Applied Soft Computing 37 (2015) 715–724

Contents lists available at ScienceDirect

Applied Soft Computing

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

Two-level iterated local search for WDM network design problem with traffic grooming

Xinyun Wu, Zhipeng Lü*, Qi Guo, Tao Ye

SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, PR China

ARTICLE INFO

Article history:
Received 17 December 2014
Received in revised form 30 May 2015
Accepted 24 August 2015
Available online 3 September 2015

Keywords:
WDM mesh network
Traffic grooming
Local search
Metaheuristic

ABSTRACT

Two important problems arise in WDM network planning: network design to minimize the operation cost and traffic grooming to maximize the usage of the high capacity channels. In practice, however, these two problems are usually simultaneously tackled, denoted as the network design problem with traffic grooming (NDG). In this paper, a mathematical formulation of the NDG problem is first presented. Then, this paper proposes a new metaheuristic algorithm based on two-level iterated local search (TL-ILS) to solve the NDG problem, where a novel tree search based neighborhood construction and a fast evaluation method are proposed, which not only enhance the algorithm's search efficiency but also provide a new perspective in designing neighborhoods for problems with graph structures. Our algorithm is tested on a set of benchmarks generated according to real application scenarios. We also propose a strengthening formulation of the original problem and a method to obtain the lower bound of the NDG problem. Computational results in comparison with the commercial software CPLEX and the lower bounds show the effectiveness of the proposed algorithm.

1. Introduction

The network design problems arise in optical telecommunication where both construction and operation are significant cost drivers. A wavelength division multiplexing (WDM) network involves three layers, which are the traffic demands between each pair of nodes, the optical network representing the lightpaths carrying the traffics between each pair of nodes, and the physical network in the WDM network [1]. The physical network seldom changes when it is constructed, while the optical network varies according to different scenarios of traffic demands.

The edges in the optical network are called lightpaths, which are responsible for transferring traffics by two transceivers at the source and sink nodes. In real applications, the cost of the transceivers is a dominant cost in the WDM network. Since each lightpath corresponds to two transceivers, the number of transceivers is twice the number of lightpaths. Therefore, assigning an exclusive lightpath to each traffic is rather expensive. In general, lightpaths have much larger capacities compared to the bandwidths of traffics. Thus, a lightpath can be usually shared by several traffics.

The network design problem with traffic grooming (NDG) can be stated as: Given the physical network and the traffic demands, the objective is to design an economical optical network which is able to carry all the traffic demands. In this paper, we study the NDG problem, which aims to design a network (the optical layer of the WDM network) with the minimum number of lightpaths while satisfying the grooming constrains. We present an integer programming model for the NDG problem, which provides a representation for the optical network and the paths for all the traffics. In order to solve the NDG problem we adopt a two-level iterated local search (TL-ILS) which is a metaheuristic already applied in a wide range of applications. To strengthen the search power of the TL-ILS algorithm, we develop a tree search based neighborhood evaluation method. The proposed algorithm is applied to tackle two sets of benchmark instances, showing its high efficiency by comparing to the public software CPLEX and the lower bound. These instances are generated according to the real application scenarios, and are public for future comparisons with other reference algorithms.

2. Related works

There are numerous studies in this topic during last decades. Wang and Gu proved the NP-completeness of the traffic grooming problem in [2], implying that effective optimization algorithms are highly demanded for this challenging problem. Chen and Rouskas

* Corresponding author. Tel.: +86 2787543885.
E-mail addresses: xavier@hust.edu.cn (X. Wu), zhipeng.lv@hust.edu.cn (Z. Lü), yeetao@gmail.com (T. Ye).

http://dx.doi.org/10.1016/j.asoc.2015.08.044
1568-4946/© 2015 Elsevier B.V. All rights reserved.

presented an effective and efficient hierarchical traffic grooming framework for WDM networks [3]. Saleh and Kamal addressed the problem of designing and provisioning of WDM networks to support many-to-many traffic grooming in order to minimize the overall network cost [4]. They also introduced two novel approximation algorithms for the many-to-many traffic grooming problem [5]. A mathematical formulation of the traffic grooming and routing problem was presented in [6] by Zhu and Mukherjee, and several fast heuristics were proposed and evaluated. Hu formulated the grooming, routing and wavelength assignment problem as integer linear programming and divided it into two sub problems: The traffic grooming, routing problem and the wavelength assignment problem [7]. In [8], Thiagarajan and Somani proposed a connection admission control scheme to ensure fairness in terms of connection blocking. Srinivasan and Somani presented a theoretical capacity correlation model to calculate the blocking probability for WDM networks with constrained grooming capability [9]. In [10], Rubio-Largo and Vega-Rodriguez solved the traffic grooming problem by proposing two novel multi-objective evolutionary algorithms, showing the excellent properties of the proposed metaheuristic algorithms. In [11], Wang et al presented a survey of traffic grooming schemes for optical networks that make use of architectures, algorithms and designing techniques that impose a hierarchical structure on the network topology. Moreover, another problem highly related with the NDG problem is called the routing and wavelength assignment problem which is beyond the scope of this paper (see [12–14,32–34]) since wavelength is not a scare resource for telecommunication companies in practice.

In spite of these numerous studies, there are few works that mainly focus on the NDG problem. In most of the previous literatures, the traffic grooming problem is combined with routing and wavelength assignment problem. These two problems are usually simultaneously tackled, which may not be able to reveal the intrinsic nature of the traffic grooming problem. Since the main purpose of the traffic grooming problem is to maximize the use of the network, it is intrinsically related to the network design problem. Therefore, our study focuses on the network design problem with traffic grooming in this paper.

The NDG problem is also quite similar to the widely studied classic network design problem: Multi-commodity capacitated network design problem (MCND) [15]. Several classic relaxation methods are analyzed and applied to several formulations of a fixed charge multi-commodity capacitated network design problem by Gendron et al in [16]. In [17], Crainic et al compared the bundle and sub-gradient methods applied to the optimization of Lagrangian duals arising from two Lagrangian relaxations, showing that the bundle methods are superior to the sub-gradient approaches. They also presented a simplex based tabu search algorithm to solve the MCND. The idea of implementing simplex is then extended to simulated annealing algorithm by Yaghini et al in [18]. The current state-of-the-art heuristic for MCND was proposed by Ghamlouche et al in [19] and it was improved by the same authors by employing a path relinking approach in [20]. The MCND problem has also derived a lot of variants, see [21–23]. The instances for the variants of the network design problem can be found in [24]. Other related problems also include dynamic routing communications network design problem [25], single sink fixed charge transportation problem [26], multiband robust capacitated network design with multiple time periods [27], fault tolerant network design [28], and so on.

However, the NDG problem is different from all the variants of MCND. The network considered in NDG problem may be a complete graph, indicating that there may be much more design variables than the MCND problem. Moreover, the flows are non splittable and the design variables are all discrete in the NDG problem, while the MCND problem is mainly focused on the network design problem since its flow routing subproblem can be easily solved. These features make the NDG problem particularly challenging to solve.

Table 1
Notations used for the NDG problem.

Symbols	Description
n	The number of nodes
m	The number of traffics
C	The capacity for each lightpath
T	The set of traffics, $t \in T$, $t = (s_t, d_t, c_t)$
s_t	The source node of traffic t
d_t	The sink node of traffic t
c_t	The bandwidth of traffic t
β_{tj}	Indicator of the source node and the sink node of a traffic $\beta_{tj} = \begin{cases} 1, & \text{if } j = d_t \\ -1, & \text{if } j = s_t \\ 0, & \text{otherwise} \end{cases}$
x_{ij}	The number of lightpaths established between nodes i and j
y_{tij}	Binary decision variable for whether traffic t passes through lightpath between nodes i and j
Z	The objective value for the NDG problem

3. The NDG problem

3.1. Problem description

The network design problem with traffic grooming is defined as follows. Given a node set $V = \{1, 2, \ldots, n\}$ and a traffic set T, in which the elements are tuples (denoted as (s_t, d_t, c_t)) representing the source and sink nodes as well as the bandwidth of each traffic. The objective is to design an optical network (denoted as $G = \{V, L\}$) and groom all the traffics in set T with the minimum number of lightpaths. The edges in G represent the lightpaths and each lightpath carries traffics no more than its capacity limit C.

Fig. 1 dipicts an example: Given a node set V with 4 nodes and a traffic set T with 5 traffics of bandwidth 1 and assuming that the capacity of each lightpath is 4, the straightforward solution of this problem is to construct a network with 5 lightpaths, each carrying one traffic, as shown in Fig. 1a. The grey bold solid lines represent the lightpaths connecting each pair of nodes, while the color lines represent the routes for all the traffics. As an alternative, we can groom traffic $(A, C, 1)$ to lightpaths AD and DC and groom traffic $(B, C, 1)$ to lightpaths AB, AD and DC, which means that traffic $(A, C, 1)$ routes $A-D-C$ and traffic $(B, C, 1)$ routes $B-A-D-C$. Thus, only 3 lightpaths are needed to groom all the 5 traffics, saving 2 lightpaths compared to the previous solution. Obviously, the network with 3 lightpaths is the optimal solution for this example.

3.2. Problem formulation

The WDM network design problem consists of designing a WDM network with the minimum number of lightpaths and grooming all the traffics in set T on the lightpaths. A candidate solution is represented by two matrices: $n \times n$ matrix $[X]$ where x_{ij} represents the number of lightpaths established between nodes i and j. $m \times n \times n$ matrix $[Y]$ where $y_{tij} = 1$ if traffic t traverses lightpath connecting nodes i and j. The graph represented by $[X]$ is an undirected graph, while the graph denoted by $[Y]$ is a digraph, which represents the paths of all traffics on the WDM network. The symbol and variable definitions are presented in Table 1. Given these notations, we can describe the WDM network design problem in a formal way as follows:

minimize: $Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}$

subject to:

$$x_{ij} = x_{ji} \quad i, j = 1, 2 \ldots n \qquad (1)$$

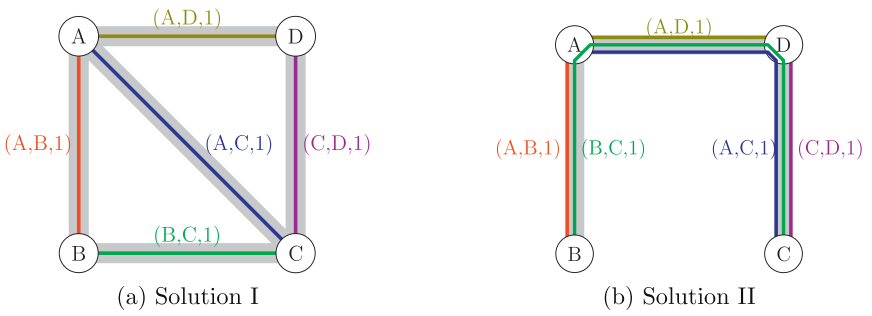

Fig. 1. An example for the NDG. (For interpretation of the references to color in the text, the reader is referred to the web version of the article.)

Table 2
Variables used in the TL-ILS algorithm.

Symbols	Description
T_u	The set consisting of the traffics which have not been groomed
O_{ij}	The overload amount of the lightpath connecting nodes i and j
O_t	The overload amount produced by traffic t
f	The objective for procedure *grooming*
Δf	The incremental change of f
L_u	The set consisting of the indexes of lightpaths
l	Index of a lightpath
c_l	The remaining bandwidth of lightpath l
t	Index of a traffic
P	A series of lightpaths that represent a path for a traffic
mv	A move which can lead to a new solution denoted by $S \oplus mv$ by applying it to solution S
$N_l([x])$	The neighborhood of the current solution in *Main Local-search*
$N_t([y])$	The neighborhood of the current solution in *Grooming*
u_{ij}	The used amount of the lightpath connecting nodes i and j
$\mathbf{0}$	Zero matrix

$$\sum_{i=1}^{n} y_{tij} - \sum_{i=1}^{n} y_{tji} = \beta_{tj} \quad \forall t \in T \quad j = 1, 2 \ldots n \tag{2}$$

$$\sum_{t \in T} c_t (y_{tij} + y_{tji}) \le C x_{ij} \quad i = 1, 2, \ldots, n-1 \quad j = i+1 \ldots n \tag{3}$$

$$x_{ij} \in \mathbb{N}_0 \quad i, j = 1, 2 \ldots n \tag{4}$$

$$y_{tij} \in \{0, 1\} \quad t \in T \quad i, j = 1, 2 \ldots n \tag{5}$$

The objective Z is to minimize the total number of lightpaths. Constraints (1) guarantee that the graph that $[X]$ represents is undirected. Constraints (2) require that the route of a traffic is an available simple path. Constraints (3) ensure that every traffic passes through feasible lightpaths, and each lightpath can only carry traffics not more than its capacity. Constraints (4) and (5) denote the integer and binary restrictions of the decision variables, respectively.

4. Solution method

In order to clearly describe our proposed Iterated Local Search algorithm [29] for the WDM network design problem, we first define some variables as shown in Table 2.

The proposed TL-ILS algorithm follows a general framework which consists of three phases: Initialization, intensification, and diversification, as described in Algorithm 1. The iterations are in lines 4–14 and stop when some termination criterion, such as the time limit, is satisfied. The best solution found so far is kept as the result. Line 3 is the initialization phase, while line 5 is the intensification phase. When the search reaches a local optimum solution, the diversification phase (perturbation in line 13) is called to prevent the algorithm from being trapped in the local optimum trap. In the following sections we describe these three phases in more details.

Algorithm 1. The main algorithm framework

```
1:  procedure MAIN($V$, $T$)
2:    $best\_Z \leftarrow \infty$
3:    $[X], [Y] \leftarrow$ INITIALIZATION ($[X], [Y], T$) /* Initialization Phase */
4:    repeat
5:      $[X], [Y] \leftarrow$ LOCAL SEARCH($[X], [Y], T$) /* Intensification Phase */
6:      $Z \leftarrow$ COUNTLIGHTPATHS($[X]$)
7:      if $Z \le best\_Z$ then
8:        $[X]_b, [Y]_b \leftarrow [X], [Y]$
9:        $best\_Z \leftarrow Z$
10:     else
11:       $[X], [Y] \leftarrow [X]_b, [Y]_b$
12:     end if
13:     $[X], [Y] \leftarrow$ PERTURBATION($[X], [Y], T$) /* Diversification Phase */
14:   until termination condition is met
15:   EXPORTSOLUTION($[X]_b, [Y]_b$)
16: end procedure
```

4.1. Initialization

Algorithm 2. Initialization

```
1:  procedure CONSTRUCTION($[X], [Y], T$)
2:    $[Y] \leftarrow \mathbf{0}$
3:    $[X] \leftarrow \mathbf{0}$
4:    for all $t \in T$ do
5:      if $\exists P$ connecting $s_t$ and $d_t$ AND $c_l \ge c_t \; \forall l \in P$ then
6:        ASSIGN($[Y], t, P$)
7:        $Flag \leftarrow true$
8:      else
9:        $Flag \leftarrow false$
10:     end if
11:     if $Flag = flase$ then
12:       $x_{s_t d_t} \leftarrow x_{s_t d_t} + 1$
13:       $x_{d_t s_t} \leftarrow x_{d_t s_t} + 1$
14:       $P \leftarrow$ {The new $l$ connecting $s_t$ and $d_t$}
15:       ASSIGN($[Y], t, P$)
16:     end if
17:   end for
18:   return $[X], [Y]$
19: end procedure
```

A feasible solution is iteratively generated in the *Initialization* phase. At each iteration, one traffic is chosen and assigned a path on the network using a greedy heuristic and a new lightpath is introduced for the traffic if the assignment fails. The detailed procedure is shown in Algorithm 2.

Notice that the traffics are shuffled at the beginning of the initialization. It is also noteworthy that in the above procedure, we need to check the feasibility of the paths between two nodes. Thus, an algorithm is required to examine whether a path between two nodes is a simple path. This algorithm will be described in Section 4.3.1.

4.2. Local search procedure

Algorithm 3. Local search procedure

```
1:  procedure LOCAL SEARCH([X], [Y], [N])
2:    $L_u \leftarrow$ INDEXOFLIGHTPATHS([x])
3:    while $L_u \neq \phi$ do
4:      $[X]_b, [Y]_b \leftarrow [X], [Y]$
5:      $l \leftarrow$ CHOOSELIGHTPATH($L_u$)
6:      $L_u \leftarrow L_u - l$
7:      $T_u \leftarrow$ Traffics passing through $l$
8:      $[X], [Y] \leftarrow$ REMOVELIGHTPATH([X], [Y], $l$)
9:      $flag \leftarrow$ GROOMING([X], [Y], $T_u$)
10:     if $flag$ = TRUE then
11:       EXPORTSOLUTION([X], [Y])
12:     else
13:       $[X], [Y] \leftarrow [X]_b, [Y]_b$
14:     end if
15:   end while
16: end procedure
```

After a feasible initial solution is constructed, the *local search* procedure (intensification phase) tries to transform the topology in order to decrease the number of lightpaths while satisfying all the constraints. The *local search* phase is composed of two local search algorithms in a multi-level architecture. The high level local search algorithm is a topology transformation algorithm, while the low level local search algorithm (Section 4.3) validates the feasibility of the current solution. Algorithm 3 presents the pseudo-code of the *local search* phase.

In a local search procedure, a solution S can be converted to a new solution by applying a move mv on it, which can be denoted by $S \oplus mv$. Let $M(S)$ be the set of all possible moves which can be applied to S, then the neighborhood of S is defined by: $N(S) = \{S \oplus mv | mv \in M(S)\}$.

In our TL-ILS algorithm, two neighborhood structures are used respectively for the two local search algorithms. The neighborhood structure $N_l([X])$ used in *local search* for the high level local search is composed of all possible moves of *lightpath-deletion*, where *lightpath-deletion* is defined as removing one lightpath from the current virtual topology. Therefore, the size of the neighborhood $N_l([X])$ is bounded by $O(\mu)$, where μ is the number of lightpaths. The neighborhood structure for the low level local search will be introduced in Section 4.3.

Let L_u denote the indexes of all the lightpaths. At each iteration, one lightpath from L_u is chosen and it is removed from the current solution tentatively. Then, the traffics carried by the removed lightpath are re-assigned to the network by running the low level local search procedure *grooming*. Subroutine *grooming* assigns these traffics to the network, ensuring that the capacity constraint is satisfied (constraint (3)). An improved solution is obtained if subroutine *grooming* returns TRUE. Otherwise, we restore the moved lightpath, and try to delete another lightpath.

In this high level local search we accept a candidate solution in the current neighborhood immediately if the deletion of one lightpath l can satisfy all the constraints, which means we use a first-improvement strategy based simple descent algorithm. Then, L_u is updated by removing lightpath l. The above procedure is repeated until it fails for the deletion of any lightpath in L_u.

4.3. Grooming

The procedure *grooming* is the key component of our proposed algorithm. It is the main factor that affects the efficiency of our algorithm. As a subroutine of *local search*, the *grooming* procedure can be considered as a low level local search algorithm in our TL-ILS algorithm. Once the topology is changed by removing one lightpath, the *grooming* procedure is called to verify if the remained network can still satisfy the capacity constraints. The main purpose of *grooming* is to groom traffics to lightpaths such that all the lightpaths are not overloaded.

In the *grooming* procedure, the objective function (denoted as f) is defined as the violation of the capacity constraints for all the lightpaths:

$$f = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} O_{ij}$$

where O_{ij} represents the overload amount of the lightpath connecting nodes i and j.

$$O_{ij} = \max\left(\sum_{t \in T} c_t (y_{tij} + y_{tji}) - Cx_{ij}, 0 \right)$$

One observes that $f_g = 0$ corresponds to a network which satisfies all the capacity constraints.

The neighborhood used in the *grooming* procedure, which is denoted as $N_t([Y])$, is composed of all feasible moves of *RouteChange*, where a *RouteChange* move changes the route of one traffic to another candidate route of this traffic. Applying a *RouteChange* move to traffic t in the solution consists in modifying the values of y_{tij} $(1 \le i \le n,\ 1 \le j \le n)$. Therefore, the size of neighborhood $N_t([Y])$ is bounded by $O(m \cdot k)$ where $m = |T|$ and k is the maximum number of candidate routes for one traffic.

For example, Fig. 2 depicts how *RouteChange* works for traffic $t = (B, C, 1)$. There are 3 candidate routes for traffic t, which are B-C, B-A-C and B-A-D-C. In the current solution, the route for traffic t is B-C (illustrated by green line in Fig. 2a). The *RouteChange* mv_g changes the route of t to another candidate route of t which is either B-A-C (Fig. 2b) or B-A-D-C (Fig. 2c).

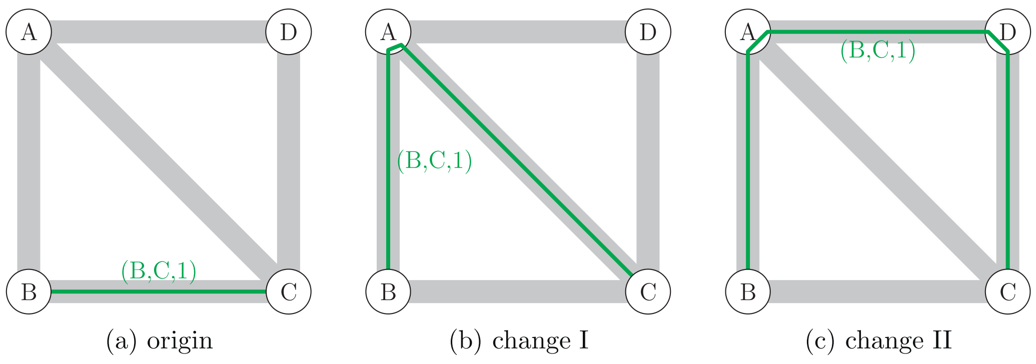

Fig. 2. *RouteChange* neighborhood move illustration.

In the *grooming* procedure, we first assign the traffics in T_u with random paths. Then, we use a best-improvement based simple descent algorithm in order to get a feasible solution. That is to say, we choose the candidate solution in $N_t([Y])$ with the minimum Δf value. The procedure stops when $f = 0$ or the minimum Δf value is non-negative.

The design of neighborhood structure is a major challenge in a local search algorithm for solving complex optimization problems like NDG. In our algorithm, this challenge is solved by dividing the original problem into two levels: One high level local search solves the lightpath minimization problem while the low level local search deals with the traffic flow problem on the lightpath network. This hierarchical local search algorithm provides a clear understanding of the original problem.

4.3.1. Evaluation of the candidate routes

Given the neighborhood structure definition in a local search algorithm, neighborhood evaluation method is another very important factor affecting the efficiency of the algorithm. In order to find the best move for each iteration, all candidate routes for each traffic should be examined. Although there are algorithms that can solve the K shortest path problem in polynomial time [30], the time complexity is still huge if we examine the candidate routes one by one.

Algorithm 4. Find move for one traffic

```
1:  procedure FINDMOVE$t$
2:    $O_t \leftarrow$ the overload caused by traffic $t$
3:    if $O_t = 0$ then
4:      return
5:    end if
6:    $cutValue \leftarrow O_t$
7:    $Queue \leftarrow \Phi$
8:    $Queue.push(s_t)$
9:    $TravelTree.root \leftarrow s_t$
10:   while $Queue \neq \Phi$ do
11:     $u \leftarrow Queue.pop()$
12:     $r' \leftarrow$ {The node sequence from $TravelTree.root$ to $u$}
13:     for all $v$ that is connected to $u$ do
14:       $O_t^* \leftarrow$ the overload will be generated if $t$ traverse $r'$
15:       if $O_t^* < cutValue$ then
16:         if $v = d_t$ then
17:           $cutValue \leftarrow O_t^*$
18:           $r \leftarrow r' \cup \{v\}$
19:           $\Delta f \leftarrow O_t^* - O_t$
20:         else
21:           if $v \notin r'$ and $|r'| <$ hop limit then
22              TravelTree.node(u).attachChild(v)
23:             Queue.push(v)
24:           end if
25:         end if
26:       end if
27:     end for
28:   end while
29:   return $r$, $\Delta f$
30: end procedure
```

In this paper, the evaluation is implemented by a tree search based algorithm as shown in Algorithm 4, which is based on a breadth-first search strategy. For each traffic, the cut value is initially set to be the overload caused by this traffic, and the value will be updated if a better route is found. The search will also be cut if the search depth is too deep, which means we do not consider very long route.

Consider an optical network example as shown in Fig. 3. The edges represent the lightpaths in the network. The 2-tuples on the lightpaths respectively represent the total traffic demand and the capacity for the corresponding lightpath. Assuming that there is a traffic t with bandwidth 2 that traverses A-B-C-D. From Fig. 3,

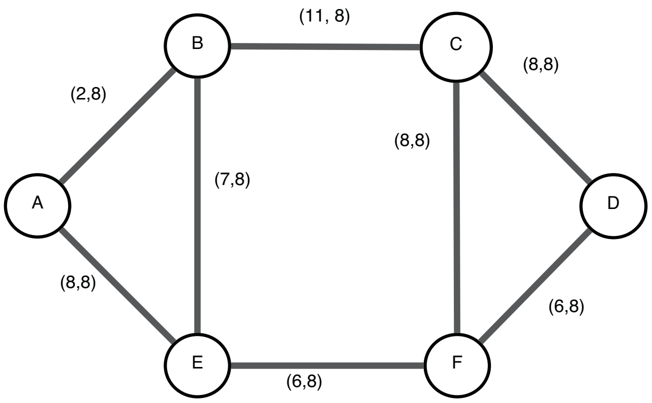

Fig. 3. The network for illustrating the evaluation of candidate routes.

one observes that the overload value caused by t is 2. Then, the evaluation for traffic t can be shown in Fig. 4.

Specifically, we implement a depth-first tree search from the source node of traffic t to evaluate all the possible candidate paths for traffic t. The steps are as follows:

1. From node A we choose an adjacent node to be the next node to examine. Here, we chose E and the search tree is as shown in Fig. 4a. Then, the number of conflicts caused by t will be at least 2, for A-E is already fully loaded. Since this value is equal to the one caused by the original route of t, the complete route belonging to this branch will produce no better results than the original one. Therefore, it is not necessary to proceed with this branch.
2. The search returns to A. Starting from B, which is another node adjacent to A, the search tree is as shown in Fig. 4b. This branch does not produce any conflict up to now, so we can continue.
3. Choose a node adjacent to B. Here, assuming C is chosen as shown in Fig. 4c. Similarly, branch A-B-C will cause 2 conflicts. It is not necessary to search further with this branch.
4. Backtrack to B, and choose node E as shown in Fig. 4d. Although branch A-B-E will cause 1 conflict, it is still better than the original path.
5. Because A is already in the current branch, we choose F to be the next node, as shown in Fig. 4d.
6. Choose node C to be the next node, as shown in Fig. 4e. Here, the branch A-B-E-F-C will cause 3 conflicts, greater than the original path. So, the search is stopped.
7. Backtrack to F, Choose D to be the next node. D is the sink node of traffic t, and the current path only produces 1 conflict, so a better path for t has been found.
8. At this point, we have finished exploring all the candidate paths for traffic t, and the best move for traffic t is to change its route to A-B-E-F-D.

It should be noteworthy that this search procedure is much faster than evaluating all the candidate paths, since there is a large number of paths that have common sub-paths. One observes that each of these common sub-paths is only examined once in the proposed search procedure, no mater how many paths there are. However, if we examine all the candidate paths one by one, the number of examinations will be fixed to the number of paths. For the example in Fig. 3, there are 6 candidate paths for traffic t and there are 3 hops in the shortest path. Therefore, the basic operation will be executed at least 18 times if we evaluate all the candidate paths one by one. However, using the proposed tree search method, the operation will be executed for only 7 times (equal to the number of edges in the search tree). The advantage of using

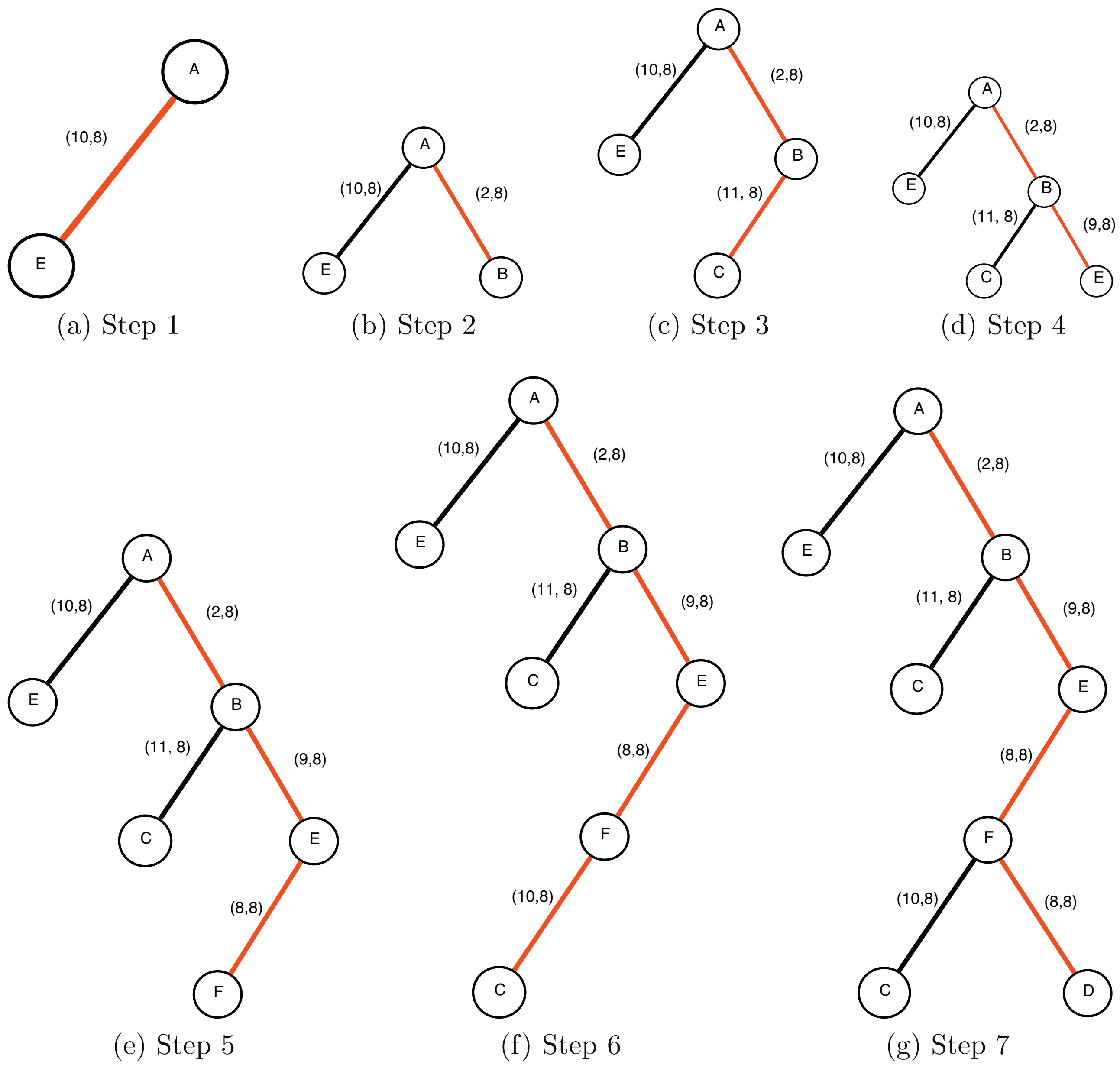

Fig. 4. Evaluation for traffic *t*.

this tree search strategy is reflected in both time efficiency and space aspects. Moreover, the hop limit can be controlled more easily by using this tree search algorithm which can be considered as an additional advantage.

4.3.2. *The incremental evaluation for grooming procedure*

Although the tree search based evaluation procedure can greatly reduce the time in the exploration of neighborhoods, the time complexity is still high when considering all the candidate paths for all the traffics at each local search iteration. In this section, we propose an incremental evaluation technique to speed up the evaluation of each move value at each local search iteration.

One observes that the overload values caused by each traffic are not totally independent. Specifically, if one traffic changes its route, the overload value caused by other traffics may also change. However, by changing the route for one traffic, there are only a few number of lightpaths which are associated with this move. Thus, for those neighborhood moves whose overload values are not affected by the move we can avoid reevaluating them.

In order to evaluate the overload value caused by each traffic in a fast manner, our *grooming* procedure maintains an $n \times n$ matrix U, from which the move values for all the possible moves can be quickly calculated. The value u_{ij} measures the usage of the lightpath connecting nodes i and j. Note that this matrix is only initialized at the beginning of each grooming procedure and will be updated in a fast manner after each iteration.

At each iteration, the overload contribution caused by traffic t to the current solution can be calculated as:

$$O_t = \sum_{ij \in p_t} \max(\min(u_{ij} - C, c_t), 0)$$

where p_t represents the current route for traffic t.

If traffic t changes its path to p'_t, the overload involved in the new path can be calculated as:

$$O^*_t = \sum_{ij \in p'_t} \max(\min(u'_{ij} - C, c_t), 0)$$

where u'_{ij} is calculated as:

$$u'_{ij} = \begin{cases} u_{ij}, & \text{for}\,(i,j) \in p_t \\ u_{ij} + c_t, & \text{for}\,(i,j) \notin p_t \end{cases}$$

Then, the incremental value of a move in *grooming* procedure can be determined as follows:

$$\Delta f = O^*_t - O_t$$

After moving traffic t from p_t to p'_t, only the u_{ij} values which are involved in path p_t or p'_t need to be updated:

$$u_{ij} \leftarrow u_{ij} + c_t, \quad \text{if}\,(i,j) \in p_t$$

or

$$u_{ij} \leftarrow u_{ij} - c_t, \quad \text{if}\,(i,j) \in p'_t$$

Usually, the route for each traffic will not be too long. Therefore, there would not be many elements that need to be updated at each local search iteration. In this way, we can evaluate the neighborhood moves for the *grooming* procedure in a fast way. In sum, this technique helps us to greatly enhance the efficiency of neighborhood evaluation in our *grooming* procedure.

4.4. Perturbation operator

The perturbation operator plays a crucial role within TL-ILS since the local search alone cannot escape from a local optimum. TL-ILS thus tries to move to new basins of attraction by applying a perturbation operator. Since there are two local search procedures in the hierarchical structure of our proposed TL-ILS algorithm, we employ two perturbation procedures, which are respectively the traffic perturbation and the topology perturbation, to help the search jump out of the local optimum trap. These two perturbation procedures thus constitute the diversification phase of our TL-ILS algorithm.

The traffic perturbation is applied in the *grooming* procedure, when the low level local search reaches a local optimum. It is implemented by giving all the overloaded traffics new random candidate routes.

The topology perturbation, which is significantly stronger than the traffic perturbation, aims at changing the topology of the lightpath network. It is applied as soon as the *local search* phase reaches a local optimum, i.e., each lightpath has been attempted to be deleted but no one is successful. The idea of the topology perturbation is to reassign all the traffics to the topology in a shuffled order and to introduce new lightpaths when the assignment fails, as shown in Algorithm 5.

Algorithm 5. Topology perturbation

```
1:  procedure TopologyPerturbation([X], [Y], T)
2:      [Y] ← 0
3:      shuffleTrafficsT
4:      for all $t \in T$ do
5:          if $\exists P$ connecting $s_t$ and $d_t$ AND $c_l \geq c_t\ \forall l \in P$ then
6:              Assign[Y], t, P
7:              Flag ← true
8:          else
9:              Backup current solution as X'
10:             Flag ← Grooming([X], [Y], {t})
11:         end if
12:         if Flag = flase
13:             Restore current solution according to X'
14:             $x_{s_t d_t} \leftarrow x_{s_t d_t} + 1$
15:             $x_{d_t s_t} \leftarrow x_{d_t s_t} + 1$
16:             P ← {The new $l$ connecting $s_t$ and $d_t$}
17:             Assign([Y], t, P)
18:         end if
19:     end for
20:     return [X], [Y]
21: end procedure
```

Our proposed TL-ILS algorithm uses these two perturbations as soon as a search stagnation is detected, leading to different levels of diversification. As we will see in the next section, TL-ILS is able to obtain highly competitive results on the tested benchmarks.

5. Experimental results

In this section, we report intensive experimental results of the proposed TL-ILS algorithm on two sets of totally 24 instances and compare our results with those obtained by the commercial software CPLEX and the lower bound.[1]

5.1. Experimental protocol

The presented TL-ILS algorithm is programmed in C++ and compiled using GNU GCC on a PC running Windows 7 with 3.1 GHz CPU and 6.0 Gb RAM. Two sets of problem instances are considered in our experiments, in total constituting 24 instances. The first set of benchmarks is composed of 2 small instances of size $n = 8$ and $m = 15 - 50$. The second set of benchmarks consists of 22 large problem instances which are generated according to the real application scenarios with size $n = 20 - 100$ and $m = 100 - 500$. Since we have agreement with our industrial partner, real world instances are not directly used in our experiments. However, this second set of instances can reflect many characteristics of real world scenarios. To obtain our computational results, each instance is solved 20 times independently with different random seeds. Each run is stopped when the running time reaches its time limit.

5.2. The lower bound

In order to assess the effectiveness of our TL-ILS algorithm, we propose a method to obtain the lower bound by introducing a strengthened NDG model and solving it by employing the public software IBM ILOG CPLEX optimization studio 12.4.

For the first two instances with small size, CPLEX is easily able to get the optimal solution, while for the large size instances, the problem is too large to be solved to optimality by CPLEX. For these large instances, the lower bound for the problem can be obtained by relaxing the integer restriction of the variables of the original model. However, the quality of this lower bound is not as good as

[1] The tested instances can be found online (https://bitbucket.org/HUST-smartLab/groominginstances) or can be obtained from the authors.

Table 3
Computational results for small-scale instances.

Instance	n	m	CPLEX	Time (s)	TL-ILS	Time (s)
s1	8	15	7	134	7	1
s2	8	20	9	193	9	1

expected. In the sequel, we show how our model can be strengthened in several ways.

Firstly, we know that each traffic should not traverse a lightpath which does not exist, which is already considered by constraints (3). However, we can further enhance it by

$$y_{tij} - x_{ij} \le 0 \quad \forall t \in T \quad i, j = 1, 2, \ldots, n \tag{6}$$

Secondly, to prevent rings from appearing in the routes of the traffics, we have:

$$y_{tij} + y_{tji} \le 1 \qquad \forall t \in T \quad i, j = 1, 2, \ldots, n \tag{7}$$

$$\sum_{i=1}^{n} y_{tij} \le 1 \quad \forall t \in T \quad j = 1, 2, \ldots, n \tag{8}$$

$$\sum_{j=1}^{n} y_{tij} \le 1 \quad \forall t \in T \quad i = 1, 2, \ldots, n \tag{9}$$

To present another valid inequality for NDG, assume n' nodes are involved in the traffic set T. Then, these n' nodes should be in the network of the final solution. We know that the minimum graph containing n' nodes should have at least $n' - 1$ edges. Mathematically, we have:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} \ge n' - 1 \tag{10}$$

Therefore, a strengthened formulation for the NDG problem can be obtained by constraints (1)–(10). We can obtain the lower bound by relaxing the integer restriction of the variables of this strengthened formulation by removing constraints (4) and (5).

5.3. *Results for small-scale instances*

There are 8 nodes in both instances. The traffic set for each instance is generated by choosing pairs of nodes randomly. The lightpath capacity is set to be 4, and the bandwidth for each traffic is equal to 1.

The results for this set of instances are shown in Table 3. The column CPLEX represents the objective value given by CPLEX. The column TL-ILS represents the best objective value for each instance obtained by our TL-ILS algorithm.

The solutions for instances s1 and s2 are illustrated in Fig. 5. In Fig. 5, the grey bold lines represent lightpaths, while the color curves represent traffics. One observes that the traffics are well packed in these lightpaths, showing the high utilization of lightpaths. In addition, our algorithm is able to obtain the objective value obtained by CPLEX, which means our algorithm is able to obtain the optimal solutions for these two small instances, showing the effectiveness of our proposed TL-ILS algorithm. In addition, the proposed TL-ILS algorithm obtains the optimal result for each instance within 1 second, while CPLEX takes several minutes to obtain the optimal solution.

5.4. *Results for large-scale instances*

In this section, 22 large instances with up to 100 nodes and 500 traffics are used. For each instance, half of the traffics have a bandwidth of 1, and the remaining traffics have a bandwidth of 2. For all the instances, the lightpath capacity is set to be 32. These instances are randomly generated according to the real world scenarios of our industrial partner, one of the biggest telecommunication companies in China. The characteristics of these instances reflect the real-life applications. The properties are shown from the name of each instance. The first number in each instance name indicates the size of the node set, and the second number indicates the size of traffic set.

Three algorithms are compared among each other in this section. *TL-ILS* is our proposed TL-ILS algorithm. *Greedy* is the initial solution generation algorithm used by our TL-ILS algorithm. *HeuristicG* is an advanced greedy algorithm with some heuristics, which uses the same framework as *Greedy*. The only difference with *Greedy* is that at each step it runs the grooming procedure in order to readjust the traffics into the network when the assignment fails. If it is still unsuccessful, a new lightpath connecting the source and the sink nodes of the traffic will be introduced.

The computational results are shown in Table 4. Column LB gives the upward rounded value of the lower bounds for each instance. Column Best reports the best results obtained by our TL-ILS algorithm over 20 independent runs and the percentages in the brackets are the probability to obtain this best objective value. Column Avg reports the average results. Column time reports the average time in seconds for the algorithm to get the best results. All the average values are the smallest integral value that is greater than or equal to the specified decimal value. Since the *Greedy* algorithm is deterministic, it gives the same results for all runs. Thus, we do not give the average value. We set a time limit of 30 min for the instances with 20 nodes and 4 h for the larger ones. For CPLEX, the time limit is set to be 10 h to calculate the lower bound. If CPLEX fails to get the result in the time limit, the value $n' - 1$ is used as the lower bound.

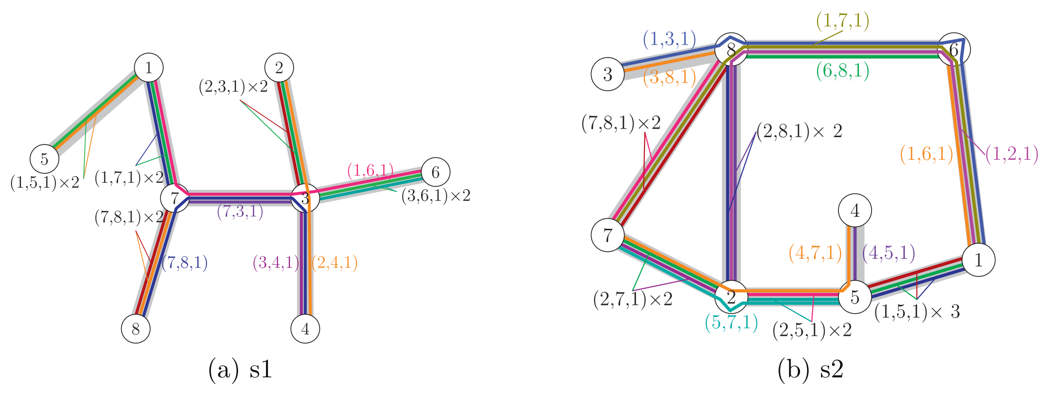

Fig. 5. The solution illustration for the two small instances.

Table 4
Results for the large-scale instances.

Instance	LB	TL-ILS			Greedy		HeuristicG		
		Best	Avg	Time	Best	Time	Best	Avg	Time
NDG20_t100.1	19	20 (20%)	21	8	23	1	21 (100%)	21	1
NDG20_t100.2	19	21 (100%)	21	1	23	1	22 (100%)	22	1
NDG20_t100.3	19	20 (100%)	20	1	21	1	21 (100%)	21	1
NDG20_t100.4	19	20 (100%)	20	1	22	1	21 (100%)	21	1
NDG20_t100.5	19	21 (100%)	21	1	21	1	21 (100%)	21	1
NDG20_t200.1	19	24 (25%)	25	81	34	1	27 (95%)	28	1
NDG20_t200.2	19	24 (10%)	26	612	33	1	27 (90%)	29	1
NDG20_t200.3	19	24 (10%)	25	641	33	1	28 (85%)	29	1
NDG20_t200.4	19	22 (5%)	25	925	35	1	29 (70%)	30	2
NDG20_t200.5	19	25 (60%)	26	176	35	1	28 (45%)	29	2
NDG20_t300.1	30	38 (55%)	39	453	48	1	42 (45%)	43	3
NDG20_t300.2	30	39 (65%)	40	624	48	1	43 (30%)	44	3
NDG20_t300.3	30	37 (10%)	39	848	53	1	41 (20%)	43	3
NDG20_t300.4	30	39 (70%)	40	375	51	1	43 (60%)	44	3
NDG20_t300.5	30	39 (80%)	40	556	50	1	43 (75%)	44	3
NDG40_t200.1	19	24 (15%)	26	734	37	6	30 (50%)	32	7
NDG40_t200.2	19	25 (60%)	26	278	36	6	29 (60%)	30	6
NDG40_t200.3	39	42 (65%)	43	319	51	1	46 (15%)	48	1
NDG40_t200.4	39	43 (5%)	44	483	55	1	48 (55%)	49	3
NDG40_t200.5	39	43 (5%)	44	188	53	1	47 (60%)	48	3
NDG40_t400	39	59 (5%)	61	1996	92	6	66 (40%)	68	16
NDG100_t500	39	69 (40%)	70	2816	107	93	75 (10%)	78	111

From Table 4, one observes that our TL-ILS obviously outperforms the two kinds of initial solution generation methods, showing the search potential of our TL-ILS algorithm. Although *HeuristicG* gives very similar results as TL-ILS in small instances, it fails to compete with the TL-ILS algorithm for larger instances, and the gap becomes larger with the increasing scale of the instances.

By comparison with the lower bounds, our TL-ILS algorithm gets pretty good results in some instances, while there are still some gaps for the remaining ones. According to our experiments, the problem becomes much harder with the problem size increasing. However, our TL-ILS obtains very good results for instances NDG40_t200.3, NDG40_t200.4 and NDG40_t200.5, which are very close to the lower bounds.

We find that the instances with a larger ratio of m to n' is much more difficult than other instances, where n' is the number of nodes involved in the traffic set T. One finds that our TL-ILS algorithm obtains relatively poor results for these instances. This is an interesting phenomenon that is worthy of further investigation.

6. Conclusion

In this paper, we have presented a TL-ILS algorithm for tackling the WDM network design problem with traffic grooming. The NDG problem is a typical optimization problem which is a quite challenging topic with high complexity, and it is widely considered in the optical communications industry as well as in academia.

By giving the mathematical formulation of the NDG problem, we propose a TL-ILS algorithm for solving it. The proposed TL-ILS algorithm follows a general framework which consists of three phases: initialization, intensification and diversification. A tree search based neighborhood evaluation method is presented, which gives a new perspective in designing neighborhoods for problems with a graph structure. An incremental neighborhood evaluation technique is also proposed to enhance the efficiency of move evaluations. In addition, we propose a new strengthened model by introducing several valid inequalities to obtain the lower bound of the NDG problem.

Two sets of totally 24 benchmark instances are generated according to real application scenarios from telecommunication company to evaluate our proposed algorithm. Computational results show the effectiveness and efficiency of our TL-ILS algorithm for solving the tested instances by comparison with the greedy and heuristic methods, as well as the lower bounds.

This study reminds us that it is essential to introduce meaningful neighborhood definition and evaluation techniques and highlight the problem specific knowledge in designing metaheuristic algorithms. Following this spirit, we hope to design even more robust and effective metaheuristic algorithms, such as the memetic algorithm [31], for solving the NDG problem as well as other network design problems.

Acknowledgments

We thank the anonymous referees for their insightful comments which help us to improve our paper significantly. The research was supported in part by National Natural Science Foundation of China under Grant numbers 61370183, 61100144 and the program for New Century Excellent Talents in University (NCET 2013). Special thanks are given to Huawei Technologies Co., Ltd. for defining this challenging and interesting problem and financially supporting this research work.

References

[1] B. Mukherjee, Optical Communication Networks, McGraw-Hill, 1997.
[2] Y. Wang, Q. Gu, On the complexity and algorithm of grooming regular traffic in WDM optical networks, J. Parallel Distrib. Comput. 68 (2008) 877–886.
[3] B. Chen, G. Rouskas, R. Dutta, On hierarchical traffic grooming in WDM networks, IEEE/ACM Trans. Netw. 16 (5) (2008) 1226–1238.
[4] M. Saleh, A. Kamal, Design and provisioning of WDM networks with many-to-many traffic grooming, IEEE/ACM Trans. Netw. 18 (6) (2010) 1869–1882.
[5] M. Saleh, A. Kamal, Approximation algorithms for many-to-many traffic grooming in optical WDM networks, IEEE/ACM Trans. Netw. 20 (5) (2012) 1527–1540.
[6] K. Zhu, B. Mukherjee, Traffic grooming in an optical WDM mesh network, IEEE J. Sel. Areas Commun. 20 (1) (2002) 122–133.
[7] J. Hu, B. Leida, Traffic grooming, routing, and wavelength assignment in optical WDM mesh networks, in: INFOCOM. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, 2004, pp. 495–501.
[8] S. Thiagarajan, A.K. Somani, Capacity fairness of WDM networks with grooming capabilities, in: Storage and Retrieval for Image and Video Databases, 2001.
[9] R. Srinivasan, A. Somani, A generalized framework for analyzing time-space switched optical networks, in: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings. IEEE, 2001, pp. 179–188.
[10] A. Rubio-Largo, M. Vega-Rodriguez, J. Gomez-Pulido, J. Sanchez-Perez, Multi-objective metaheuristics for traffic grooming in optical networks, IEEE Trans. Evol. Comput. 17 (4) (2013) 457–473.

[11] H. Wang, G.N. Rouskas, Hierarchical traffic grooming: a tutorial, Comput. Netw. 69 (2014) 147–156.

[12] T.F. Noronha, M.G. Resende, C.C. Ribeiro, A biased random-key genetic algorithm for routing and wavelength assignment, J. Glob. Optim. 50 (3) (2011) 503–518.

[13] A. Rubio-Largo, M. Vega-Rodriguez, Applying MOEAs to solve the static routing and wavelength assignment problem in optical WDM networks, Eng. Appl. Artif. Intell. 26 (5–6) (2013) 1602–1619.

[14] M.T. Chen, B.M. Lin, S.S. Tseng, Ant colony optimization for dynamic routing and wavelength assignment in WDM networks with sparse wavelength conversion, Eng. Appl. Artif. Intell. 24 (2) (2011) 295–305.

[15] B. Gendron, T.G. Crainic, A. Frangioni, Multicommodity Capacitated Network Design, Springer, 1999.

[16] B. Gendron, T.G. Crainic, Relaxations for multicommodity capacitated network design problems, Tech. Rep., Univ., CRT, 1994.

[17] T.G. Crainic, A. Frangioni, B. Gendron, Bundle-based relaxation methods for multicommodity capacitated fixed charge network design, Discret. Appl. Math. 112 (2001) 73–99.

[18] M. Yaghini, M. Momeni, M. Sarmadi, A simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design, Appl. Soft Comput. 12 (9) (2012) 2997–3003.

[19] I. Ghamlouche, T.G. Crainic, M. Gendreau, Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design, Oper. Res. 51 (4) (2003) 655–667.

[20] I. Ghamlouche, T.G. Crainic, M. Gendreau, Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design, Ann. Oper. Res. 131 (1–4) (2004) 109–133.

[21] Y. Agarwal, P. Venkateshan, Survivable network design with shared-protection routing, Eur. J. Oper. Res. 238 (3) (2014) 836–845.

[22] Y. Agarwal, k-Partition-based facets of the network design problem, Networks 47 (3) (2006) 123–139.

[23] F. Hamid, Y.K. Agarwal, A polyhedral approach for solving two facility network design problem, in: Network Optimization, Springer, 2011, pp. 92–97.

[24] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0 – survivable network design library, Networks 55 (2010) 276–286.

[25] J. Xu, S.Y. Chiu, F. Glover, Tabu search for dynamic routing communications network design, Telecommun. Syst. 8 (1997) 1–23.

[26] B. Alidaee, G.A. Kochenberger, A note on a simple dynamic programming approach to the single sink fixed charge transportation problem, Transp. Sci. 39 (1) (2005) 140–143.

[27] F. D'Andreagiovanni, J. Krolikowski, J. Pulaj, A fast hybrid primal heuristic for multiband robust capacitated network design with multiple time periods, Appl. Soft Comput. 26 (2015) 497–507.

[28] B. Kaushik, N. Kaur, A.K. Kohli, Achieving maximum reliability in fault tolerant network design for variable networks, Appl. Soft Comput. 13 (7) (2013) 3211–3224.

[29] H. Lourenco, O. Martin, T. Stützle, Iterated local search, in: Handbook of Metaheuristics, Vol. 57 of International Series in Operations Research and Management Science, Springer US, 2003, pp. 320–353.

[30] J.Y. Yen, Finding the k shortest loopless paths in a network, Manage. Sci. 17 (11) (1971) 712–716.

[31] U. Benlic, J.K. Hao, A multilevel memetic approach for improving graph k-partitions, IEEE Trans. Evol. Comput. 15 (5) (2011), 624–472.

[32] F. Palmieri, U. Fiore, S. Ricciardi, Selfish routing and wavelength assignment strategies with advance reservation in inter-domain optical networks, Comput. Commun. 35 (3) (2011) 366–379.

[33] F. Palmieri, U. Fiore, S. Ricciardi, A minimum cut interference-based integrated RWA algorithm for multi-constrained optical transport networks, J. Netw. Syst. Manage. 16 (4) (2008) 421–448.

[34] F. Palmieri, U. Fiore, S. Ricciardi, Constrained minimum lightpath affinity routing in multi-layer optical transport networks, J. High Speed Netw. 17 (4) (2010) 185–205.