Computer Networks 86 (2015) 27–39

Contents lists available at ScienceDirect

Computer Networks

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

GRASP for traffic grooming and routing with simple path constraints in WDM mesh networks

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

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

ARTICLE INFO

Article history:
Received 23 June 2014
Revised 17 March 2015
Accepted 24 April 2015
Available online 18 May 2015

Keywords:
Metaheuristics
Traffic grooming and routing
WDM mesh network
GRASP
Local search

ABSTRACT

This paper studies the traffic grooming and routing problem with simple path constraint (denoted as GR) in WDM mesh networks. To the best of our knowledge, the simple path constraint has not been studied in previous literature. However, this non-trivial constraint is essential in practice and it makes the traffic grooming and routing problem become much more challenging due to the introduction of quadratic constraints. By giving the mathematical formulation of the GR problem, we propose a greedy randomized adaptive search procedure (GRASP) for solving the GR problem, and introduce a mechanism to tackle the interaction between the grooming problem and the routing problem. To test the performance of the proposed GRASP algorithm, we apply it to tackle three sets of totally 38 instances generated according to real-world scenarios. Computational results show the efficacy of the GRASP algorithm in terms of both solution quality and search efficiency by comparison with public software LocalSolver and the lower bounds obtained by CPLEX.

1. Introduction

In order to expand the capacity of optical networks, wavelength division multiplexing (WDM) has been widely used nowadays. In a WDM network, optical fibers can carry multiple data streams, which are called traffic demands, by assigning each to a different wavelength. Each wavelength can be viewed as a channel, called lightpath, providing a connection between two nodes. Lightpaths are terminated at each end by transceivers which are optoelectronic equipments, enabling to convert optical signals into electronic ones for transmission.

Once a lightpath is established, it needs to be routed and assigned a wavelength. This is referred to as the routing and wavelength assignment (RWA) problem [1]. In order to maximize the usage of the lightpaths, telecommunication carriers adopt a technique that consists of efficiently grooming low speed traffic streams into high capacity channels. This technique is referred to as the RWA problem with traffic grooming (GRWA). Some similar problems in transportation and logistics, such as the capacitated network design problem (see [2–5]), the capacitated arc routing problem [6], the multi-objective multicast routing problem [7] and the bottleneck network flow problem [8], have been extensively studied in the last decades.

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, minimizing the number of transceivers is equivalent to minimizing the number of lightpaths.

Since the traffic grooming problem has proven to be NP-hard in ring networks [9] and the ring network is a special case of mesh networks, the traffic grooming problem in mesh network is also NP-hard. Moreover, the grooming problem in WDM networks is usually involved with other problems which makes it much more complicated [10]. In

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

http://dx.doi.org/10.1016/j.comnet.2015.04.013
1389-1286/© 2015 Elsevier B.V. All rights reserved.

order to solve these problems, effective optimization algorithms are essential for this challenging problem. Chen and Rouskas presented an effective and efficient hierarchical traffic grooming framework for WDM networks [11]. Saleh and Kamal addressed the problem of designing and provisioning of WDM networks to support many-to-many traffic grooming aiming at minimizing the overall network cost [12]. They also introduced two novel approximation algorithms for the many-to-many traffic grooming problem [13]. In [14], 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. Wang and Hou proposed a new multi-granularity grooming algorithm based on integrated grooming auxiliary graph, which not only performs traffic grooming and waveband switching together, but also solves the traffic-diversity problems effectively, especially for port savings [15]. Another direction in WDM networks is to optimize the energy consumption, see [16–18].

In spite of these numerous studies on the traffic grooming and routing problem, a non-trivial constraint, referred to as simple path constraint, was ignored in previous studies. The simple path constraint implies that the physical route of a traffic demand should visit each node for exactly once.

Although considering the simple path constraint will slightly increase the number of lightpaths and hence the transceiver costs, it is still highly demanded by the telecommunication companies in practice, because the simple path constraint is required by the end users in telecommunication companies.[1] As our industrial partner states, the simple path constraint is a hard constraint requited by the telecommunication companies. According to real world scenarios, if this constraint is not satisfied, it will be difficult to manage the traffic demands and may increase the disorders of the system, such that the increased management costs, such as personnels, trainings, etc., will offset the benefits of saving the transceiver costs.

In previous studies, the traffic grooming and lightpath routing problem can be separated into two independent subproblems, where the routing problem is usually considered with the wavelength assignment problem (see [19,20]). However, with this simple path constraint, the traffic grooming and lightpath routing problems are closely linked with each other. It is impossible to solve these two sub-problems separately, since solving one subproblem always affects the feasibility of the other. That is to say, the complexity of the problem with simple path constraint is not only reflected on the size of the problem, but also the inextricable nature of interdependent sub-problems. Many other real-world problems have such composite structures, e.g., optimizing the transportation of water tanks [21], PCB design in VLSI and the travelling thief problem [22].

Although exact methods are effective for small size problems, it becomes much more difficult when the size of the problem becomes large. In this case, the only promising method is to reply on effective metaheuristic algorithms, such as tabu search [23], simulated annealing [24], iterated local search [25] and memetic algorithm [26], etc. These metaheuristic algorithms have shown to be able to solve very large scale problems to near optimality in a reasonable time. However, in order to solve the real-world problems effectively and efficiently, metaheuristic algorithms should be adapted to the specific structure of the problem.

In this paper we study the traffic grooming and routing problem with the simple path constraint (GR), which aims to design a virtual topology network with the minimum number of lightpaths while satisfying a number of constraints. For this purpose, we adopt a greedy randomized adapted search procedure (GRASP) [27] which is a metaheuristic utilized in a wide range of applications (see [28,29] etc.). To test its performance, the proposed GRASP algorithm is applied to tackle 38 benchmark instances, showing its efficacy in terms of both solution quality and efficiency by comparing with the public software LocalSolver.

The rest of the paper is organized as follows. Section 2 describes the GR problem. Section 3 describes our GRASP algorithm in details. Computational results and comparisons with the reference algorithms are presented in Section 4, before concluding the paper in Section 5.

2. Grooming and routing problem with simple path constraints

2.1. Problem description

The GR problem can be described as follows: Given the physical mesh network and a set of traffic demands, the objective of the problem is to design a virtual topology which consists of a number of lightpaths on the physical network and to groom all the traffic demands on this virtual topology network and route all lightpaths on the physical network, such that the total number of the lightpaths (or the transceivers) is minimized. The physical mesh network is an undirected graph, and the traffic demands and the lightpaths are both non-directional.

Two kinds of constraints should be taken into account: One is the bandwidth capacity constraint for each lightpath; the other is the simple path constraint for the physical route of each traffic demand, which means that the physical route of a traffic demand should be a simple path. One should notice that multiple lightpaths can be established between the same pair of source and sink nodes.

Let us consider a small example: Fig. 1 shows an optimal solution for a 5 nodes network instance with 6 traffic demands and lightpath capacity of 3. The black lines represent the physical network while the color dash lines represent the lightpaths. The traffic demand ⟨A,C,1⟩ represents the traffic demand with bandwidth 1 from node A to node C. Here all the 6 traffic demands are satisfied by using 4 lightpaths, and the physical route for each traffic demand is a simple route.

The simple path constraint increases the difficulty of the GR problem especially on sparse networks. For example, Fig. 2 shows an optimal solution without considering the simple path constraint. The physical route for ⟨A,B,1⟩ is not a simple path. It passes through nodes A, B, C, and B. However, if the simple path constraint is considered, it is impossible to satisfy all the 6 traffic demands by using only 4 lightpaths. In

[1] This non-trivial constraint is regulated in a project with one of the largest telecommunication companies in China.

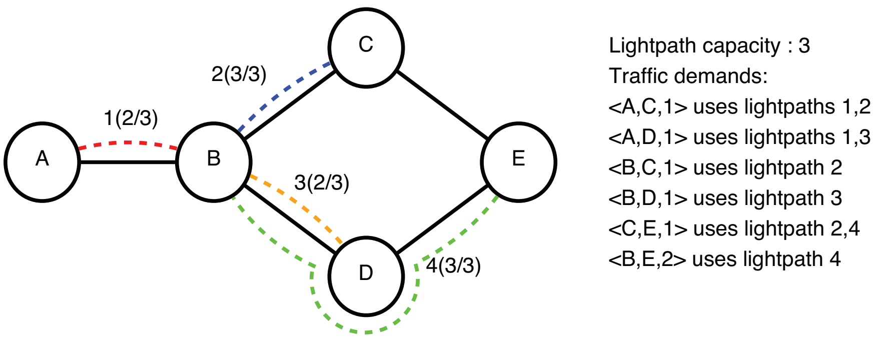


Fig. 1. An example for the GR problem.

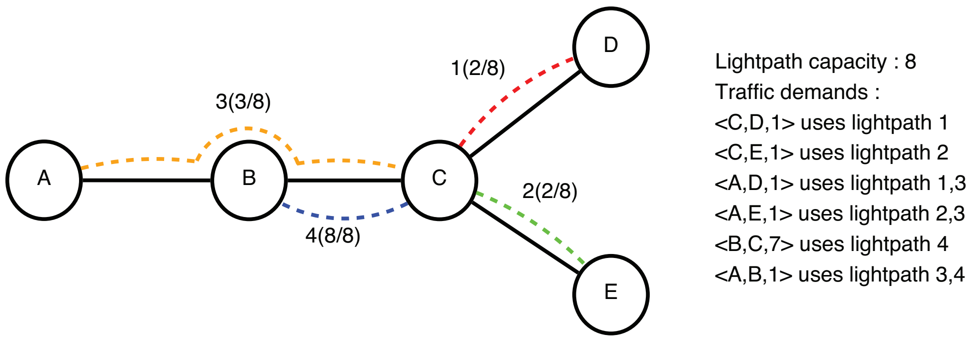


Fig. 2. A case without considering the simple path constraint.

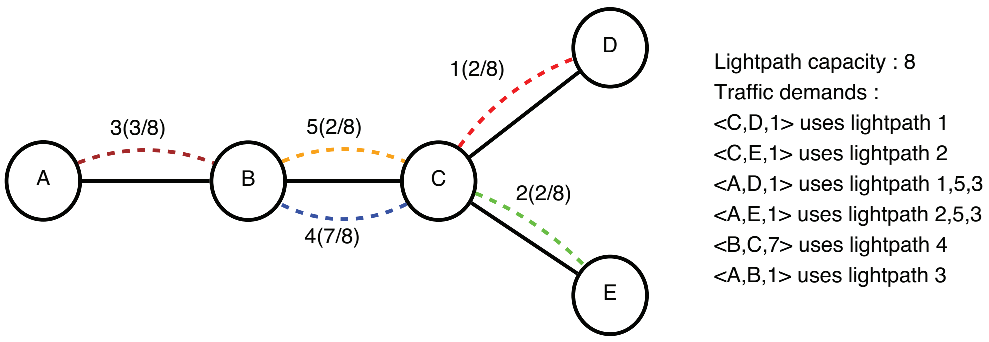


Fig. 3. A case considering the simple path constraint.

this situation, the optimal solution requires at least 5 lightpaths. Fig. 3 shows one of the optimal solutions.

2.2. Problem formulation

The GR problem consists of a set of m traffic demands $T = \{t_1, t_2, \ldots, t_m\}$ to be routed in the physical network $[N]_{n\times n}$ with n nodes, with the purpose to minimize the number of lightpaths which carry the traffic demands. Each element $t \in T$ is a three-tuple $t = (s_t, d_t, c_t)$ indicating that traffic demand t with bandwidth of c_t starts from node s_t and terminates at node d_t.

Since the worst case for a GR problem is to provide each traffic demand a lightpath, we assume that the number of lightpaths will not exceed the number of traffic demands.

A candidate solution is represented by four matrices: array $[x]$, the lightpath binary variables, where $x_i = 1$ if the lightpath index of i is in use and $x_i = 0$ vice versa; $m \times n \times n$ matrix $[z]$, the route decision variables for each lightpath, where $z_{lij} = 1$ if lightpath l passes through the physical route connecting nodes i and j and $z_{lij} = 0$ vice versa; $m \times m$ matrices $[y]$ and $[y']$, the lightpaths binary variables for each traffic demand. Since the lightpaths are non-directional but their routes are directional, these two matrices are used to represent the lightpath options for traffic demands. We define $y_{tl} = 1$ if traffic demand t traverses lightpath l (from s_l to d_l) and $y_{tl} = 0$ vice versa; $y'_{tl} = 1$ if traffic demand t traverses lightpath l reversely (from d_l to s_l) and $y'_{tl} = 0$ vice versa.

There are other two decision variables in our formulation. They are $m \times n \times n$ matrix $[s]$, where $s^l_{ij} = 1$ if lightpath l's source is node i and its sink is node j and $s^l_{ij} = 0$ vice versa; $m \times n \times n$ matrix ψ, where $\psi^t_{ij} = 1$ if traffic demand t traverses the lightpath connecting nodes i and j.

Table 1
Notations used for the mathematical formulation of the GR problem.

Symbols	Description
n	The total number of nodes in the physical network
m	The total number of traffic demands
B	The capacity for each lightpath
T	Set of traffic demands, $t \in T, t = (s_t, d_t, c_t), \|T\| = m$
$[N]_{n\times n}$	Adjacency matrix for the physical network
b^t_j	Indicate the source and sink nodes of a traffic demand $b^t_j = \begin{cases} 1, & \text{for } j = d_t \\ -1, & \text{for } j = s_t \\ 0, & \text{otherwise} \end{cases}$
s^l_{ij}	End points decision variable for lightpath l
x_l	Decision variable for the lightpath
y_{tl} and y'_{tl}	Lightpath decision variable for the traffic demand
ψ_{tij}	Decision variable for the route of the traffic demand
z_{lij}	Decision variable for the route of the lightpath

The symbol and variable definitions are presented in Table 1. Given these notations, we can describe the GR problem in a formal way as follows:

Minimize: $\sum_{i=1}^{m} x_i$

Subject to:

$$y_{tl} + y'_{tl} \le x_l \quad t, l = 1, 2, \dots m \tag{1}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} s^l_{ij} = 1 \quad l = 1, 2, \dots m \tag{2}$$

$$\psi_{tij} \le \sum_{l=1}^{m} s^l_{ij} y_{tl} + \sum_{l=1}^{m} s^l_{ji} y'_{tl} \quad t = 1, 2, \dots, m \text{ and } i, j = 1, 2, \dots, n \tag{3}$$

$$\sum_{i=1}^{n} \psi_{tij} - \sum_{i=1}^{n} \psi_{tji} = b^t_j \quad t = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n \tag{4}$$

$$z_{lij} \le N_{ij} \quad l = 1, 2, \dots, m \text{ and } i, j = 1, 2, \dots, n \tag{5}$$

$$\sum_{i=1}^{n} z_{lij} - \sum_{i=1}^{n} z_{lji} = \sum_{i=1}^{n} s^l_{ij} - \sum_{i=1}^{n} s^l_{ji} \quad l = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n \tag{6}$$

$$\sum_{l=1}^{m} (y_{tl} + y'_{tl}) \left(\sum_{i=1}^{n} z_{lij} + \sum_{i=1}^{n} z_{lji} \right) \le 2 \quad t = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n \tag{7}$$

$$\sum_{t=1}^{m} c_t (y_{tl} + y'_{tl}) \le B x_l \quad l = 1, 2, \dots, m \tag{8}$$

$$x_l, y_{tl}, y'_{tl}, z_{lij}, \psi_{tij}, s^l_i \in \{0, 1\} \quad t, l = 1, 2, \dots, m \text{ and } i, j = 1, 2, \dots, n \tag{9}$$

The objective function $\sum_{i=1}^{m} x_i$ calculates the total number of lightpaths used in the network. The constraints are explained as follows: (1) guarantees that lightpath l used by traffic demand t is an available one, and one lightpath cannot be used by a certain traffic demand both forwardly and reversely; (2) ensures that each lightpath has proper source and sink nodes; (3) and (4) require that the route of each traffic demand on the virtual topology is feasible; (5) and (6) ensure that the physical path for each lightpath is feasible; (7) is the simple path constraint for each traffic demand; (8) ensures that each lightpath can only carry traffic demand not more than its capacity.

It can be seen from the formulation that if we ignore the routing problem, the problem would be similar to the non-bifurcated network design problem (NND) [30]. The NND problem is a variant of the multi-commodity network design problem (MCND) [2], but is more difficult to solve than the MCND problem due to the introduction of integer variables.

There are two main differences between the GR problem and the NND problem. First, the objective of the GR problem is to construct a virtual topology based on a physical network, while the NND problem focuses on which edge should be constructed for the given network. Second, the solution topology of the GR problem can be a multi-graph while that of NND is a simple graph, which means that there can be multiple edges (lightpaths) between the same pair of source and sink nodes on the solution topology of the GR problem, and each lightpath has its own identical physical route. This greatly increases the flexibility of the solution of the GR problem. Moreover, the simple path constraint greatly decreases the number of feasible solutions for the GR problem. All these aspects indicate that GR problem is a challenging problem compared with previous similar problems.

There may exist other formulations for the GR problem, which may be easier to be solved by integer programming method. However, it is out of the scope of this paper.

3. GRASP algorithm for GR

In order to better describe our proposed GRASP algorithm for solving the GR problem, we first define some symbols as shown in Table 2.

GRASP is a well-known metaheuristic methodology used in a wide range of applications. Our GRASP algorithm follows a general framework which is an iterative process where each iteration consists of two phases: A construction phase and a local search phase. The general framework of our GRASP algorithm is described in Algorithm 1.

3.1. Construction

The construction phase of our GRASP algorithm generates a feasible solution in an iterative way. At each construction iteration, one traffic demand is chosen and assigned a lightpath route which traverses nodes with no loops, and a new lightpath is introduced for the traffic demand if the assignment fails, as shown in Algorithm 2.

Before the construction phase, the traffic demands are sorted in a descending order by the number of hops on the virtual topology, which means that the traffic demand using more lightpaths will have higher priority to be reassigned to the network. At the first iteration of our GRASP algorithm, the traffic demands are assigned in a random order, since there is no original assignment for each traffic demand.

Usually, too long lightpath is not good for the following operations of the algorithm. Thus, we want the lightpaths to be as short as possible. So, when one assignment fails, instead

Table 2
Symbols used in the GRASP algorithm.

Symbols	Description
O_l	The overload value of the lightpath l.
C_t	The number of duplicating nodes in the physical route for traffic demand t.
f_g	The objective for procedure *Grooming*.
f_r	The objective for procedure *Routing*.
Δf_g	The incremental objective value of f_g.
Δf_r	The incremental objective value of f_r.
L_u	The set consisting of the indexes of lightpaths.
l	Index of a lightpath.
c^l	The unused bandwidth of lightpath l.
t	Index of a traffic demand.
S	A candidate solution, consists of $[x]$, $[y]$, $[z]$.
$\|S\|$	The number of lightpaths that S uses.
S_c	The current solution.
S_b	The best solution found so far.
S'_b	The best solution found so far in procedure LightpathMin.
$N_{LM}(S)$	The neighborhood of S in *Lightpaths-Minimization*.
$N_g(S)$	The neighborhood of S in *Grooming*.
$N_r(S)$	The neighborhood of S in *Routing*.
$\mathbf{0}$	Zero matrix.

Algorithm 1 The main framework.

```
1: procedure MAIN($[N]$, $T$)
2:     $|S_b| \leftarrow \infty$
3:     $S_c \leftarrow \mathbf{0}$
4:     repeat
5:         $S_c \leftarrow$ CONSTRUCTION($S_c$, $T$)
6:         $S_c \leftarrow$ LIGHTPATHMIN($S_c$)
7:         if $|S_c| \leq |S_b|$ then
8:             $|S_b| \leftarrow |S_c|$
9:         else
10:            $|S_c| \leftarrow |S_b|$
11:        end if
12:        if Meet the Termination condition then
13:            EXPORTSOLUTION($S_c$)
14:            EXIT
15:        end if
16:    until Forever
17: end procedure
```

Algorithm 2 Construction.

```
1: procedure CONSTRUCTION($S_c$)
2:     SORT(T)
3:     Remove all the traffic demands from the virtual topology
4:     for all $t \in T$ do
5:         if $\exists P$ connecting $s_t$ and $d_t$ AND $c^l \geq c_t \forall l \in P$ AND $P$ is a simple path then
6:             ASSIGN($S_c$, $t$, $P$)
7:         else
8:             $P \leftarrow \Phi$
9:             $sublen \leftarrow distance(s_t, d_t)$
10:            for all nodes $n$ on virtual topology do
11:                if $\exists P'$ connecting $s_t$ and $n$ AND $c^l \geq c_t \forall l \in P'$ then
12:                    if $distance(n, dt) < sublen$ then
13:                        $P \leftarrow P' \cup$ {A new $l$ connecting $n$ and $d_t$}
14:                        $sublen \leftarrow distance(n, dt)$
15:                    end if
16:                end if
17:            end for
18:            if $P \neq \Phi$ then
19:                ASSIGN($S_c$, $t$, $P$)
20:            else
21:                $P \leftarrow$ {A new $l$ connecting $s_t$ and $d_t$}
22:                ASSIGN($S_c$, $t$, $P$)
23:            end if
24:        end if
25:    end for
26: end procedure
```

of introducing a specific lightpath for the traffic demand right away, we detect whether the unused bandwidth of the existing virtual topology can be used to add a shorter lightpath to satisfy this traffic demand.

It is noteworthy that in the above procedure, a tree search based algorithm is used to find the feasible path for one traffic demand, while the same technique is used in Section 3.2. This technique is one of the main factors affecting the efficiency of our algorithm.

3.2. Lightpath-Minimization

The *Lightpath-Minimization* phase is aimed at decreasing the number of lightpaths and is composed of three local search algorithms [25] in a hierarchical structure. The high level local search algorithm is a virtual topology transformation algorithm, while two low level local search algorithms (Section 3.2.1 and 3.2.2) solve the grooming and routing problem, respectively. The pseudo-code of the *Lightpath-Minimization* phase is shown in Algorithm 3.

It is widely believed that one of the most important features of a local search algorithm is the definition of its neighborhood. In a local search procedure, applying a move mv to a solution S leads to a new solution denoted by $S \oplus mv$.

Algorithm 3 Lightpath-minimization.

```
1:  procedure LIGHTPATHMIN($S_c$)
2:      $L_u \leftarrow$ INDEXOFLIGHTPATHS([$x$])
3:      $S'_b \leftarrow S_c$
4:      repeat
5:          $l \leftarrow$ CHOOSELIGHTPATH($L_u$)
6:          $L_u \leftarrow L_u \setminus \{l\}$
7:          $T_u \leftarrow$ Traffic demands passing through $l$
8:          $S_c \leftarrow$ REMOVELIGHTPATH($S_c, l$)
9:          $flag \leftarrow$ GROOMING($S_c, T_u$)            ▷ Section 3.2.1
10:         if $flag$ = TRUE then
11:             $flag \leftarrow$ ROUTING($S_c$)              ▷ Section 3.2.2
12:         end if
13:         if $flag$ = TRUE then
14:             $S'_b \leftarrow S_c$
15:         else
16:             $S_c \leftarrow S'_b$
17:         end if
18:     until $L_u = \phi$
19:     return $S'_b$
20: end procedure
```

Let $M(S)$ be the set of all possible moves which can be applied to S, then the neighborhood N of S is defined by: $N(S) = \{S \oplus mv | mv \in M(S)\}$.

In our GRASP algorithm, there are three neighborhood structures respectively for the three local search algorithms. The neighborhood structure $N_{\text{LM}}(S)$ used in *Lightpaths-Minimization* 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 $N_{\text{LM}}(S)$ is bounded by $O(\mu)$, where μ is the number of lightpaths.

More specifically, let L_u denote the indices of all the lightpaths where all the lightpaths are sorted by favoring long distance between the source and sink nodes. At each iteration, we choose one lightpath from L_u and tentatively remove the lightpath from the current solution. Then, we try to reassign those traffic demands carried by the removed lightpath on the virtual topology by calling the subroutines *Grooming* and *Routing*. Subroutine *Grooming* assigns these traffic demands to the virtual topology, ensuring that the capacity constraint is satisfied (constraint (8)). If subroutine *Grooming* returns TRUE, subroutine *Routing* is run to ensure that the physical route of each traffic demand is a simple path (constraint (7)).

In this high level local search, we use a first-improvement based local search procedure, where we accept a candidate solution in the current neighborhood if the deletion of one lightpath l can still satisfy all the constraints. 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.

3.2.1. Grooming

As a subroutine of *Lightpath-Minimization*, the *Grooming* procedure can be considered as a low level local search algorithm in our GRASP algorithm. Once the virtual topology changes by removing one lightpath, the *Grooming* procedure is called to verify if the remained virtual topology can still satisfy the capacity constraint. The main purpose of *Grooming* is to assign traffic demands to lightpaths such that all the lightpaths are not overloaded. There are two steps in the Grooming procedure: (1) randomly choose a lightpath route for the traffic demands passing through the deleted lightpath; (2) employ a local search to eliminate the total overload. There are cases where no route is available for some traffic demands if one lightpath has been deleted. The *Grooming* procedure returns FALSE if there are traffic demands that cannot find a feasible lightpath route even without considering the simple path constraint.

Since the main purpose of *Grooming* is to eliminate the overload on the virtual topology, one straightforward way is to define the objective function as the violation of the capacity constraints for all the lightpaths:

$$f_g = \sum_{l=1}^{m} O_l$$

where O_l represents the overload value of the lightpath l.

$$O_l = x_l \cdot \max\left(\sum_{t=1}^{m} c_t(y_{tl} + y'_{tl}) - B, 0\right)$$

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

However, in our experiments, we observe that this definition fails to describe the essence of the GR problem and it makes the whole algorithm inefficient. Another candidate definition is:

$$f_g = \lambda \cdot \sum_{l=1}^{m} O_l + \sum_{t=1}^{m} C_t$$

where λ is a large positive constant that is experimentally set to 1000, and C_t represents the number of duplicated nodes in the physical route for traffic demand t, which is calculated as:

$$C_t = \sum_{j=1}^{n} \max\left(\sum_{l=1}^{m}(y_{tl} + y'_{tl})\left(\sum_{i=1}^{n} z_{lij} + \sum_{i=1}^{n} z_{lji}\right) - 2, 0\right)$$

This new objective function (which we call the advanced objective function) makes it easier to get a feasible solution for the *Grooming* procedure. We conducted a comparison and analysis between these two objective functions in Section 4.

The neighborhood used in the *Grooming* procedure, denoted as $N_g(S)$, is composed of all feasible moves of *T-routeChange*, where a *T-routeChange* move changes the route of one traffic demand to another candidate route. The size of neighborhood $N_g(S)$ is bounded by $O(m \cdot k)$ where k is the maximum number of candidate routes for one traffic demand.

At each iteration of the Grooming procedure, we choose the candidate solution in $N_g(S)$ with the minimum Δf_g value. The procedure stops when $f_g = 0$ or the maximum times of perturbation is reached. A branch and bound algorithm is used to find the best route to change for each traffic demand. The algorithm is shown in Algorithm 4. f_g^t represents the objective value contributed by traffic demand t in the current solution, and $f_g^{t,P}$ represents the objective value that t will cause if it switches its route to P. P is a sequence of nodes on the virtual topology. Note that this part is one of the main

Algorithm 4 FindMoveForTraffic.

```
1: procedure FindMoveForTraffic($t$, $virtualTopology$)
2:     $Queue \leftarrow \Phi$
3:     $bestPath \leftarrow \Phi$
4:     $cutValue \leftarrow f_g^t$
5:     $Queue.push$(source node of $l$)
6:     $Tree.root \leftarrow$source node of $l$
7:     while $Queue \neq \Phi$ do
8:         $u \leftarrow Queue.pop()$
9:         $P \leftarrow$ {The path from $Tree.root$ to $u$ on $Tree$}
10:        for all $v \in$ {nodes that adjacent to u in virtualTopology} do
11:            if $v \notin P$ AND $|P| < 6$ AND $f_g^{t,P} < cutValue$ then
12:                $Queue.push(v)$
13:                $Tree.node(u).attach(v)$
14:            end if
15:            if $v =$ sink node of $l$ AND $f_g^{t,P\cup v} < f_g^{t,bestPath}$ then
16:                $bestPath \leftarrow \{P, v\}$
17:                $cutValue \leftarrow f_g^{t,\{P,v\}}$
18:            end if
19:        end for
20:    end while
21:    return $bestPath$
22: end procedure
```

factors that affect the efficiency of our algorithm. A perturbation procedure will be triggered if the search finds a local optimum and $f_g \neq 0$. The perturbation procedure randomly changes the route of the traffic demand which has contribution to the objective. The procedure returns TRUE if $f_g = 0$ or the best f_g found so far is less than λ. Otherwise, it returns FALSE.

The procedure stops when the local optimum is less than λ. In order to prevent the algorithm from being trapped in local optimum, the procedure also stops when the perturbation time limit ρ is reached, where ρ is defined as:

$$\rho = \left\lfloor \max\left(\frac{3}{1-10\lambda}(f_g - 10\lambda), 0\right) \right\rfloor$$

This variable perturbation limit can make our algorithm more flexible with different situations. The procedure will execute more perturbations if f_g is close to λ when there is chance to get a feasible solution for grooming, and stops when it detects that the problem is less likely to be solved (a very large f_g). Based on this equation, the procedure will execute the perturbation operator for at most 3 times, and stops when the local optimum value is larger than 10λ.

Usually, the route length for a traffic demand should not be too long, in order to save memory and computing time. The hop limit for a certain traffic demand can be defined as 1.5 times of the shortest distance between the source and sink nodes on the virtual topology.

3.2.2. Routing

Once the *Grooming* procedure returns TRUE, which means that all the capacity constraints are satisfied, the *Routing* procedure is run to verify if the routing constraints can be satisfied too. The main purpose of *Routing* is to make the physical route of each traffic demand be a simple path while keeping the capacity constraints satisfied.

Specifically, the objective function f_r is defined as:

$$f_r = \sum_{t=1}^{m} C_t$$

where C_t is the same as that in Section 3.2.1.

One observes that $f_r = 0$ corresponds to a solution for which the physical route of each traffic demand is a simple path. Similar to *Grooming*, the neighborhood structure for *Routing* (denoted as $N_r(S)$) is composed of all feasible moves of *L-routeChange*, where a *L-routeChange* move changes the route of one lightpath to another candidate route of this lightpath. The size of neighborhood $N_g(S)$ is bounded by $O(\mu \cdot \kappa)$, where μ is the number of lightpaths, and κ is the maximum number of candidate routes for a lightpath.

At each iteration of the local search in procedure *Routing*, we choose the candidate solution in $N_r(S)$ with the minimum Δf_r value. A similar branch and bound algorithm as described Algorithm 4 is used to evaluate the routes for each lightpath. Perturbation for this procedure will be triggered if the procedure finds a local optimum and $f_r \neq 0$. The perturbation will randomly change the route for the lightpath which causes conflicts. The procedure stops at $f_r = 0$ and returns TRUE. Otherwise, it stops when the perturbation limit ρ_r is reached and returns FALSE.

As in Section 3.2.1, ρ_r is defined as:

$$\rho_r = \left\lfloor \max\left(\frac{6}{1-100}(f_r - 100), 0\right) \right\rfloor$$

Therefore, the procedure will implement the perturbation for at most 6 times, and stops when the local optimum value is larger than 100.

3.3. An improved construction phase

In spite of the simple form of the *Construction* procedure, its solution quality is not satisfied. We can improve the *Construction* procedure by taking advantage of the procedures *Grooming* and *Routing*. Specifically, when one fails to assign a traffic demand to the virtual network directly, we call the procedures *Grooming* (Section 3.2.1) and *Routing* (Section 3.2.2) to readjust the traffic demands into the current network. If this is impossible, we add a new lightpath to carry the traffic demand. In this way, we can generate a virtual topology with much less number of lightpaths.

4. Experimental results

In this section, we report intensive experimental results of the proposed GRASP algorithm on three sets of totally 38 instances and compare our results with the results obtained by LocalSolver and the lower bounds.[2]

4.1. Test instances

Three sets of test problem instances are considered in our experiments, in total constituting 38 instances. The first

[2] The tested instances and our results will be made public upon the publication of this paper for future comparisons.

set of benchmarks is composed of 8 small instances in ring and mesh structure of size $n = 8$–10 and $m = 15$–70. The second set of benchmarks consists of 24 large problem instances with size ranging from $n = 20$–100 and $m = 200$–500, which are divided into 4 groups. Note that these instances are generated according to the characteristics of the real-world scenarios in our industrial partner, one of the largest Chinese telecommunication companies. The third set includes instances of large amount of traffic demands from real world networks that were used in the literature [14,31].

4.2. Experimental protocol

Our GRASP algorithm is programmed in C++ and compiled using MinGW on a PC running Windows 7 with 3.1 GHz CPU and 4.0 Gb RAM. In order to analyze the important features of the GRASP algorithm, three algorithms are used for the second set of benchmarks:

GRASP: The full featured GRASP.
GRASP$_{pG}$: GRASP with *Grooming* using the straightforward objective function.
Local search: Local search algorithm with *Grooming* using the advanced objective function.

Note that the results for the first set of benchmarks in Table 4 are obtained by the full featured GRASP.

4.3. The lower bound

In order to assess the performance of our GRASP algorithm, we obtain the lower bound by solving a relaxed model or using an estimated lower bound.

The original formulation is not suitable to be solved using the integer programming method because of the two quadratic constraints, while relaxing the quadratic constraint (3) will make the formulation meaningless. So we use another formulation to describe the *Grooming* problem without considering the route for each lightpath. The new formulation can be described as follows:

Minimize: $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \chi_{ij}$

Subject to:

$$\chi_{ij} = \chi_{ji} \quad i, j = 1, 2, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} \eta_{tij} - \sum_{i=1}^{n} \eta_{tji} = b_j^t \quad t = 1, 2, \ldots m \text{ and } j = 1, 2, \ldots, n \tag{11}$$

$$\sum_{t=1}^{m} c_t(\eta_{tij} + \eta_{tji}) \le B\chi_{ij} \quad i, j = 1, 2, \ldots, n \tag{12}$$

$$\chi_{ij} \in N \quad i, j = 1, 2, \ldots, n \tag{13}$$

$$\eta_{tij} \in \{0, 1\} \quad t = 1, 2, \ldots m \text{ and } i, j = 1, 2, \ldots, n \tag{14}$$

Here χ_{ij} represents the number of lightpaths established between nodes i and j; μ_{tij} represents the route for each traffic demand, where $\eta_{tij} = 1$ if traffic demand t passes through the lightpath which connects nodes i and j and $\eta_{tij} = 0$ vice versa.

Since this problem can be considered as a simplified version of the GR problem, its optimal objective value can be considered as a lower bound for the GR problem. In the following sections the lower bounds are calculated by CPLEX 12.4 using this formulation. However, for the large instances in Sections 4.5 and 4.6, CPLEX fails to give the optimal solutions for the simplified model due to the large size of the problem. For these problems, we obtain the lower bounds by relaxing the integer restriction for each variable in the model or estimating a value using the following method.

Assume n' nodes are involved in the traffic demand set T, then these n' nodes should be in the network of the final solution. As we know, the minimum graph that contains n' nodes should have at least $n' - 1$ edges. So, $n' - 1$ can be considered as a lower bound for the problem. This lower bound matches well for those instances which have few number of traffic demands.

Both lower bound methods are used for each instance, and better result is considered to be the lower bound of the instance in the following sections.

Table 3
Properties of the small-scale instances.

Instance	Nodes	Edges	Traffic demands	B
ring_1	8	8	15	8
ring_2	10	10	20	8
mesh_1	10	11	20	8
mesh_2	10	15	40	8
mesh_3	10	15	50	8
mesh_4	10	12	60	8
mesh_5	10	12	70	8
mesh_6	10	13	70	8

4.4. Results for the small-scale instances

The physical networks for the first two instances are in ring structure, and the last six ones are in mesh structure. The mesh physical networks are constructed by choosing pairs of nodes randomly. The traffic demand set for each instance is also generated randomly. The main features of these instances are shown in Table 3. The lightpath capacity is set to be 8, and the bandwidth for each traffic demand is 1.

In this section, the performance of the GRASP is compared with the public software LocalSolver 3.1 [32]. LocalSolver is an effective general purpose optimization software, which has shown its effectiveness in the ROADEF/EURO Challenge 2012[3] and many other problems [32].

Due to the introduction of the quadratic constraints, the original model of the GR problem can hardly be solved by CPLEX. For the smallest instance ring_1, CPLEX runs for hours without giving a feasible solution and is crashed due to the lack of memory. So we do not compare CPLEX with our algorithm here.

The best objective values found by GRASP and LocalSolver for the 8 instances are given in Table 4. Our GRASP algorithm and LocalSolver are run for 10 times for each instance. The gap is calculated as: (UB − LB)/LB. The best objective value UB is provided by GRASP, LocalSolver 5.1 while the lower bound LB is calculated as described in Section 4.3. The time

[3] The ROADEF/EURO challenge 2012 dedicated to the machine reassignment problem in collaboration with Google. http://challenge.roadef.org/2012/en/index.php.

Table 4
Result for the small-scale instances.

		GRASP			LocalSolver		
Instance	LB	Best	Gap (%)	Time (s)	Best	Gap (%)	Time (s)
ring_1	7	7	0	1	9	28.6	1
ring_2	9	9	0	1	11	22.2	1
mesh_1	9	9	0	6	11	22.2	7
mesh_2	10	11	10	7	14	40	144
mesh_3	11	12	9.1	6	16	45.5	258
mesh_4	13	17	30.8	9	19	46.1	144
mesh_5	14	18	28.6	8	21	50	243
mesh_6	14	17	21.4	9	21	50	173

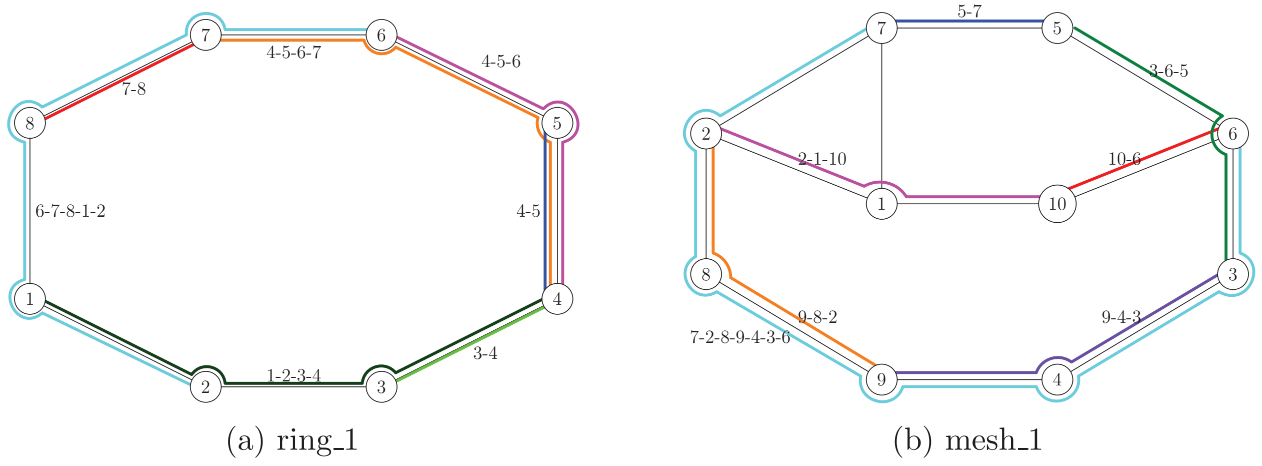


Fig. 4. The solutions for ring_1 and mesh_1.

limits of each instance for GRASP and LocalSolver are set to be 600 s.

One observes that our GRASP outperforms LocalSolver in all the instances. Particularly, for three out of eight instances, our algorithm is able to obtain the optimal solution by hitting the lower bound. For the remaining instances, our algorithm obtains the objective values very close to the lower bound, while LocalSolver gives the results with larger gaps. This experiment shows the efficacy of our GRASP algorithm. Moreover, for all the instances, LocalSolver takes several minutes to get its best results, while our GRASP algorithm obtains the optimal solutions or near-optimal solutions within at most 10 s.

The solutions for instances ring_1 and mesh_1 are illustrated in Fig. 4. In Fig. 4, the black lines represent the physical network, while the color curves represent the lightpaths.

4.5. Results for the large-scale instances

In this section, four groups of totally 24 instances with different size are used, the smallest with 20 nodes, 25 edges and 200 traffic demands and the largest with 100 nodes, 300 edges and 500 traffic demands. Each of groups R1, R2 and R3 has 5 individual instances with physical networks in different densities. Group G contains 9 instances of which the physical networks are flat graphs. In each instance, half of the traffic demands have a bandwidth of 1, and the remaining traffic demands have a bandwidth of 2. For all the instances, the lightpath capacity is set to be 32. Table 5 shows the main features of these 24 instances.

Table 5
Characteristics of the large-scale instances.

Group	Instance	Nodes	Edges	Traffic demands	B
R1	R20_200_1.1	20	42	200	32
	R20_200_1.2	20	31	200	32
	R20_200_1.3	20	42	200	32
	R20_200_1.4	20	36	200	32
	R20_200_1.5	20	37	200	32
R2	R20_200_1.1	20	51	200	32
	R20_200_2.2	20	49	200	32
	R20_200_2.3	20	57	200	32
	R20_200_2.4	20	51	200	32
	R20_200_2.5	20	60	200	32
R3	R20_200_1.1	20	79	200	32
	R20_200_3.2	20	74	200	32
	R20_200_3.3	20	75	200	32
	R20_200_3.4	20	67	200	32
	R20_200_3.5	20	71	200	32
G	G20_200_1	20	37	200	32
	G20_200_2	20	39	200	32
	G20_200_3	20	24	200	32
	G20_200_4	20	39	200	32
	G20_200_5	20	35	200	32
	G40_200_1	40	80	200	32
	G40_200_2	40	105	200	32
	G40_400	40	90	400	32
	G100_500	100	300	500	32

We run three versions of our algorithms for this set of instances and the time limit is set to be 2 h. The detailed computational results are shown in Table 6. Column LB gives the lower bounds for each instance which are obtained as

Table 6
Results for the large-scale instances.

Instance	LB	GRASP Best	GRASP Avg	GRASP$_{pG}$ Best	GRASP$_{pG}$ Avg	Local search Best	Local search Avg
R20_200_1.1	19	**27**	27.2	34	35	30	30.6
R20_200_1.2	19	**26**	26.8	38	38.3	35	35.3
R20_200_1.3	19	**25**	25.7	31	33.5	31	31.6
R20_200_1.4	19	**27**	27.4	34	35	33	33.3
R20_200_1.5	19	**28**	28.3	36	36.3	34	34.5
R20_200_2.1	19	**24**	24.3	33	33.1	27	28
R20_200_2.2	19	**24**	24.3	32	32.2	30	30.3
R20_200_2.3	19	**26**	26.2	34	34.9	31	31.4
R20_200_2.4	19	**25**	25.5	33	33.2	29	30.8
R20_200_2.5	19	**22**	22.4	26	31.8	30	30.3
R20_200_3.1	19	**22**	22.7	23	23.2	28	28.5
R20_200_3.2	19	**23**	23.2	25	25.1	27	27.6
R20_200_3.3	19	**23**	23.7	25	25.3	26	26.2
R20_200_3.4	19	**23**	23.2	23	23.9	27	27.7
R20_200_3.5	19	**23**	23.6	24	24.1	25	25.7
G20_200_1	19	**27**	27.6	37	38.1	34	34.6
G20_200_2	19	**26**	27.4	38	38	34	34
G20_200_3	19	**27**	28	41	41.6	38	38.7
G20_200_4	19	**26**	26.9	36	37.2	34	34.3
G20_200_5	19	**26**	27.2	35	35.9	34	34
G40_200_1	19	**26**	26.6	33	35.9	33	33.2
G40_200_2	19	**23**	23.7	26	29.1	30	30.4
G40_400	39	**67**	67.6	84	85.1	74	74.5
G100_500	39	**67**	68.5	77	79.9	77	77

Table 7
Comparison for the stability of the reference algorithms.

Group	GRASP	GRASP$_{pG}$	Local search
R1	0.632658	0.75889	0.7852146
R2	0.5222467	0.949104	1.4015166
R3	0.6012434	0.506313	0.6223253
G	0.7756371	1.719572	0.4185623

described in Section 4.3. Column Best reports the best result obtained by our algorithm over 10 independent runs (The values in bold show better output than the ones from other algorithms). However, LocalSolver fails to obtain a competitive solution for any instance of this group. Therefore, we do not report its results in this section. Table 7 presents the average standard deviation of each algorithm for each group based on this experiment and Fig. 5 illustrates the performance evolving with the running time for instances R20_200_1.3, R20_200_2.2, R20_200_3.1 and G40_200_2, which demonstrate the significance of our GRASP algorithm in terms of both solution quality and search efficiency.

Table 9
Results of the COST239 and NSF instances.

Network	Traffic set	LB	GRASP	GRASP$_{pG}$
COST239	TS1	37	41 (10.8%)	43 (16.2%)
	TS2	62	71 (14.5%)	71 (14.5%)
	TS3	115	127 (10.4%)	127 (10.4%)
NSF	TS1	42	48 (14.3%)	52 (23.8%)
	TS2	69	81 (17.4%)	82 (18.8%)
	TS3	122	138 (13.1%)	140 (14.8%)

4.6. Results for the COST239 and NSF networks

Two optical topologies used in the literature [14,31] are tested in this section. These two networks are called the European optical network (COST239) and the National Science Foundation network (NSF). Each network is combined with different traffic demands to form the instances of this section. The characteristics of the instances are given in Table 8. Columns 1, 3 and 12 represent the number of traffic demands with bandwidth of 1, 3 and 12, respectively. The best results for these instances given by GRASP and GRASP$_{pG}$ are presented in Table 9. The percentage in the brackets are the gap value for each instance and the time limit for each instance is also 2 h on our computer. One finds that GRASP$_{pG}$ obtains the same results with GRASP on network COST239 with traffic demand sets TS2 and TS3, while it gets worse results compared with GRASP on other instances.

Based on the above experiments, one observes that our GRASP algorithm is competitive not only for small scale instances but also for large scale instances in terms of both solution quality and efficiency.

4.7. Discussion on the grooming procedure

In order to evaluate whether the refined objective function is essential for the *Grooming* procedure, we now discuss this important feature of our proposed GRASP algorithm. We focus on the search capability of GRASP, compared with GRASP$_{pG}$. The main difference between the two procedures is the objective function in *Grooming* procedure. In GRASP$_{pG}$, we only consider the overload for each lightpath, while in GRASP, the nodes duplicates as well as the lightpath overload are both considered.

From Tables 6 and 9 we can easily find that GRASP has obvious advantage over GRASP$_{pG}$. For all the instances, GRASP can obtain better solutions than GRASP$_{pG}$. Furthermore, the local search algorithm with the advanced objective function can get better results in low density networks compared with GRASP$_{pG}$, indicating that this advanced objective function is

Table 8
Characteristics of the COST239 and NSF instances.

Network	Nodes	Edges	Traffic set	1	3	12	Capacity
COST239	11	26	TS1	714	402	85	96
			TS2	1264	711	195	
			TS3	2746	1306	358	
NSF	14	42	TS1	1494	722	172	192
			TS2	2988	1444	344	
			TS3	5976	2888	688	

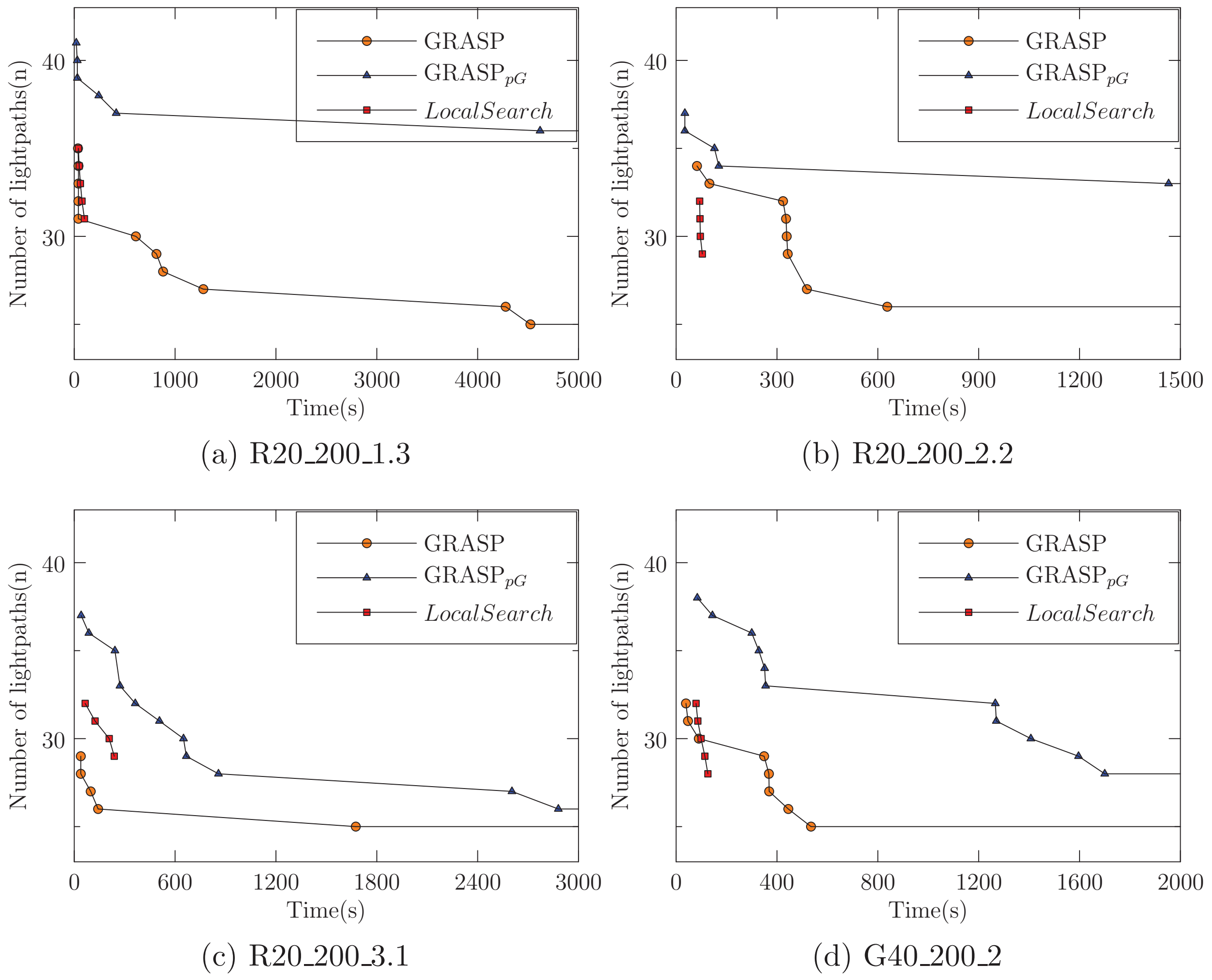


Fig. 5. Evolution of the objective value with the running time.

one of the most essential parts of our proposed algorithm. By analyzing the problem structure, we can find that the grooming problem and the routing problem are closely related with each other. Solving the grooming problem can help to partially solve the routing problem. Otherwise, the routing problem will become much difficult to solve. With the refined objective function, the advanced objective function helps to reduce the complexity of the routing problem. However, the two subproblems are not so closely related with each other when the network density increases. This may be the main reason why GRASP outperforms GRASP$_{pG}$ in all the cases, but the gap is not obvious for the instances with high network densities. In real practice, the WDM networks usually appear to be a sparse graph, so the mechanism that we propose here is useful.

From the above experiments, one finds that the instances with larger ratio of m to n are much more difficult to solve than other instances. Our GRASP algorithm obtains relatively poor results for these instances, and the instances with sparse network seem to be more difficult than those with dense networks. One reason might lie in the fact that there are less options for the route of each lightpath. This is an interesting phenomenon that is worthy of further investigation.

5. Conclusion

In this paper, we have presented a GRASP algorithm for tackling the grooming and routing problem with simple path constraint. The grooming and routing problem is widely considered in the optical telecommunications industry as well as in academia, while the simple path constraint has never been taken into account due to its high complexity.

By giving the mathematical formulation of the GR problem, we present in this paper a GRASP algorithm for solving it. The GRASP algorithm follows a general framework of an iterative process where each iteration consists of two phases: A construction phase and a local search phase. A number of distinguishing features are integrated in the algorithm. First, we have introduced two problem specific neighborhood structures for the grooming and routing problems and a refined objective function for the grooming procedure.

This refined objective function establishes a good connection between the two sub problems, helping the algorithm to obtain high quality results in an efficient way. It also gives good directions for those challenging problems which have such composite structures. Second, we combine the branch and bound algorithm with metaheuristics, introducing an approach to evaluate the neighborhoods in graph structure problems. Last but not the least, for the purpose of providing a better initial solution, we propose a mechanism of *Construction* to squeeze the traffic demands into the current virtual topology network as much as possible.

The performance of the GRASP algorithm is assessed on three sets of totally 38 problem instances which are generated according to real application scenarios as well as from real networks. Intensive computational results show the effectiveness and efficiency of our GRASP algorithm for solving the tested instances by comparison with the public software LocalSolver and the lower bounds.

The success of our GRASP algorithm reminds us that it is essential to introduce meaningful diversification mechanisms and highlight the problem specific knowledge in designing metaheuristic algorithms. In addition, the interdependence between sub-problems should be deeply considered when dealing with such complex problem with composite structures. Following this spirit, we hope to design even more robust and effective metaheuristic algorithms for solving the grooming and routing problem as well as other real-world composite optimization problems.

Acknowledgments

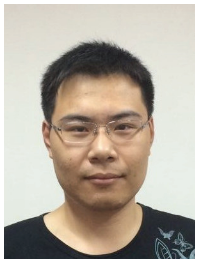
We thank the anonymous referees for their valuable comments to improve our paper significantly. The research was supported in part by the National Natural Science Foundation of China, China under grant number 61370183, 61100144, and the Program for New Century Excellent Talents in University, China (NCET 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] B. Gendron, T.G. Crainic, A. Frangioni, Multicommodity Capacitated Network Design, Springer, 1999.
[3] V. Sridhar, J.S. Park, Benders-and-cut algorithm for fixed-charge capacitated network design problem, Eur. J. Oper. Res. 125 (3) (2000) 622–632.
[4] I. Correia, L. Gouveia, F.S. da Gama, Discretized formulations for capacitated location problems with modular distribution costs, Eur. J. Oper. Res. 204 (2) (2010) 237–244.
[5] I. Ghamlouche, T.G. Crainic, M. Gendreau, Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design, Oper. Res. 51 (4) (2003) 655–667.
[6] E. Bartolini, J.-F. Cordeau, G. Laporte, An exact algorithm for the capacitated arc routing problem with deadheading demand, Oper. Res. 61 (2) (2013) 315–327.
[7] Y. Xu, R. Qu, Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods, J. Oper. Res. Soc. 62 (2) (2011) 313–325.
[8] A.P. Punnen, R. Zhang, Bottleneck flows in unit capacity networks, Inf. Process. Lett. 109 (6) (2009) 334–338.
[9] 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.
[10] H. Wang, G.N. Rouskas, Hierarchical traffic grooming: a tutorial, Comput. Netw. 69 (0) (2014) 147–156.
[11] B. Chen, G. Rouskas, R. Dutta, On hierarchical traffic grooming in WDM networks, IEEE/ACM Trans. Netw. 16 (5) (2008) 1226–1238.
[12] 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.
[13] 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.
[14] A. Rubio-Largo, M. Vega-Rodriguez, J. Gomez-Pulido, J. Sanchez-Perez, Multiobjective metaheuristics for traffic grooming in optical networks, IEEE Trans. Evol. Comput. 1 (2012) 457–473.
[15] X. Wang, W. Hou, L. Guo, J. Cao, D. Jiang, A new multi-granularity grooming algorithm based on traffic partition in IP over WDM networks, Comput. Netw. 55 (3) (2011a) 807–821.
[16] X. Wang, W. Hou, L. Guo, J. Cao, D. Jiang, Energy saving and cost reduction in multi-granularity green optical networks, Comput. Netw. 55 (3) (2011b) 676–688.
[17] W. Hou, L. Guo, X. Wei, X. Gong, Multi-granularity and robust grooming in power- and port-cost-efficient IP over wdm networks, Comput. Netw. 56 (10) (2012) 2383–2399.
[18] G. Wu, G. Mohan, Power-efficient integrated routing of sublambda connection requests with traffic splitting in IP over WDM networks, Comput. Netw. 70 (0) (2014) 16–29.
[19] J.-M. Hyppolite, P. Galinier, S. Pierre, A tabu search heuristic for the routing and wavelength assignment problem in multigranular optical networks, Photon. Netw. Commun. 15 (2) (2008) 123–130.
[20] C. Dzongang, P. Galinier, S. Pierre, A tabu search heuristic for the routing and wavelength assignment problem in optical networks, IEEE Commun. Lett. 9 (5) (2005) 426–428.
[21] J. Stolk, I. Mann, A. Mohais, Z. Michalewicz, Combining vehicle routing and packing for optimal delivery schedules, OR Insight 26 (3) (2013) 167–190.
[22] M. Bonyadi, Z. Michalewicz, L. Barone, The travelling thief problem: the first step in the transition from theoretical problems to realistic problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 1037–1044.
[23] F. Glover, M. Laguna, et al., Tabu Search, Springer, 1997.
[24] S. Kirkpatrick, M. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[25] H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search, in: International Series in Operations Research and Management Science, 2003, pp. 321–354.
[26] P. Moscato, C. Cotta, A. Mendes, Memetic algorithms, in: New Optimization Techniques in Engineering, Springer, 2004, pp. 53–85.
[27] T. Feo, M. Resende, Greedy randomized adaptive search procedures, J. Global Optim. 6 (2) (1995) 109–133.
[28] S. Consoli, K. Darby-Dowman, N. Mladenovi, J.M. Prez, Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem, Eur. J. Oper. Res. 196 (2) (2009) 440–449.
[29] D.-H. Lee, J.H. Chen, J.X. Cao, The continuous berth allocation problem: a greedy randomized adaptive search solution, Transp. Res. Part E: Logistics Transp. Rev. 46 (6) (2010) 1017–1029.
[30] E. Bartolini, A. Mingozzi, Algorithms for the non-bifurcated network design problem, J. Heuristics 15 (3) (2009) 259–281.
[31] A. Rubio-Largo, M.A. Vega-Rodrìguez, D.L. González-‘Alvarez, An improved multiobjective approach inspired by the flashing behaviour of fireflies for traffic grooming in optical wdm networks, Appl. Soft Comput. 21 (0) (2014) 617–636.
[32] T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua, Localsolver 1.x: a black-box local-search solver for 0–1 programming, 4OR 9 (3) (2011) 299–316.

Xinyun Wu received the BSc degree in computer science and technology from Naval University of Engineering, PLA, China, in 2009. He is a PhD candidate at School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include computational intelligence, traffic grooming routing and wavelength assignment algorithm in WDM network, and multicommodity network design problem.

Tao Ye received the PhD degree in computer science and technology from Huazhong University of Science and Technology, China, in 2012. His research interests include computational intelligence, circle packing problem and traffic grooming algorithm in WDM network.



Qi Guo received the BSc degree in computer science in 2012. Now he is a PhD candidate at the School of Computer Science and Technology at Huazhong University of Science and Technology, majored in computational intelligence, large-scale combinatorial optimization and scheduling problem.

Zhipeng Lü received the BSc degree in applied mathematics from Jilin University, China, in 2001 and the PhD degree in computer software and theory from Huazhong University of Science and Technology, China, in 2007. He was a postdoctoral research fellow at LERIA, Department of Computer Science, University of Angers, France, from 2007 to 2011. He is a professor at the School of Computer Science and Technology of Huazhong University of Science and Technology and director of the Laboratory of Smart Computing and Optimization (SMART) since March 2011. His research interests include artificial intelligence, computational intelligence, green computing, and adaptive metaheuristics for solving large-scale real-world and theoretical combinatorial optimization and constrained satisfaction problems.