 shows that the differences in the performance of the algorithm between the different settings are small, which indicates the stability of the matheuristic. However, the parameter settings ([5, 10], 1.0$|L|$, 0.1$|L|$) lead to the best performance.

4.3. Computational results

In this section we compare the proposed matheuristic algorithm (using parameter settings ([5, 10], 1.0$|L|$, 0.1$|L|$)) with other algorithms using the full set of instances. The reference algorithms compared with the proposed matheuristic are as follows.

- CPLEX: The public general solver CPLEX using the linearization model introduced in Section 2.3.
- TL-ILS: The TL-ILS algorithm proposed in [32].
- RTS: The RTS algorithm which is the original method for solving NBNDP introduced in [5].

The computational results are shown in Table 2. Columns Best report the best results obtained by each algorithm over ten independent runs (only one run for CPLEX). The time limit was set to 8 hours. Column Average reports the average objective value obtained by the proposed matheuristic algorithm. Columns Time report the average time consumed by each algorithm to get the best result. Column Iteration reports the average number of iterations for the matheuristic to get the best result. The computational results are presented in Table 2 where dash marks indicate that the corresponding algorithm cannot get a feasible solution within the time limit.

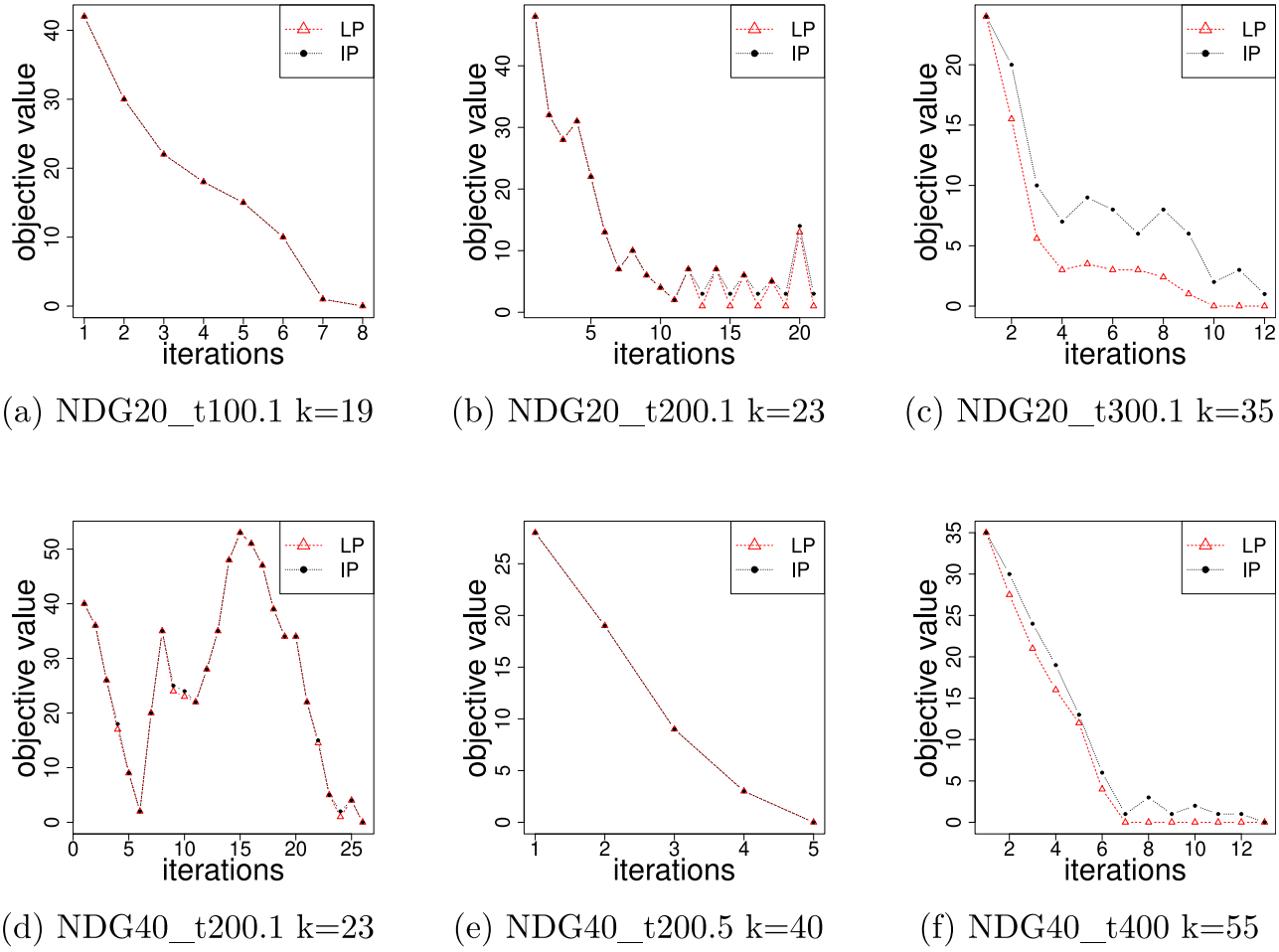
Table 2 shows that using the linearization formulation, CPLEX can obtain feasible solutions for 3 small instances with 8 nodes and 20 traffic demands. However, CPLEX fails to prove optimality even for these small instances within the time limit. For the 22 remaining moderate and large instances, CPLEX fails to give a feasible solution within the time limit or crashes due to lack of memory. We have additionally tested CPLEX using a generalized form of the flow balance equations (5) by representing them as greater than equality constraints rather than equality constraints, and find that CPLEX performs essentially the same with this version of the model. By contrast, our proposed Matheuristic obtains the same results for these 3 small instances within 12 seconds.

One can see from Table 2 that the difference between the minimum score of the matheuristic algorithm and the TL-ILS algorithm ranges from 0 to 7. The difference tends to increase with the scale of the vertex set and the number of traffic demands. For the NDG20_t300 instances the results produced by the TL-ILS are 12% to 15% larger than the ones produced by the matheuristic. For the smaller instances, the matheuristic also improves the results by 1 or 2 compared to the TL-ILS except for NDG20_t200.4. This indicates that the matheuristic algorithm outperforms the TL-ILS by an important margin.

Table 2
Computational results.

Instance	Matheuristic				CPLEX			TL-ILS			RTS-1			RTS-2		
	Best	Average	Time (s)	Iteration	Best	Time (h)	Diff	Best	Time (s)	Diff	Best	Time (s)	Diff	Best	Time (s)	Diff
NDG8_t20.1	13	13	12	140	13	4	0	13	5	0	13	1	0	13	1	0
NDG8_t20.2	14	14	2	23	14	4	0	14	1	0	14	1	0	14	1	0
NDG8_t20.3	14	14	1	2	14	3	0	14	1	0	14	1	0	14	1	0
NDG20_t100.1	19	19.0	8	9	–	–	–	20	8	1	20	2	1	20	13	1
NDG20_t100.2	19	19.0	10	12	–	–	–	21	1	2	21	2	2	20	25	1
NDG20_t100.3	19	19.0	10	12	–	–	–	20	1	1	20	32	1	20	45	1
NDG20_t100.4	19	19.0	7	20	–	–	–	19	1	1	20	97	1	20	130	1
NDG20_t100.5	19	19.0	8	10	–	–	–	21	1	2	20	95	1	20	98	1
NDG20_t200.1	23	23.0	385	157	–	–	–	24	82	1	33	490	10	33	508	10
NDG20_t200.2	23	23.0	232	89	–	–	–	24	622	1	33	151	10	33	285	10
NDG20_t200.3	23	23.0	547	318	–	–	–	24	632	1	33	324	10	33	570	10
NDG20_t200.4	22	22.0	239	79	–	–	–	22	925	0	33	242	11	33	320	11
NDG20_t200.5	23	23.0	928	296	–	–	–	25	169	2	33	142	10	33	794	10
NDG20_t300.1	34	34.5	5162	30	–	–	–	38	459	4	59	11	25	57	378	23
NDG20_t300.2	34	34.5	4699	43	–	–	–	39	631	5	58	180	24	58	512	24
NDG20_t300.3	33	33.7	4559	42	–	–	–	37	902	4	56	461	23	55	917	22
NDG20_t300.4	34	34.9	4464	26	–	–	–	39	377	5	57	215	23	56	871	22
NDG20_t300.5	34	34.4	6450	34	–	–	–	39	556	5	58	68	24	55	623	21
NDG20_t400.1	76	77.6	7161	13	–	–	–	81	1270	5	118	38	39	109	35	30
NDG20_t400.2	78	78.0	6963	17	–	–	–	80	2921	2	121	341	43	105	173	27
NDG20_t400.3	78	78.0	2634	12	–	–	–	79	3468	1	124	46	46	105	380	27
NDG20_t400.4	75	76.4	3426	12	–	–	–	80	2314	5	125	37	50	107	37	32
NDG20_t400.5	77	77.4	6883	13	–	–	–	81	2548	4	121	196	44	109	27	32
NDG40_t200.1	22	22.0	1452	755	–	–	–	24	790	2	32	813	10	32	335	10
NDG40_t200.2	22	22.3	963	943	–	–	–	25	278	3	33	87	11	32	812	10
NDG40_t200.3	39	39.0	3689	34	–	–	–	42	356	3	51	155	12	50	3985	11
NDG40_t200.4	39	39.0	239	39	–	–	–	43	472	4	51	2386	12	51	4813	12
NDG40_t200.5	39	39.0	260	41	–	–	–	43	159	4	52	56	13	52	349	13
NDG40_t400	52	52.0	8794	94	–	–	–	59	2384	7	91	786	39	90	8854	38
NDG100_t500	62	62.1	6312	19	–	–	-	69	2913	7	112	1504	50	111	5300	49

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012



(a) NDG20_t100.1 k=19 (b) NDG20_t200.1 k=23 (c) NDG20_t300.1 k=35

(d) NDG40_t200.1 k=23 (e) NDG40_t200.5 k=40 (f) NDG40_t400 k=55

Fig. 4. Accuracy of the evaluation by the linear programming model.

From the above experiment we also observe that the solution resembles a tree structure if there are relatively less traffic demands compared to the size of the node set (i.e., a smaller ratio of the number of traffic demands to the number of nodes). However, as the demands increase, tree structure networks are not sufficient to hold all the demands. Thus, our algorithm produces a structure more akin to a mesh structure in such cases involving busy networks.

The RTS method is a tabu search based algorithm, which performs quite well on the NBNDP instances according to [5]. We re-implemented and adapted the method to solve the NDGP by considering the initial network to be a complete graph and neglecting the flow costs. The input parameters for the two RTS algorithms are set as described in [5] except that the termination condition is made same as in our proposed Matheuristic algorithm. Since there is no limit on the number of edges between each pair of nodes in NDGP, two variants of the RTS are considered.

- RTS-1: The RTS algorithm, neglecting the flow costs, where at most one lightpath is allowed between each pair of nodes in the network.
- RTS-2: The RTS algorithm, neglecting the flow costs, where at most two lightpaths are allowed between each pair of nodes in the network.

From Table 2, one observes that the two RTS algorithms are comparable to the proposed Matheuristic and the TL-ILS on small instances and perform similarly to the TL-ILS on instances with up to 20 nodes and 100 traffic demands. However, as the traffic demands increase, the gap between RTS and our Matheuristic grows larger. One possible reason might lie in the fact that the neighborhood structure of RTS employs a substantial amount of heuristic information about the flow-cost on the edges while there is no flow-cost in the NDGP. Thus, RTS is outperformed by our Matheuristic and TL-ILS on NDG instances when the number of traffic demands becomes large. In addition, RTS-2 slightly outperforms RTS-1 when more computational time is allowed, since it can examine a larger search space than RTS-1. This fact also demonstrates that a strategy that does not restrict the number of lightpaths between each pair of nodes can help to find better solutions. In sum, our proposed Matheuristic outperforms the RTS algorithm for solving the NDGP.

Table 3 reports the number of parallel lightpaths and the usage percentage of lightpaths for the best solution of each instance obtained by our proposed matheuristic. If there is more than one lightpath created between the same pair of nodes, they are counted as parallel lightpaths. The lightpaths usage percentage is calculated as:

$$\frac{\sum_{l=1}^{K}\sum_{t=1}^{m} b_t(v_l^t + w_l^t)}{\sum_{l=1}^{K} Cx_l} \times 100\%$$

One observes that the lightpath usage percentage is high when the ratio between the traffic demands and the number of nodes is relatively high. The instances with less traffic demands get a lower

Table 3
Lightpaths status.

Instance	Parallel	Usage (%)	Instance	Parallel	Usage (%)
NDG20_t100.1	0	61	NDG20_t400.1	8	98
NDG20_t100.2	0	62	NDG20_t400.2	6	95
NDG20_t100.3	0	59	NDG20_t400.3	10	96
NDG20_t100.4	0	58	NDG20_t400.4	6	96
NDG20_t100.5	0	55	NDG20_t400.5	6	94
NDG20_t200.1	0	89	NDG40_t200.1	0	86
NDG20_t200.2	0	88	NDG40_t200.2	0	87
NDG20_t200.3	0	87	NDG40_t200.3	0	59
NDG20_t200.4	0	87	NDG40_t200.4	0	55
NDG20_t200.5	0	88	NDG40_t200.5	0	53
NDG20_t300.1	0	98	NDG8_t20.1	2	85
NDG20_t300.2	0	99	NDG8_t20.2	0	89
NDG20_t300.3	0	98	NDG8_t20.3	0	93
NDG20_t300.4	0	98	NDG40_t400	0	92
NDG20_t300.5	0	99	NDG100_t500	0	98

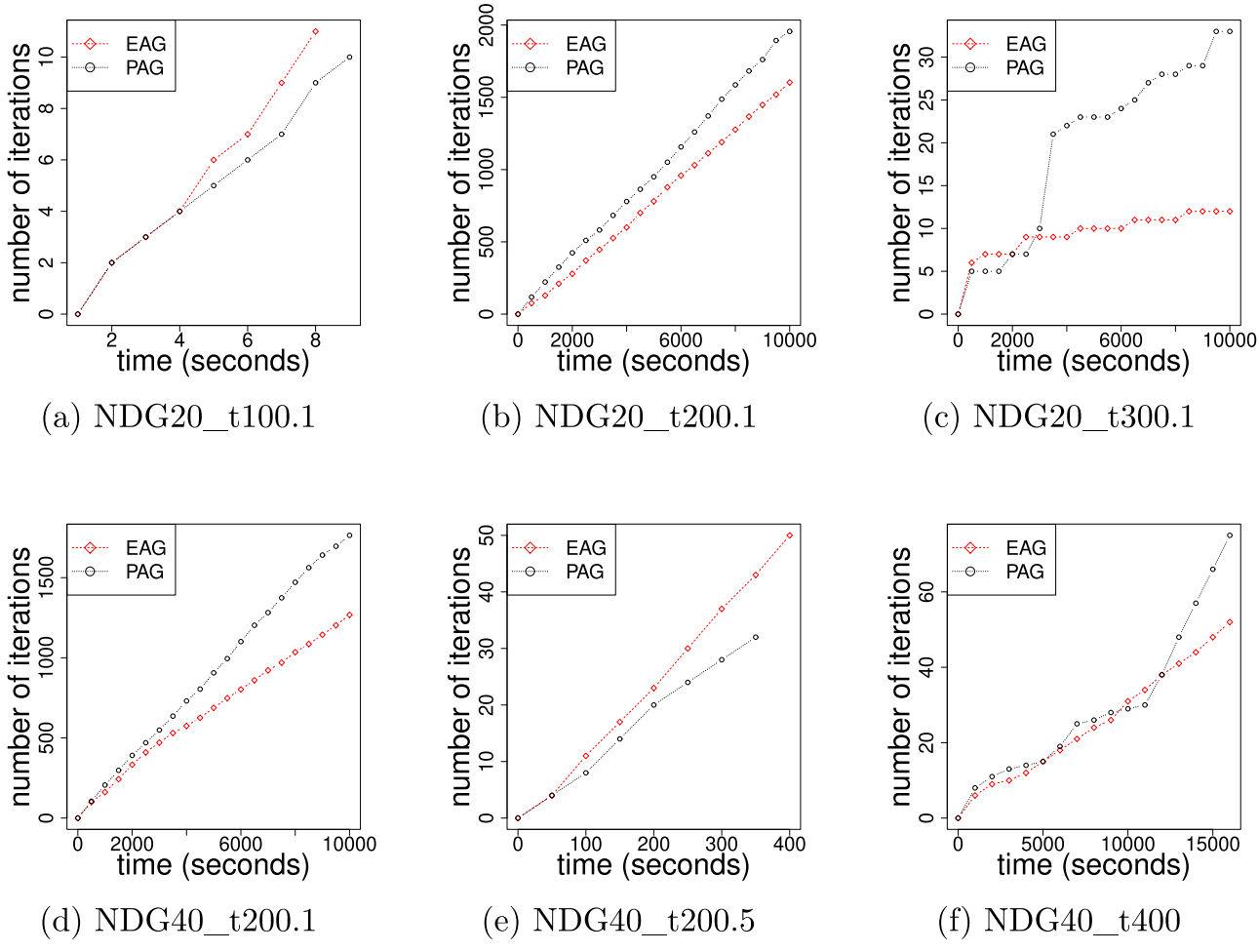


Fig. 5. Computational performance comparison.

percentage of lightpath usage due to the fact that the lightpath capacity is much larger than the bandwidth of traffic demands.

4.4. Analysis of the neighborhood evaluation technique

For a neighborhood search method, it is particularly important to be able to rapidly evaluate the neighborhood. As described in Section 3.2.5, a relaxed linear programming model is used to estimate the effects of the twist operator. Two experiments were carried out to analyze this evaluation technique on some representative instances, which are NDG20_t100.1, NDG20_t200.1, NDG20_t300.1, NDG40_t200.1, NDG40_t200.5 and NDG40_t400. Similar results were observed on other instances. The time limit for each run was set to 4 hours.

4.4.1. The accuracy of the evaluation technique

As described in Section 3.2.5, after each move the exact integer programming problem is solved only if the cost function reaches zero. In order to observe the accuracy of the evaluation, we have created a variant of the algorithm which solves the IP model after each move. The cost function values given both by the relaxed linear programming model and the integer programming model are recorded and compared to see the differences. The results are shown in Fig. 4.

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012

Table 4
Computational performance comparison.

Instance	LB	EAG			PAG		
		Best	Average	Time (s)	Best	Average	Time (s)
NDG20_t100.1	19	19	19	8	19	19	8
NDG20_t200.1	19	22	22.9	661	22	22.8	3001
NDG20_t300.1	30	35	35.4	2743	34	34.9	5506
NDG40_t200.1	19	22	22.5	468	22	22.3	969
NDG40_t200.5	39	39	39	481	39	39	327
NDG40_t400	39	55	55	5550	53	53.8	10440
NDG100_t500	39	63	63	4046	62	62.8	5594

We see that for most cases, the linear programming model correctly evaluates the actual cost of the configuration. The plots almost coincide in most of the cases, especially for some easy instances like NDG20_t100.1 and NDG40_t200.5. For other instances the gap between the original problem and the relaxation is very narrow except for instance NDG20_t300.1 which has relatively more traffic demand density than other instances (300 traffic demands on 20 nodes). This may be the reason why the evaluation is less accurate for this instance since there will be more combinations of the routings of different traffic demands with respect to the network scale and the gap between the ILP and LP models becomes large. The above experiments indicate that our proposed evaluation technique works well with the twist operator.

4.4.2. The effectiveness of the evaluation technique

Solving the integer programming problem at each iteration will certainly give the algorithm more accuracy, however the computation is very time consuming and most of the time the detailed information of the configuration is not important. As we mentioned in Section 3.2.5, only the overload amount on each lightpath is needed to produce a new solution of the problem. Thus, the integer programming problem is only solved when the evaluation method predicts a feasible solution, and then the algorithm tries to prove that prediction by using integer programming.

Here, we estimate how much time is saved by solving only the relaxed model during the search procedure. The following two strategies are considered: the proposed algorithm which only solves the LP during the search process except when the cost function value equals zero (PAG); the algorithm used in the previous section which will solve an integer programming problem at each iteration (EAG). The evaluation of the average CPU time with the number of iterations is shown in Fig. 5.

Fig. 5 discloses that the matheuristic with PAG is faster than that with EAG on difficult instances, but seems worse for simple instances. However, the difference on simple instances is negligible, since both algorithms obtain the optimal solution in very short time. For difficult instances the two algorithms exhibit a similar performance at early stages, but the gap gradually widens as k decreases.

Table 4 shows the detailed results of this experiment, revealing that PAG gets better average and best results than EAG. The table also shows that PAG uses more CPU time. However, this is because of the time limit of four hours. PAG happens to get better results in the time limit, while EAG may need much more time to get the same result as PAG.

From this experiment we learn that the k-NDGP problem gets more and more difficult as k decreases, and the proposed evaluation technique proves very helpful when k is small.

5. Conclusion

We have proposed a hybrid algorithm embedding LP and MIP within a tabu search method for tackling the NDGP, and have assessed the performance of our algorithm on instances from the literature. Computational results show that the proposed algorithm is highly effective compared to an existing iterated local search heuristic and succeeds in improving the upper bounds (best known objective function values) for all the instances except one in the benchmark set. Further analysis indicates that the neighborhood evaluation technique proposed in this paper is essential to the computational efficiency of the proposed algorithm. Possible directions for future research to enhance our algorithm include introducing more advanced local search techniques (such as advanced tabu search strategies), together with more advanced diversification frameworks (such as scatter search or path relinking) that rely more heavily on strategy in place of randomization. In addition, depending on the demand structure, there may be critical pairs of nodes handling most of the demands, therefore making the network partly resemble to a tree structure. It would be informative to analyze the demand structure and divide the whole problem into component problems which can be solved separately, thus reducing running time and further improving the solution quality.

Acknowledgments

The research was supported in part by the National Natural Science Foundation of China under grant number 61370183, 71320107001 and the program for New Century Excellent Talents in University (NCET 2013). We are grateful to Dr. Jean-François Cordeau whose comments and discussions with us have helped to improve the quality of the paper significantly.

References

[1] Agarwal Y, Venkateshan P. Survivable network design with shared-protection routing. Eur J Oper Res 2014;238(3):836–45.
[2] Agarwal YK. K-partition-based facets of the network design problem. Networks 2006;47(3):123–39.
[3] Álvaro Rubio-Largo, Zhang Q, Vega-Rodrguez MA. A multiobjective evolutionary algorithm based on decomposition with normal boundary intersection for traffic grooming in optical networks. Inf Sci (Ny) 2014;289:91–116.
[4] Azodolmolky S, Klinkowski M, Marin E, Careglio D, Pareta JS, Tomkos I. A survey on physical layer impairments aware routing and wavelength assignment algorithms in optical networks. Comput Netw 2009;53(7):926–44.
[5] Bartolini E, Mingozzi A. Algorithms for the non-bifurcated network design problem. J Heuristics 2009;15(3):259–81.
[6] Belgacem L, Charon I, Hudry O. A post-optimization method for the routing and wavelength assignment problem applied to scheduled lightpath demands. Eur J Oper Res 2014;232(2):298–306.
[7] Chatterjee BC, Sarma N, Sahu PP. Priority based routing and wavelength assignment with traffic grooming for optical networks. J Opt Commun Netw 2012;4(6):480–9.
[8] Chen B, Rouskas GN, Dutta R. On hierarchical traffic grooming in WDM networks. IEEE/ACM Trans Netw 2008;16(5):1226–38.
[9] Crainic TG, Frangioni A, Gendron B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. Discrete Appl Math 2001;112(1):73–99.
[10] Dutta R, Rouskas GN, Baldin I. Traffic grooming: balancing choice and service in optical networks. In: Advanced photonics for communications. Optical Society of America; 2014. p. PM4C.1.
[11] Gendron B, Crainic TG, Frangioni A. Multicommodity capacitated network design. In: Telecommunications network planning. Boston: Springer; 1999. p. 1–19.
[12] Glover F. Tabu search-part i. ORSA JComput 1989;1(3):190–206.

[13] Glover F. Tabu search-part II. ORSA JComput 1990;2(1):4–32.
[14] Glover F, Laguna M. Tabu search. Springer, New York; 1997.
[15] Hamid F, Agarwal YK. A polyhedral approach for solving two facility network design problem. In: Network optimization. Springer; 2011. p. 92–7.
[16] Hu JQ, Leida B. Traffic grooming, routing, and wavelength assignment in optical wdm mesh networks. In: INFOCOM 2004. Twenty-third annual joint conference of the IEEE computer and communications societies, 1; 2004. p. 495–501.
[17] Huang W, Lü Z, Shi H. Growth algorithm for finding low energy configurations of simple lattice proteins. Phys Rev E 2005;72(1):016704.
[18] Lee C, Cao X, Yoshikane N, Tsuritani T, Rhee J-KK. Scalable software-defined optical networking with high-performance routing and wavelength assignment algorithms. Opt Express 2015;23(21):27354–60.
[19] Liu M, Tornatore M, Mukherjee B. Survivable traffic grooming in elastic optical networks-shared protection. J Lightwave Technol 2013;31(6):903–9.
[20] Maniezzo V, Stützle T, Voß S. Matheuristics: hybridizing metaheuristics and mathematical programming. New York: Springer; 2009.
[21] Mukherjee B. Optical communication networks. McGraw-Hill; 1997.
[22] Palmieri F, Fiore U, Ricciardi S. Selfish routing and wavelength assignment strategies with advance reservation in inter-domain optical networks. Comput Commun 2012;35(3):366–79.
[23] Peng B, Lü Z, Cheng T. A tabu search/path relinking algorithm to solve the job shop scheduling problem. Comput Oper Res 2015;53:154–64.
[24] Rubio-Largo Á, Vega-Rodríguez MA. Applying MOEAs to solve the static routing and wavelength assignment problem in optical WDM networks. Eng Appl Artif Intell 2013;26(5):1602–19.
[25] Rubio-Largo A, Vega-Rodriguez MA, Gomez-Pulido JA, Sanchez-Perez JM. Multiobjective metaheuristics for traffic grooming in optical networks. IEEE Trans Evol Comput 2013;17(4):457–73.
[26] Saleh MA, Kamal AE. Design and provisioning of WDM networks with many–to-many traffic grooming. IEEE/ACM Trans Netw 2010;18(6):1869–82.
[27] Saleh MA, Kamal AE. Approximation algorithms for many-to-many traffic grooming in optical WDM networks. IEEE/ACM Trans Network 2012;20(5):1527–40.
[28] Vignac B, Vanderbeck F, Jaumard B. Reformulation and decomposition approaches for traffic routing in optical networks. Networks 2016. doi:10.1002/net.21672.
[29] Wang Y, Gu Q. On the complexity and algorithm of grooming regular traffic in WDM optical networks. J Parallel Distrib Comput 2008;68(6):877–86.
[30] Wang Y, Hao J-K, Glover F, Lü Z, Wu Q. Solving the maximum vertex weight clique problem via binary quadratic programming. J Comb Optim 2016;32(2):531–49.
[31] Wang Z, Lü Z, Ye T. Multi-neighborhood local search optimization for machine reassignment problem. Comput Oper Res 2016;68:16–29.
[32] Wu X, Lü Z, Guo Q, Ye T. Two-level iterated local search for WDM network design problem with traffic grooming. Appl Soft Comput 2015;37:715–24.
[33] Wu X, Yan S, Wan X, Lü Z. Multi-neighborhood based iterated tabu search for routing and wavelength assignment problem. J Comb Optim 2015;32(2):445–68.
[34] Wu X, Ye T, Guo Q, Lü Z. GRASP for traffic grooming and routing with simple path constraints in WDM mesh networks. Comput Netw 2015;86:27–39.
[35] Xu H, Lü Z, Yin A, Shen L, Buscher U. A study of hybrid evolutionary algorithms for single machine scheduling problem with sequence-dependent setup times. Comput Oper Res 2014;50:47–60.
[36] Yazar B, Arslan O, Karaşan OE, Kara BY. Fiber optical network design problems: a case for turkey. Omega (Westport) 2016;63:23–40.
[37] Zhang J, Ji Y, Song M, Zhao Y, Yu X, Zhang J, et al. Dynamic traffic grooming in sliceable bandwidth-variable transponder-enabled elastic optical networks. J Lightwave Technol 2015;33(1):183–91.
[38] Zhang S, Martel C, Mukherjee B. Dynamic traffic grooming in elastic optical networks. IEEE J Sel Areas Commun 2013;31(1):4–12.
[39] Zhang X, Qiao C. An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings. IEEE/ACM Trans Netw 2000;8(5):608–17.
[40] Zhu K, Mukherjee B. Traffic grooming in an optical WDM mesh network. IEEE J Sel Areas Commun 2002;20(1):122–33.

Please cite this article as: X. Wu, Z. Lü and F. Glover, A matheuristic for a telecommunication network design problem with traffic grooming, Omega, https://doi.org/10.1016/j.omega.2018.11.012