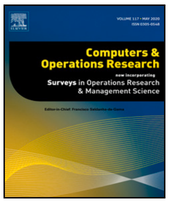
Computers & Operations Research 160 (2023) 106396

Contents lists available at ScienceDirect

Computers and Operations Research

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



Dual-neighborhood iterated local search for routing and wavelength assignment

Zhipeng Lü[a], Yuan Fang[b], Zhouxing Su[a,*], Yang Wang[c], Xinyun Wu[d], Fred Glover[e]

[a] *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China*
[b] *Alkaid, Huawei Technologies Co., Ltd., Xi'an, 710075, China*
[c] *School of Management, Northwestern Polytechnic University, Xi'an, 710072, China*
[d] *Hubei University of Technology, Wuhan, 430068, China*
[e] *Entanglement, Inc., Boulder, 80302, CO, USA*

ARTICLE INFO

Dataset link: https://github.com/HUST-Smart/DNILS-RWA

Keywords:
Routing and wavelength assignment
Iterated local search
Ejection chain
Metaheuristics

ABSTRACT

The minimum routing and wavelength assignment (min-RWA) problem is a classic and challenging NP-hard combinatorial optimization problem which aims to reduce the required wavelengths in wavelength-division multiplexing optical networks. In order to tackle this problem, we present a dual-neighborhood iterated local search (DN-ILS) by alternately evaluating a shift-shaking and a swap-shaking neighborhood to improve the current solution. Both neighborhoods are based on an ejection chain-based shaking (ECS) procedure, while the high-level procedures are shift moves and swap moves, respectively. These two move operators change the wavelengths of certain lightpaths, while the ECS refines routing for the related lightpaths. Experimental results on commonly used benchmark instances demonstrate that our proposed DN-ILS algorithm outperforms the best methods in the literature by improving the best known results for 35 out of 113 instances while matching the best known results for the remaining ones. The success of the shift-shaking and swap-shaking neighborhoods inspires us to apply them to other challenging optimization problems with multi-level decisions and strong constraints.

1. Introduction

Wavelength-division multiplexing (WDM) networks provide large bandwidth which allows transmitting different messages through the same fiber without mutual interference by assigning different wavelengths to different lightpaths traversing the same fiber, where a lightpath is a channel connecting two nodes through a series of consecutive fiber links (Lee et al., 2002). In practice, each lightpath is usually assigned a single wavelength, i.e., wavelength conversion is not allowed, which is known as the wavelength consistency constraint (Dutta et al., 2000).

Two important combinatorial optimization problems are involved in the WDM network: the traffic grooming problem for network design (Dawande et al., 2007) and the routing and wavelength assignment (RWA) problem for network operation (Chlamtac et al., 1992). The traffic grooming aims to minimize the cost of building a network which satisfies transmission demands (Wu et al., 2020), while RWA focuses on minimizing the consumption of wavelength resources for traffic routing in a fixed network. In this paper, we study the min-RWA problem, which deals with the assignment of wavelengths to all lightpaths as well as their detailed routing.

Wavelengths have become limited and expensive resources with the ever-growing lightpath requests (Wu et al., 2015). Accordingly, the min-RWA problem, which is NP-hard (Erlebach and Jansen, 2001), aims to route all lightpath requests using as few wavelengths as possible, becoming a focus of widespread attention. Numerous approaches and variants of the RWA were reviewed in Dutta et al. (2000) and Zang et al. (2000). In a first study, Chlamtac et al. (1992) proposed a greedy algorithm for the RWA. Then, various mathematical formulations were developed for the RWA, including integer linear programming (Ramaswami and Sivarajan, 1995), linear programming relaxation (Krishnaswamy and Sivarajan, 2001) and column generation (Jaumard et al., 2009). In view of the multi-level problem structure of min-RWA, there are usually two types of solution approaches. On the one hand, Li and Simha (2000) and Noronha and Ribeiro (2006) divided it into two subproblems: routing problem and wavelength assignment problem. On the other hand, Manohar et al. (2002) solved these two subproblems simultaneously. In addition, Skorin-Kapov (2007) pointed out the relation between bin packing and min-RWA, where the lightpath requests are regarded as items and copies of the original network are considered

* Corresponding author.
E-mail address: suzhouxing@hust.edu.cn (Z. Su).

https://doi.org/10.1016/j.cor.2023.106396
Received 27 December 2022; Received in revised form 23 June 2023; Accepted 17 August 2023
Available online 25 August 2023

as bins. Based on this analogy, Noronha et al. (2008) developed the best-fit decreasing (BFD) heuristic. Noronha et al. (2011) integrated BFD into a biased random-key genetic algorithm. More recently, Martins et al. (2012) proposed a variable neighborhood descent (VND) algorithm and Wu et al. (2016) designed a multi-neighborhood iterated tabu search (MN-ITS) for min-RWA. As far as we know, the most effective algorithms for min-RWA are VND and MN-ITS, and neither of them dominates the other on the hard instances introduced by Noronha et al. (2008).

In this paper, we propose an effective dual-neighborhood iterated local search (DN-ILS) algorithm based on shift-shaking and swap-shaking neighborhood structures for solving min-RWA. Our main contributions can be summarized as follows:

- Instead of directly solving min-RWA, we solve a series of decision subproblems k-RWA where k denotes the number of wavelengths used and is kept fixed in each subproblem. The idea of solving each k-RWA problem is to first assign all the lightpaths to k bins (macro-level decision) and then route the lightpaths in the same bin (micro-level decision) with the objective of minimizing the total overload of links in all the k bins. The search space of the min-RWA is much reduced and thus helps to conduct an intensified search.
- We propose two new effective neighborhoods called shift-shaking and swap-shaking for k-RWA and embed them into an iterated local search framework. In addition, we combine a two-level neighborhood evaluation, an incremental neighborhood evaluation and a caching technique to accelerate the search.
- Computational experiments on three sets of 113 benchmark instances demonstrate that our proposed algorithm outperforms the best methods in the literature by improving the best known results for 35 out of 113 instances while matching the best records for all remaining ones. More importantly, DN-ILS obtains the lower bound solutions of 8 instances for the first time.
- We conduct extensive experiments to verify the merits of the key components of our algorithm, including the two-level neighborhood evaluation, the incremental neighborhood evaluation and the caching technique.

Our algorithm extends and improves an earlier version named SAS-ILS (Fang et al., 2020). SAS-ILS shares similar local search framework with the newly proposed DN-ILS, but only employs the shift neighborhood to change the wavelength of a single lightpath. SAS-ILS also integrates an ejection chain-based shaking procedure to refine the routings of lightpaths assigned with the same wavelength. The basic idea of the ejection chain method (Glover, 1996) is to slightly relax the constraints so that more promising moves can be evaluated. The improvements of DN-ILS lies in the following aspects:

- We design a new neighborhood which swaps the wavelength assignments of two lightpaths. The search space of the swap-shaking neighborhood is larger than that of the shift-shaking neighborhood, thus making it easier to escape from local optima.
- We propose an effective neighborhood selection mechanism to achieve a trade-off between effectiveness and efficiency, based on the observation that the swap-shaking neighborhood usually results in larger improvement but is more time-consuming to evaluate.
- For generating the initial k-RWA solution, we break ties by adding a secondary objective (hop) to determine a bin when we fail to find a “best-fit” bin.
- We adopt two techniques to accelerate the local search, which are an incremental evaluation technique and a caching technique, respectively.
- We perform extensive experiments on four simplified versions of DN-ILS to provide insight into the importance of the DN-ILS ingredients.
- This improved version of our algorithm further improves the best-known results for 23 additional instances, and obtains the optimal solutions for 4 additional ones.

2. Problem formulations

Given an optical network $G = (V, E)$ where V is the vertex set and E is the set of directed links, let L be the lightpath set in which each lightpath $l \in L$ is associated with a source vertex s_l and a destination vertex d_l. Let $W = \{1, 2, \ldots, |W|\}$ be a sufficiently large set of wavelengths. Then, the min-RWA problem aims to assign each lightpath a wavelength and to route each lightpath to minimize the total number of used wavelengths, under the constraint that no two lightpaths in the same wavelength traverse the same link. There are three types of binary decision variables: When assigned a value of 1, λ_{lw} indicates that lightpath l is assigned the wavelength w, F_{ij}^{lw} indicates that lightpath l assigned the wavelength w traverses link (i, j), and F_w indicates that wavelength w is used. Then, the mixed-integer programming (MIP) model of the min-RWA problem can be defined as follows:

$$\min \sum_{w\in W} F_w \tag{1}$$

$$\text{s.t.} \sum_{l\in L} F_{ij}^{lw} \leq F_w, \quad \forall w \in W, (i,j) \in E \tag{2}$$

$$\sum_{i\in V} F_{ij}^{lw} - \sum_{k\in V} F_{jk}^{lw} = \begin{cases} -\lambda_{lw}, \text{ if } j = s_l; \\ \lambda_{lw}, \text{ if } j = d_l; \\ 0, \text{ otherwise.} \end{cases} \quad \forall l \in L, w \in W, j \in V \tag{3}$$

$$\sum_{w\in W} \lambda_{lw} = 1, \quad \forall l \in L \tag{4}$$

$$F_{ij}^{lw}, F_w, \lambda_{lw} \in \{0, 1\}, \ \forall l \in L, w \in W, (i,j) \in E \tag{5}$$

Objective (1) minimizes the number of activated wavelengths. Constraints (2) ensure that each used wavelength on each link will be counted in the objective function. They also impose the restriction that at most one lightpath assigned a given wavelength can traverse a given link. Constraints (3) require that each lightpath has a simple route from its source to its destination. Constraints (4) ensure that each lightpath is assigned a single wavelength.

The requirement that no more than k wavelengths are used gives rise to the decision version of the min-RWA problem called k-RWA, where we need to route each lightpath in L and assign a wavelength to it. In other words, W includes exactly k wavelengths ($|W| = k$), and the MIP model of k-RWA contains no objective but includes an additional constraint (6) together with Constraints (3)–(5), as summarized by:

$$\text{s.t.} \sum_{l\in L} F_{ij}^{lw} \leq 1, \quad \forall w \in W, (i,j) \in E \tag{6}$$

Constraints (3)–(5)

We refer to this model as k-RWA-D. A feasible solution of k-RWA-D is a feasible solution of min-RWA with an objective value of k. Fig. 1 gives an example of k-RWA with $k = 3$.

3. Dual neighborhood iterated local search

We decompose the min-RWA into a series of k-RWA decision subproblems, which is a popular framework for many optimization algorithms. For example, it is a common technique to reformulate the graph coloring problem into a series of k-coloring problems (Lü and Hao, 2010a); The p-center problem can be transformed into a set of fixed-radius covering problems (Zhang et al., 2020); The minimum vertex covering problem is usually decomposed into a series of k-set covering problems (Cai et al., 2013); The rectangle packing problem can be regarded as multiple strip packing problems (Wei et al., 2011). To be specific, we begin with a reasonable k. Once we find a feasible solution

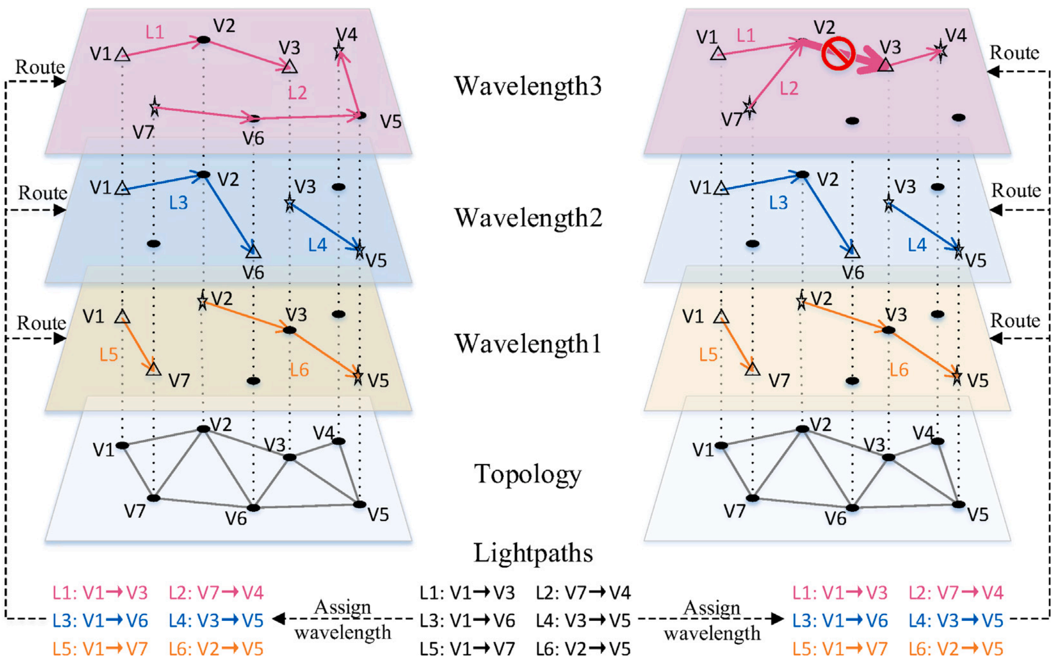


Fig. 1. An example of k-RWA with $k = 3$. There are 7 vertices and 11 bidirectional links in the topology, and 6 different lightpaths. The left figure presents a feasible solution of k-RWA, where each lightpath is assigned a wavelength and no two lightpaths traverse the same link if they are assigned the same wavelength. The right figure presents the infeasible routing for the two lightpaths $L1$ and $L2$ which are assigned Wavelength3.

for the k-RWA problem, we set $k \leftarrow k-1$ to seek a better solution for the min-RWA problem. If no feasible solution is found for the k-RWA problem, the algorithm stops and returns $k+1$ as the best objective value for the original min-RWA.

In this section, we present an iterated local search approach for the k-RWA problem based on the shift-shaking and swap-shaking neighborhoods. We also present several techniques to improve the efficiency of the proposed DN-ILS algorithm, including a two-level evaluation strategy, an incremental neighborhood evaluation method, and a caching technique for neighborhood evaluation.

3.1. Problem analysis and main framework

We regard the k-RWA problem as putting all the lightpaths (items) into k bins and routing the lightpaths such that the routing of any pair of lightpaths in the same bin do not share any link, i.e., the routing in each bin is conflict-free.

Moreover, inspired by Lü and Hao (2010b) for solving the graph coloring problem, we reformulate the decision problem k-RWA-D as an optimization problem k-RWA-O, where the MIP model for k-RWA-O can be obtained by relaxing constraints (6) in k-RWA-D and minimizing the total violations of these constraints. We say a link in a bin is overloaded if the number of lightpaths traversing this link is greater than one. Let $\delta_{ij}^{w} = max\{0, \sum_{l\in L} F_{ij}^{lw} - 1\}$ be the overload of link (i,j) in bin B_w and let δ^{w} stand for the total overload of bin B_w. Then, the k-RWA-O problem can be formulated as:

$$\min f(X) = \sum_{w\in W} \delta^{w} = \sum_{w\in W} \sum_{(i,j)\in E} \delta_{ij}^{w} \tag{7}$$

$$\text{s.t.} \sum_{l\in L} F_{ij}^{lw} \leq 1 + \delta_{ij}^{w}, \quad \forall w \in W,\ (i,j) \in E \tag{8}$$

$$\delta_{ij}^{w} \geq 0, \quad \forall w \in W,\ (i,j) \in E \tag{9}$$

Constraints (3)–(5)

Constraints (8) and (9) restrict δ_{ij}^{w} to be the overload of link (i,j) in bin B_w. The objective function $f(X)$ is the total overload. Apparently, a feasible solution X for k-RWA-D requires $f(X) = 0$ in k-RWA-O, where $X = \{B_1, B_2, \ldots, B_k\}$ and each bin B_w for $w = 1, 2, \ldots, k$ consists of some lightpaths from L assigned with wavelength w and their specific routing.

With this reformulation, the landscape of the solution space is smoothed which is more friendly to hill climbing algorithms like iterated local search. In detail, by relaxing the link overload constraints, i.e., allowing multiple lightpaths pass through the same edge using the same wavelength (but with penalty), we convert each k-RWA decision subproblem into a new optimization problem. This transformation replaces the coarse-grained wavelength number objective function with the fine-grained overload penalty, so that two solutions with the same objective value in the original optimization problem can be differentiated. Furthermore, combined with the ejection chain procedure introduced in Section 3.3.4, the gradient will not easily get vanished so that the search can escape from the local optima and saddle points with less effort compared with the original min-RWA model.

Based on the k-RWA-O model, our DN-ILS algorithm employs a simple but effective iterated local search framework, which is widely used to solve various combinatorial optimization problems (Carello et al., 2004; Katayama and Narihisa, 1999). The main framework of DN-ILS is described in Algorithm 1. Starting from an initial solution, the proposed algorithm executes a dual-neighborhood local search procedure and a perturbation procedure by turns.

Specifically, the local search procedure alternates between performing a shift-shaking move and a swap-shaking move until a local optimum is reached. Then, the perturbation procedure makes some small changes to the best solution found so far to drive the search into a new promising area. The search-and-perturb loop is repeated until reaching a specified time limit or the current solution X becomes feasible for k-RWA-D, i.e., $f(X) = 0$.

3.2. Initial solution

Algorithm 2 shows our initial solution generation procedure for the k-RWA problem, which adapts the BFD-RWA algorithm in Skorin-Kapov (2007). The main difference between our initial solution generation and BFD-RWA lies in two aspects: (1) the number of bins in our

Algorithm 1 DN-ILS: Main framework for k-RWA

```
Input: The physical network $G$, the lightpath set $L$ and the number of wavelengths $k$
Output: The best solution found so far $X_{best}$
1: $X \leftarrow$ Init-Solution($G, L, k$) /* Section 3.2 */
2: $X_{best} \leftarrow X$
3: repeat
4:     $X \leftarrow$ DN-LS($X$) /* Section 3.4 */
5:     if $f(X) = 0$ then
6:         return $X$
7:     end if
8:     if $f(X) < f(X_{best})$ then
9:         $X_{best} \leftarrow X$
10:    end if
11:    $X \leftarrow$ Perturbation($X_{best}$) /* Section 3.5 */
12: until the time limit is met
```

Algorithm 2 Init-Solution: Initial solution generation

```
Input: The physical network $G$, the lightpath set $L$ and the number of wavelengths $k$
Output: The initial solution $X$
1: Create $k$ bins $B_1, B_2, ..., B_k$ by copying the network $G$
2: $W \leftarrow \{B_1, B_2, ..., B_k\}$
3: Sort the lightpaths of $L$ in non-increasing order by the length of their shortest routes that minimizes $hop$
4: for $l_i$ in $L$ do
5:     Find the "best-fit" bin $B_w \in W$ for $l_i$
6:     if such bin does not exist then
7:         Determine a bin $B_w \in W$ by minimizing $overload(route(l_i, w))$ as the primary objective and $hop(route(l_i, w))$ as the secondary objective
8:     end if
9:     Insert $l_i$ with $route(l_i, w)$ into bin $B_w$
10: end for
11: $X \leftarrow (B_1, B_2, ..., B_k)$
12: return $X$
```

algorithm is fixed, whereas this number is the objective to be optimized in BFD-RWA; (2) BFD-RWA always keeps the routing of all lightpaths conflict-free while we minimize the conflicts of lightpaths.

Specifically, we duplicate the network G to create k copies and stipulate that each network B_w corresponds to a bin w. For each link (i, j) of B_w, we use $overload(w, i, j) = \delta^w_{ij}$ and $hop(w, i, j) = 1$ to measure its weight, respectively. For each lightpath l_i assigned to bin B_w, we use $route(l_i, w)$ to denote the links traversed by l_i in B_w. The total weight of this route is denoted by the overload weight $overload(route(l_i, w))$ and the hop weight $hop(route(l_i, w))$. Initially, the shortest route of each lightpath in any bin using the hop weight is calculated independently. Then, we sort all lightpaths $l \in L$ by the hop route length in non-increasing order. For the first lightpath in the sorted list, we choose a "best-fit" bin for it to insert. The "best-fit" bin must satisfy the following conditions: (1) there exists a route for lightpath l_i in this bin that does not conflict with other lightpaths, i.e., $overload(route(l_i, w)) = 0$; (2) inserting lightpath l_i in this bin will yield the shortest hop route length subject to the condition $hop(route(l_i, w)) < \max(diam(G), \sqrt{|E|})$. When a "best-fit" bin does not exist, we minimize $overload(route(l_i, w))$ as the primary objective and $hop(route(l_i, w))$ as the secondary objective to determine a bin to insert. We repeat the above-mentioned procedure until all the lightpaths $l_i \in L$ are inserted into proper bins.

3.3. Neighborhoods definition

The definition of the neighborhood structure is one of the most important features to distinguish different local search algorithms. In this section, we first present the main idea of our Shift-Shaking (SS) and Swap-Shaking (SWS) neighborhoods. Then, we describe the common component of SS and SWS: an ejection chain shaking procedure.

3.3.1. Main idea

In RWA, the assignments of lightpaths to bins and the routing of lightpaths are macro-level and micro-level decisions, respectively. The decisions at these two levels can seriously interfere with each other. Thus, we consider changing the wavelength and the route of a lightpath as a whole. The overload is not caused by a single lightpath, so we need to modify the routing of other related lightpaths when resolving conflicts. As a result, we design two new neighborhoods, combining a shift (swap) move with two shaking procedures so as to make decisions at the macro and micro levels simultaneously. Specifically, we perform a shift (swap) move to change the wavelength of one lightpath (two lightpaths) while executing the shaking procedure to reroute related lightpaths in bins B_{old} and B_{new} to reduce the total overload as much as possible.

3.3.2. Shift-shaking neighborhood

Given a solution X, operator $mv_1(l_c, B_{old}, B_{new})$ denotes the Shift-Shaking neighborhood move, and $X' = X \oplus mv_1(l_c, B_{old}, B_{new})$ denotes a neighboring solution of X. The operator $mv_1(l_c, B_{old}, B_{new})$ has two components: (1) A high-level shift move to re-assign the bin (i.e., wavelength) of a conflicting lightpath l_c from its current bin B_{old} to another bin B_{new}; (2) Two low-level shaking operations to optimize the routing of the lightpaths in bins B_{old} and B_{new}. Let $M_1(X)$ be the set of all possible moves of mv_1. Then, we can use $N_1(X) = \{X' : X' = X \oplus mv_1, mv_1 \in M_1(X)\}$ to denote the neighborhood of X. Fig. 2 gives a small example of our SS neighborhood.

3.3.3. Swap-shaking neighborhood

Given a solution X, the Swap-Shaking neighborhood is defined by operator $mv_2(l_1, l_2)$, which transforms X into a neighboring solution $X' = X \oplus mv_2(l_1, l_2)$. The operator $mv_2(l_1, l_2)$ also has a two-level structure: (1) A high-level swap move to swap the bins (wavelengths) of two lightpaths (l_1 and l_2); (2) Two low-level shaking operations to further improve the routing in the old bins of l_1 and l_2. Let $M_2(X)$ be the set of all possible moves of mv_2, then $N_2(X) = \{X' : X' = X \oplus mv_2, mv_2 \in M_2(X)\}$ represents the neighborhood of X.

3.3.4. Ejection chain-based shaking procedure

The ejection chain-based shaking (ECS) procedure ECS(l_c, B, X) inserts lightpath l_c into a new bin B and reroutes the lightpaths in B to minimize the total overload in bin B, as described in Algorithm 3. The ejection chain method has been successfully used to tackle many NP-hard problems (Rego and Glover, 2010), including traveling salesman problem (Glover, 1996), vehicle routing problem (Rego and Roucairol, 1996), generalized assignment problem (Yagiura et al., 2006) and quadratic multiple knapsack problem (Peng et al., 2016).

As shown in Algorithm 3, there are mainly four subroutines in our ECS procedure: GreedyInsert, EjectionMove, ReroutingMove and TrialMove. Specifically, GreedyInsert inserts lightpath l_c into bin B and finds a route r that minimizes $overload(r)$ as the primary objective and $hop(r)$ as the secondary objective, while keeping the routing of other lightpaths unchanged (line 1). EjectionMove removes one of the most overloaded lightpaths l_e from this bin to form an initial partial solution (line 2). ReroutingMove first removes a conflicting lightpath and then greedily inserts it back into the bin by GreedyInsert to form a new partial solution (line 8). Based on one of the partial solutions

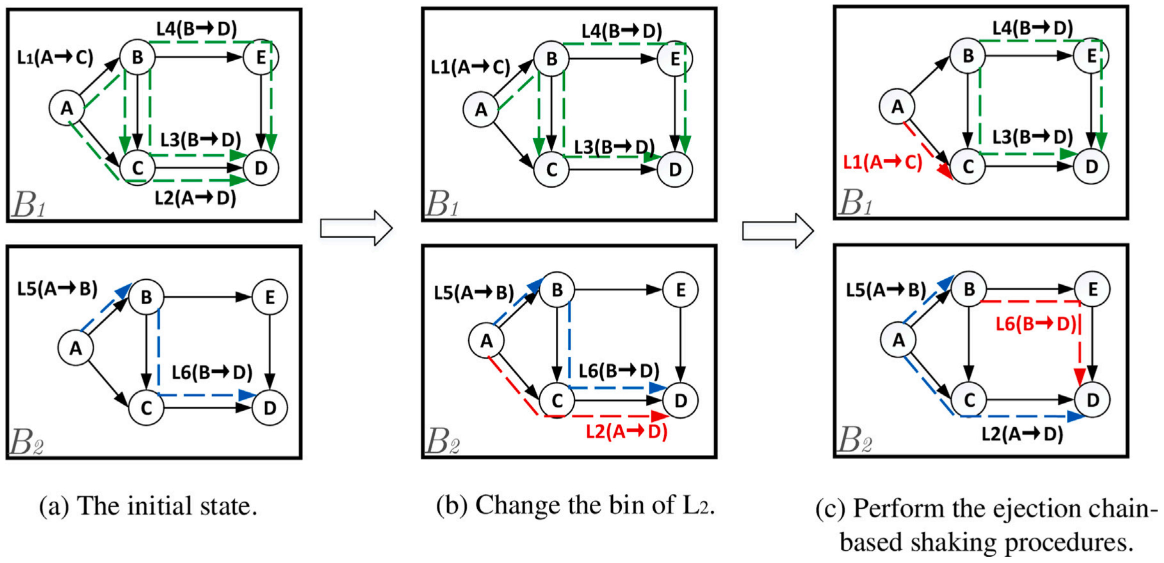

L1(A→C)
L4(B→D)
L3(B→D)
L2(A→D)
L5(A→B)
L6(B→D)
B1
B2
(a) The initial state.
(b) Change the bin of L2.
(c) Perform the ejection chain-based shaking procedures.

Fig. 2. Illustration of making an SS neighborhood move $mv_1(L_2, B_1, B_2)$. (a) Initially, links $\vec{BC}$ and $\vec{CD}$ in B_1 are overloaded ($\delta^1 = 2$) and B_2 is conflict-free ($\delta^2 = 0$). (b) Changing the bin of L_2 from B_1 to B_2 without shaking cannot reduce the total overload ($\delta^1 = 1$ and $\delta^2 = 1$). (c) Performing two shaking procedures for bins B_1 and B_2 will reroute L_1 and L_6, thus eliminating the overload ($\delta^1 = 0$ and $\delta^2 = 0$).

Algorithm 3 ECS: Ejection chain-based shaking procedure

```
Input: A conflicting lightpath $l_c$, a bin $B$ and the current solution $X$
Output: The best solution $X_{best}$ after inserting $l_c$ into $B$
1: $X^I \leftarrow$ GreedyInsert($l_c, B, X$), $X_{best} \leftarrow X^I$
2: $\{X^R, l_e\} \leftarrow$ EjectionMove($X^I$)
3: $no\_improve\_iter \leftarrow 0$
4: while $no\_improve\_iter < max\_iter$ do
5:     $X^R_{prev} \leftarrow X^R$, $P \leftarrow \emptyset$
6:     $L_o \leftarrow \{l_i \in L(B) : l_i$ is a conflicting lightpath in $B\}$
7:     for $l_i$ in $L_o$ do
8:         $X^R_i \leftarrow$ ReroutingMove($l_i, B, X^R$)
9:         $P \leftarrow P \cup X^R_i$
10:    end for
11:    if $rand[0,1) \leq \alpha$ then
12:        $X^R \leftarrow$ a random partial solution in $P$
13:    else
14:        $X^R \leftarrow$ the best partial solution in $P$
15:    end if
16:    $X^T \leftarrow$ TrialMove($l_e, B, X^R$)
17:    if $f(X^T) < f(X_{best})$ then
18:        $X_{best} \leftarrow X^T$
19:    end if
20:    if $f(X^R) \geq f(X^R_{prev})$ then
21:        $no\_improve\_iter \leftarrow no\_improve\_iter + 1$
22:    else
23:        $no\_improve\_iter \leftarrow 0$
24:    end if
25: end while
26: return $X_{best}$
```

generated by ReroutingMove, TrialMove greedily inserts the initially ejected lightpath l_e into the bin by GreedyInsert to build a complete solution (line 16).

GreedyInsert and EjectionMove are performed once to obtain an initial partial solution, while ReroutingMove and TrialMove are iteratively performed until stagnation is encountered (line 4). At each iteration, we first reroute each conflicting lightpath in B, which will produce a set P of candidate partial solutions (lines 6–10). Then, in order to obtain a new complete solution X^T by TrialMove, a random solution or the best solution in the candidate set P is chosen with probability α and $1 - \alpha$ to be the reference solution X^R, respectively (lines 11–16). As a result, a new complete solution can be obtained, and the best one will be saved by replacing X_{best} (lines 17–19). This process is repeated until the local optimum is reached, i.e., the partial solution X^R has not been improved for max_iter consecutive steps.

Furthermore, when performing ECS on the old bin of l_c, the subroutines GreedyInsert, EjectionMove and TrialMove are skipped, i.e., we just shake the bin to reach a local optimum. The reason for this simplification is that the removal of lightpath l_c releases resources so that other lightpaths can utilize them. Thus, it becomes unnecessary to eject another lightpath l_e, which means that TrialMove also does not need to be performed. For illustrative purposes, Fig. 3 demonstrates the ECS procedure on a small example.

3.4. Dual-neighborhood local search

Algorithm 4 describes the dual-neighborhood local search procedure. At each iteration, the algorithm performs a local search based on the shift-shaking neighborhood (SS-LS, Algorithm 5) with probability sc or the swap-shaking neighborhood (SWS-LS, Algorithm 6) with probability $1 - sc$. In this way, the shift-shaking and swap-shaking neighborhoods can be explored alternately to diversify the search.

It can be time-consuming to evaluate all moves at each iteration of the local search. To improve the search efficiency, SS-LS and SWS-LS adopt the following four methods. The first two speed up the search by restricting the search space, while the remaining two accelerate the neighborhood evaluation: (1) Only one of the conflicting lightpaths is randomly chosen and all neighborhood moves that assign it to a new bin are considered; (2) The neighborhood moves are evaluated by a two-level method which combines a coarse-grained approach and a fine-grained approach; (3) An incremental evaluation technique is used; (4) A caching technique is employed to store the unchanged neighborhood evaluation results after a move.

3.4.1. SS-based local search

As presented in Algorithm 5, SS-LS begins at a randomly picked conflicting lightpath l_c from X (line 2). Then, our SS neighborhood is constructed, which is the set of solutions obtained by moving lightpath

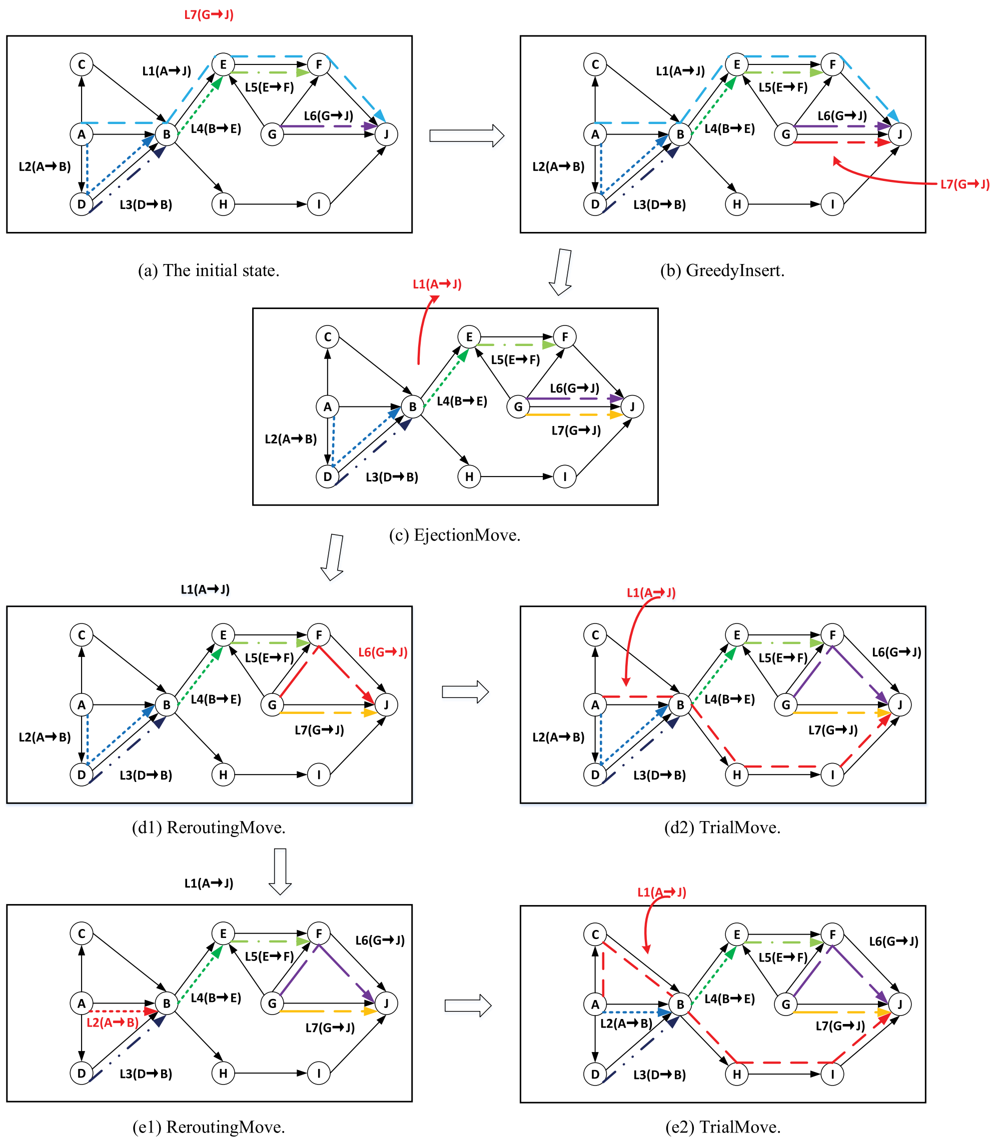


Fig. 3. An example of the ECS procedure. (a) presents the initial state of a bin, where there are 6 lightpaths ($L_1 - L_6$) and 3 overloaded links ($\vec{DB}$, $\vec{BE}$ and $\vec{EF}$) in the bin ($\delta = 3$). We want to insert L_7 into the bin by ECS. (b) shows the state after inserting L_7 by GreedyInsert. There are three possible routes for L_7 in the bin: $G \to J$ ($\delta = 4$), $G \to F \to J$ ($\delta = 4$) and $G \to E \to F \to J$ ($\delta = 5$). Therefore, the route found by GreedyInsert is $G \to J$. (c) presents the state after EjectionMove, which removes the most overloaded lightpath, i.e., L_1. (d1) and (e1) respectively show the reference solution X_{d1} ($\delta = 1$) and X_{e1} ($\delta = 0$) obtained by performing ReroutingMove, while (d2) and (e2) respectively present the complete solution X_{d2} ($\delta = 1$) and X_{e2} ($\delta = 0$) obtained by inserting the ejected lightpath into their corresponding reference solution. Therefore, the best complete solution is X_{e2}.

l_c from its original bin to another one. This not only reduces the size of the neighborhood and improves efficiency, but also diversifies the search as an additional advantage. In detail, we remove l_c from its old bin B_{old} which results in a partial solution X' (line 4). Next, we try to insert lightpath l_c in the best bin according to the two-level evaluation method, as illustrated in Fig. 4. The evaluation of new bins is carried out in two steps: First, a quick coarse-grained evaluation considers all bins except B_{old} to obtain a set of candidate bins; Second, a fine-grained evaluation finds the best bin for l_c and the corresponding routing of affected lightpaths in the candidate bin set. Technically, we construct the set $N_{insert}(l_c, X')$ which greedily minimizes the overload, i.e., the solutions obtained by GreedyInsert(l_c, B_{new}, X') which inserts l_c into B_{new} with a min-cost route. The candidate bin set C_B is the ones corresponding to the m-best $f(X'')$ among $X'' \in N_{insert}(l_c, X')$ (line 5). The fine-grained evaluation employs an ejection chain-based shaking procedure to reroute the lightpaths in each candidate bin to reduce the overload resulting from the insertion of lightpath l_c. The search will move to the best neighbor X^* if it is better than X. If not, we still give a small probability β to accept X^* in order to avoid getting trapped in local optima. SS-LS performs the above procedure until it fails to improve the best solution X_{best} within a specified number of iterations (max_iter2).

3.4.2. SWS-based local search

As presented in Algorithm 6, SWS-LS is similar to Algorithm 5, except that its search space is much larger than that of SS-LS. Starting from a randomly selected conflicting lightpath l_1 (line 2), SWS-LS evaluates all the solutions obtained by swapping l_1 with another lightpath. Next, we select the best lightpath to swap with l_1 by a two-level evaluation strategy, which obtains m candidate lightpaths by the coarse-grained evaluation and selects the best lightpath from the m candidates by the fine-grained evaluation. Algorithm 7 presents the coarse-grained evaluation for SWS, which starts by selecting the m-best bins for l_1 to insert by GreedyInsert. Then, we insert l_1 into the m selected bins to obtain the candidate set T_L, which contains all the conflicting lightpaths after inserting l_1. The m-best lightpaths to swap are selected from T_L, by performing a swap move with GreedyInsert.

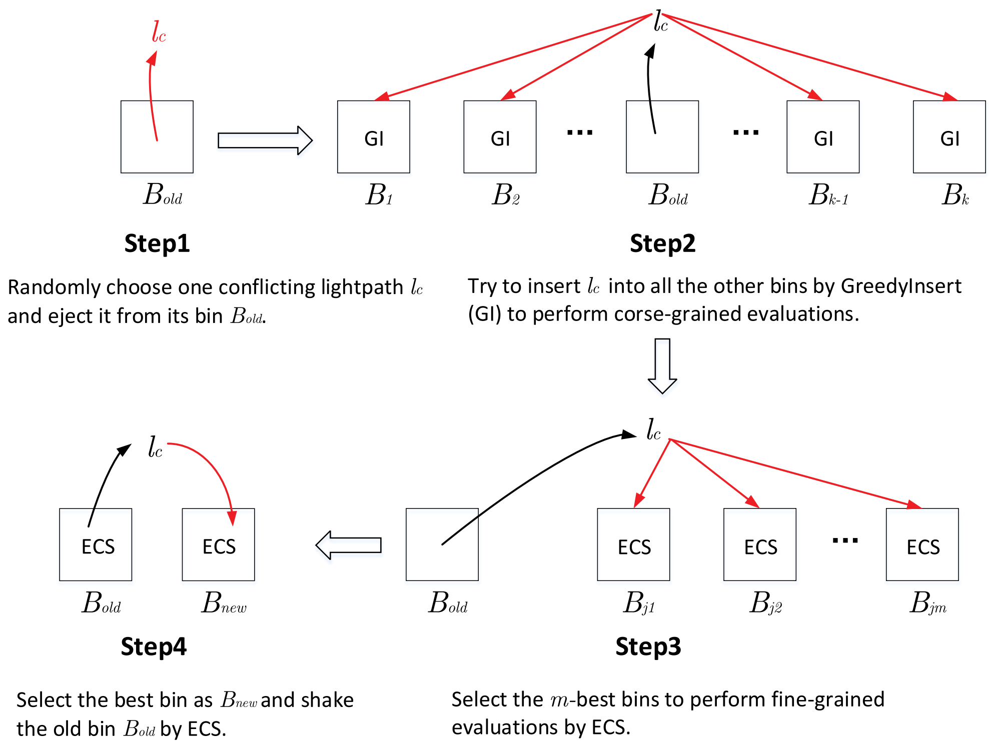

l_c
GI
B_{old}
B_1
B_2
B_{k-1}
B_k
Step1
Randomly choose one conflicting lightpath l_c and eject it from its bin B_{old}.
Step2
Try to insert l_c into all the other bins by GreedyInsert (GI) to perform corse-grained evaluations.
ECS
B_{new}
B_{j1}
B_{j2}
B_{jm}
Step3
Select the m-best bins to perform fine-grained evaluations by ECS.
Step4
Select the best bin as B_{new} and shake the old bin B_{old} by ECS.

Fig. 4. Two-level neighborhood evaluation for the SS neighborhood.

Algorithm 4 DN-LS: Dual-neighborhood local search

```
Input: The current solution $X$
Output: The best solution $X_{best}$ found in the local search
1: $no\_improve\_iter \leftarrow 0$
2: $X_{best} \leftarrow X$
3: while $no\_improve\_iter < max\_iter2$ do
4:   if rand$[0,1) < sc$ then
5:     $X^* \leftarrow$ SS-LS($X$)
6:   else
7:     $X^* \leftarrow$ SWS-LS($X$)
8:   end if
9:   if $f(X^*) \geq f(X)$ then
10:    $no\_improve\_iter \leftarrow no\_improve\_iter + 1$
11:  end if
12:  if $f(X^*) < f(X)$ or $rand[0,1) < \beta$ then
13:    $X \leftarrow X^*$
14:    $no\_improve\_iter \leftarrow 0$
15:  end if
16:  if $f(X^*) < f(X_{best})$ then
17:    $X_{best} \leftarrow X^*$
18:  end if
19: end while
```

Algorithm 5 SS-LS: SS-based Local search procedure

```
Input: The current solution $X$
Output: The best solution $X_{best}$ found in the local search
1: $X_{best} \leftarrow X$
2: Pick a random conflicting lightpath $l_c$ from $X$
3: $B_{old} \leftarrow$ the bin of $l_c$
4: $X' \leftarrow$ Remove $l_c$ from $B_{old}$
5: Search $X'' \in N_{insert}(l_c, X')$ whose $f(X'')$ is the $m$-best and add the
   bin to set $C_B$ /* Coarse-grained evaluation */
6: $f^* \leftarrow \infty$
7: for $B_{new}$ in $C_B$ do /* Fine-grained evaluation */
8:   $X'' \leftarrow$ ECS($l_c, B_{new}, X'$)   /* Shake new bin $B_{new}$ */
9:   if $f(X'') < f^*$ then
10:    $f^* \leftarrow f(X'')$
11:    $X^* \leftarrow X''$
12:  end if
13: end for
14: $X^* \leftarrow$ ECS($\varnothing, B_{old}, X^*$)   /* Shake old bin $B_{old}$ */
15: if $f(X^*) < f(X)$ or $rand[0,1) < \beta$ then
16:   $X_{best} \leftarrow X^*$
17: end if
```

Finally, we evaluate the candidates using ECS and select the best lightpath l_2 to swap with l_1.

3.4.3. Incremental evaluation

According to the definition of the neighborhood structure, there are only minor differences between the current solution and its neighbors, i.e., only one or several lightpaths in at most two bins are changed. Therefore, to evaluate the objective function value of the neighboring solutions, we employ an incremental evaluation technique that only considers the changed bins for accelerating the search. We denote the two bins that are changed by moving from the current solution X to the candidate neighboring solution X' by B_1 and B_2. Let Δ^w be the increment of the overload of bin B_w. The objective value of X' can be calculated as:

$$f(X') = f(X) + \Delta^1 + \Delta^2 \tag{10}$$

To calculate the increment of the overload of a changed bin B_w, we need to consider all the changed (deleted, inserted or rerouted) lightpaths in B_w. The rerouting move for one lightpath can be decomposed into first deleting the lightpath from its bin and then reinserting it into the bin with another route. Therefore, the changes of a bin can be expressed as a process of deleting and/or inserting particular lightpaths.

Algorithm 6 SWS-LS: SWS-based local search procedure

```
Input: The current solution $X$
Output: The best solution $X_{best}$ found in the local search
 1: $X_{best} \leftarrow X$
 2: Randomly select a conflicting lightpath $l_1$ from $X$
 3: $B_1 \leftarrow$ the bin of $l_1$
 4: $X' \leftarrow$ Remove $l_1$ from $B_1$
 5: $C_L \leftarrow$ Obtain-m-lightpath($l_1, X'$) /* Coarse-grained */
 6: for $l_2$ in $C_L$ do /* Fine-grained evaluation */
 7:     $B_2 \leftarrow$ the bin of $l_2$
 8:     $X' \leftarrow$ Remove $l_2$ from $B_2$
 9:     $X'' \leftarrow$ ECS($l_2, B_1, X'$)   /* Shake new bin $B_1$ */
10:     $X'' \leftarrow$ ECS($l_1, B_2, X''$)   /* Shake new bin $B_2$ */
11:     if $f(X'') < f^*$ then
12:         $f^* \leftarrow f(X'')$
13:         $X^* \leftarrow X''$
14:     end if
15: end for
16: if $f(X^*) < f(X)$ or $rand[0,1) < \beta$ then
17:     $X_{best} \leftarrow X^*$
18: end if
```

Algorithm 7 Obtain-m-lightpath: Coarse-grained evaluation

```
Input: The lightpath to swap $l_1$ and the current solution $X'$
Output: $m$-best candidate lightpaths set $C_L$ to be swapped
 1: Search $X'' \in N_{insert}(l_1, X')$ whose $f(X'')$ is the $m$-best and add the
    bin into set $C_B$
 2: $T_L \leftarrow \emptyset$
 3: for $B_2$ in $C_B$ do
 4:     GreedyInsert($l_1, B_2, X'$)
 5:     $T_L \leftarrow T_L \cup$ the conflicted lightpaths in $B_2$
 6: end for
 7: $C_L \leftarrow \emptyset$
 8: for $l_2$ in $T_L$ do
 9:     $B_2 \leftarrow$ the bin of $l_2$
10:     $X'' \leftarrow$ GreedyInsert($l_1, B_2, X'$)
11:     $X'' \leftarrow$ GreedyInsert($l_2, B_1, X''$)
12:     if $f(X'')$ is the $m$-best then
13:         $C_L \leftarrow C_L \cup \{l_2\}$
14:     end if
15: end for
```

Let D_w and I_w respectively denote the sets of deleted and inserted lightpaths in B_w and Δ_{ij}^w denote the increment of the overload of link (i,j) in bin B_w. Let R_l and R'_l respectively denote the route of lightpath l in X and X'. Then, the increment of the overload of B_w can be calculated as:

$$\Delta^w = \sum_{l\in D_w} \sum_{(i,j)\in R_l} \Delta_{ij}^w + \sum_{l\in I_w} \sum_{(i,j)\in R'_l} \Delta_{ij}^w \tag{11}$$

By recording the number of lightpaths on link (i,j) in bin B_w as n_{ij}^w, we can quickly calculate the increment of overload on this link. When l is deleted from B_w, the overload will decrease by 1 for each link (i,j) on the route of l that contains more than one lightpath and will remain unchanged for links containing only one lightpath, as shown in Eq. (12). When l is inserted into B_w, the overload will increase by 1 if there are one or more lightpaths, and will remain unchanged if there are no other lightpaths on each link (i,j) that lightpath l traverses, as given by Eq. (13).

$$\Delta_{ij}^w = \begin{cases} -1, & \text{if } n_{ij}^w > 1 \\ 0, & \text{if } n_{ij}^w = 1 \end{cases} \quad \forall (i,j) \in R_l \tag{12}$$

Algorithm 8 Perturbation: Perturbation procedure for k-RWA

```
Input: Previous best found solution $X_{best}$
Output: The new solution $X'$ after perturbation
 1: $L_c \leftarrow$ the conflicting lightpath set of $X_{best}$
 2: $p \leftarrow min(P, |L_c|)$
 3: $L_p \leftarrow$ randomly select $p$ lightpaths from $L_c$
 4: $X' \leftarrow$ remove all the lightpaths in $L_p$ from $X_{best}$
 5: for $l_i$ in $L_p$ do
 6:     $\Psi \leftarrow$ select $\gamma$% of bins from $W$ randomly
 7:     Try to insert $l_i$ into each bin $B_w \in \Psi$ with a route that minimizes
        $\delta^w$
 8:     Choose the bin $B_w \in \Psi$ and the route of $l_i$ such that the increment
        of overload is the smallest in $\Psi$
 9:     $X' \leftarrow$ insert $l_i$ into bin $B_w$
10: end for
11: return $X'$;
```

$$\Delta_{ij}^w = \begin{cases} 1, & \text{if } n_{ij}^w \geq 1 \\ 0, & \text{if } n_{ij}^w = 0 \end{cases} \quad \forall (i,j) \in R'_l \tag{13}$$

3.4.4. Caching technique for neighborhood evaluation

Algorithm 5 changes only two bins after one iteration of local search. However, for a selected lightpath l_c, we need to evaluate all the bins (except the old bin) to select the best bin to insert, which is time-consuming. In fact, to determine the best bin, we only need to calculate the new objective value of the candidate solution X' after inserting l_c into a bin by GreedyInsert or ECS based on solution X.

We exploit this fact by recording the increment of the overload between X' and X after inserting l_c into bin B_w as $\Delta_{l_c}^w$, creating a cache for those bins which are not changed after one neighborhood move. Therefore, when l_c is selected to shift its bin again, we can quickly retrieve $\Delta_{l_c}^w$ from the cache for those bins which have not been changed since the last cache recalculation and then derive the objective value of the candidate solution X' based on the current solution by $f(X') = f(X) + \Delta_{l_c}^w$. After performing a neighborhood move, all values in the cache related to the two changed bins will be recalculated according to the incremental evaluation method. For the SWS neighborhood, similar techniques can be applied.

3.5. Perturbation

Algorithm 8 gives the perturbation procedure. It first randomly picks a subset of conflicting lightpaths L_p from the conflicting lightpath set L_c of X_{best}, where $|L_p| = min(P, |L_c|)$ and P is a parameter. Then, we cancel the routing of each lightpath $l_i \in L_p$, and insert l_i into the best bin among a set of candidate bins where the overload is minimized based on GreedyInsert. The candidate bin sets consist of γ% bins randomly picked from all bins.

4. Computational results and analysis

In this section, we report and compare the experimental results of several state-of-the-art reference algorithms for RWA and our DN-ILS on the commonly used benchmark instances.[1]

[1] The code of our DN-ILS algorithm and the obtained best known solutions are available at https://github.com/HUST-Smart/DNILS-RWA.

Table 1
Settings of parameters.

Parameter	Candidate values	Final value	Description	Section
max_iter	2, 3, 4	3	The maximum number of consecutive non-improvement iterations in ECS.	3.3
α	0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3	0.2	The probability of choosing a random solution in ECS.	3.3
β	0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3	0.1	The probability of accepting worse solution in local search procedure.	3.4
m	2, 3, 4	2	The number of candidate bins in local search procedure.	3.4
max_iter2	750, 800, 850, 900, 1000	850	The maximum number of consecutive non-improvement iterations in local search procedure.	3.4
P	3, 4, 5	4	The maximum number of selected conflicting lightpaths in perturbation procedure.	3.5
γ	10, 20, 30, 40	20	The percent of selected bins in perturbation procedure.	3.5

4.1. Benchmark instances and experimental protocol

Our experiments were performed on three sets of instances which are denoted by W, Y and Z, respectively. Set W contains 13 real-world instances which were widely studied in the literature. Sets Y and Z were introduced in Noronha et al. (2008) and each set includes up to 100 nodes and 10,000 lightpaths, which are of larger scale and thus more challenging. There are 75 instances in set Y which can be further divided into three groups whose densities of links are 0.03 (Y3), 0.04 (Y4) and 0.05 (Y5), respectively. The 25 instances in set Z are built on $n \times m$ grids embedded on the torus where each node only connects to its nearest four neighboring nodes. The initial k of each instance is obtained by executing the BFD-RWA method described in Skorin-Kapov (2007). Our DN-ILS method then repeatedly solves the k-RWA by decreasing k by 1 after obtaining a conflict-free solution until the time limit is reached.

Our DN-ILS algorithm was programmed in C++ and tested with a 2.3 GHz E5-2698 CPU and 8 GB RAM for 5 independent runs on each instance. As the reference algorithms were tested on 2 GHz CPUs under a 5-minute time limit, we set the time limit (from computing the initial k to the best k found so far) of DN-ILS to 4 min, to make a fair comparison by considering the difference of CPU performance.

Table 1 presents the descriptions and settings of several key parameters used in our DN-ILS algorithm, where we consider all combinations of the candidate values for each parameter on several representative instances and apply the final selected value for all the tested instances.

4.2. Comparison with reference algorithms

We compare DN-ILS with the reference algorithms VND (Martins et al., 2012), MN-ITS (Wu et al., 2016) and SAS-ILS (preliminary version of DN-ILS) (Fang et al., 2020). Table 2 presents the results of the set W instances. Tables 3–5 report the results of the sets Y3, Y4 and Y5 instances, respectively. Table 6 gives the results of the set Z instances. For each instance, column LB reports the lower bound obtained by relaxing the wavelength consistency constraints and integer constraints of the flow decision variables (Banerjee and Mukherjee, 1996). For each algorithm, column Obj presents the best results and column Gap gives the relative objective gap as a percentage, i.e., $(Obj - LB)/LB \times 100\%$. Column Succ reports the success rate over 5 runs of our DN-ILS. Column CPU presents the average CPU time for obtaining the best results, i.e., the total computational time from obtaining the initial wavelength number to stopping at the best one. Row Average gives the average objective gap, CPU time and success rate on the listed instances. For each algorithm on each instance, if the objective value is optimal, i.e., it matches the lower bound, the corresponding result is marked with “*”. The improved best known results are indicated in bold, while the tied results are indicated in italics.

Table 2 shows that all three algorithms can reach the lower bounds for all the 13 instances in set W. The average time of our algorithm is less than a second and is slightly shorter than MN-ITS. Tables 3–5 disclose that DN-ILS obtains the optimal solutions on 48 out of 75 instances in set Y, and reaches the lower bounds for 3 instances (Y.3.60.4, Y.3.80.4 and Y.3.100.4) for the first time. These results also demonstrate the robustness and efficiency of DN-ILS, for it obtains the best results with almost 100% success rate and within a very short time for all the instances in set Y. In detail, DN-ILS hits the best results for 52 out of 75 instances within a minute, and takes 80.35 s, 36.66 s and 21.25 s on average to reach the best results for sets Y3, Y4 and Y5, respectively. Regarding the performance of the reference algorithms on set Y, even if MN-ITS failed to hit the new upper bounds within 5 min on 33 instances, it still spent 110.38 s, 44.24 s and 34.18 s on average for converging to its best results, which is over 20% longer than our DN-ILS. Similar phenomenon can be found when comparing the efficiency of SAS-ILS and the proposed DN-ILS. Comparing these outcomes to the best results obtained by VND and MN-ITS, DN-ILS is able to find better solutions on 22 instances, while matching the best known results on the remaining 53 ones.

In Table 6, we additionally compare DN-ILS with MN-ITS and VND on set Z. For these instances, DN-ILS reaches the lower bounds for 11 out of 25 instances and improves the best known results for 13 instances, while matching the best known results for the remaining 12 ones. In particular, the lower bounds for instances Z.4 $\times$ 25.60, Z.4 $\times$ 25.100, Z.5 $\times$ 20.80, Z.6 $\times$ 17.100 and Z.8 $\times$ 13.20 are obtained for the first time, indicating that these 5 instances are closed. DN-ILS is also very robust for this dataset. The success rates of our proposed algorithm are 100% on set Z except for only 4 instances. However, one can observe that DN-ILS improves the best results on these 4 instances. In addition, the success rates will be 100% if we stop upon reaching the previous best known results. Regarding computational efficiency, DN-ILS reaches the best results on 12 instances within a minute and spends 72.87 s on average for obtaining the best results on set Z, while MN-ITS can only obtain worse results with 46.91% longer run time.

In summary, DN-ILS improves the best known results for 35 out of 113 instances, while matching the best known results for all remaining ones. Moreover, DN-ILS reaches the lower bounds on 8 instances for the first time. These statistics demonstrate the advantages of the proposed DN-ILS algorithm in terms of both effectiveness and efficiency.

4.3. Importance of DN-ILS ingredients

To evaluate the merits of the shift-shaking and swap-shaking neighborhoods, the two-level neighborhood evaluation approach, the incremental evaluation method and the caching technique, we conducted experiments to compare DN-ILS with its simplified versions (SD-ILS, SEC-ILS, SSWI-ILS, SSWC-ILS) on the four largest instances (Y.3.100.1, Y.4.100.5, Z.6 $\times$ 17.100 and Z.10 $\times$ 10.100) with the best k reported in Tables 2 to 6. Specifically, SD-ILS employs a different neighborhood which keeps the high-level shift/swap move but replaces the two shaking procedures with GreedyInsert (thus greedily finding a route for the lightpath to be inserted into the new bin, while keeping the routes of other lightpaths in the bin unchanged). SEC-ILS also employs the same shift and swap neighborhoods, but evaluates all bins by fine-grained ejection chain-based shaking. SSWI-ILS replaces the incremental evaluation technique with the naive neighborhood evaluation, where the objective is calculated from scratch. SSWC-ILS disables the caching technique of neighborhood evaluations, which means that it directly

Table 2
Computational results and comparisons on set W instances.

Instance	LB	VND		MN-ITS			DN-ILS			
		Obj	Gap (%)	Obj	Gap (%)	CPU (s)	Obj	Gap (%)	CPU (s)	Succ (%)
ATT	20	20*	0.00	20*	0.00	3.43	20*	0.00	3.78	100.00
ATT2	113	113*	0.00	113*	0.00	2.64	113*	0.00	1.85	100.00
Brasil	48	48*	0.00	48*	0.00	0.07	48*	0.00	0.06	100.00
EON	22	22*	0.00	22*	0.00	0.01	22*	0.00	0.01	100.00
Finland	46	46*	0.00	46*	0.00	0.46	46*	0.00	0.40	100.00
NSF.1	22	22*	0.00	22*	0.00	0.25	22*	0.00	0.18	100.00
NSF.3	22	22*	0.00	22*	0.00	0.43	22*	0.00	0.31	100.00
NSF.12	38	38*	0.00	38*	0.00	1.46	38*	0.00	1.14	100.00
NSF.48	41	41*	0.00	41*	0.00	0.23	41*	0.00	0.32	100.00
NSF2.1	21	21*	0.00	21*	0.00	0.03	21*	0.00	0.04	100.00
NSF2.3	21	21*	0.00	21*	0.00	0.08	21*	0.00	0.08	100.00
NSF2.12	35	35*	0.00	35*	0.00	0.34	35*	0.00	0.26	100.00
NSF2.48	39	39*	0.00	39*	0.00	0.03	39*	0.00	0.03	100.00
Average			0.00		0.00	0.73		0.00	0.65	100.00

Table 3
Computational results and comparisons on set Y3 instances.

Instance	LB	VND		MN-ITS			SAS-ILS				DN-ILS			
		Obj	Gap (%)	Obj	Gap (%)	CPU (s)	Obj	Gap (%)	CPU (s)	Succ (%)	Obj	Gap (%)	CPU (s)	Succ (%)
Y.3.20.1	27	29	7.41	29	7.41	73.57	29	7.41	61.09	100.00	29	7.41	25.82	100.00
Y.3.20.2	33	33*	0.00	33*	0.00	0.26	33*	0.00	0.21	100.00	33*	0.00	0.15	100.00
Y.3.20.3	29	29*	0.00	29*	0.00	33.10	29*	0.00	19.71	100.00	29*	0.00	16.34	100.00
Y.3.20.4	26	28	7.69	28	7.69	43.38	28	7.69	25.02	100.00	28	7.69	15.31	100.00
Y.3.20.5	28	28*	0.00	29	3.57	201.96	28*	0.00	154.65	100.00	28*	0.00	156.70	100.00
Y.3.40.1	53	57	7.55	57	7.55	100.67	57	7.55	77.83	100.00	**56**	5.66	131.63	80.00
Y.3.40.2	59	59*	0.00	59*	0.00	1.06	59*	0.00	0.55	100.00	59*	0.00	0.47	100.00
Y.3.40.3	61	61*	0.00	61*	0.00	247.23	61*	0.00	135.06	100.00	61*	0.00	14.46	100.00
Y.3.40.4	50	54	8.00	54	8.00	40.60	54	8.00	31.8	100.00	**53**	6.00	107.65	60.00
Y.3.40.5	53	56	5.66	56	5.66	93.74	56	5.66	47.93	100.00	**55**	3.77	186.61	80.00
Y.3.60.1	81	87	7.41	86	6.17	122.81	86	6.17	105.89	100.00	**85**	4.94	137.47	60.00
Y.3.60.2	89	89*	0.00	89*	0.00	1.50	89*	0.00	1.13	100.00	89*	0.00	1.00	100.00
Y.3.60.3	91	91*	0.00	91*	0.00	155.10	91*	0.00	80.45	100.00	91*	0.00	14.44	100.00
Y.3.60.4	78	80	2.56	80	2.56	113.46	79	1.28	87.74	100.00	**78***	0.00	131.08	40.00
Y.3.60.5	77	82	6.49	82	6.49	131.08	**81**	5.19	135.99	100.00	**81**	3.90	88.60	20.00
Y.3.80.1	106	115	8.49	114	7.55	143.72	114	7.55	124.38	100.00	**113**	6.60	161.60	100.00
Y.3.80.2	117	117*	0.00	117*	0.00	3.27	117*	0.00	2.29	100.00	117*	0.00	1.68	100.00
Y.3.80.3	118	118*	0.00	118*	0.00	227.40	118*	0.00	144.03	80.00	118*	0.00	123.70	80.00
Y.3.80.4	105	106	0.95	106	0.95	150.20	**105***	0.00	86.91	100.00	**105***	0.00	63.86	100.00
Y.3.80.5	104	109	4.81	109	4.81	197.97	108	3.85	121.61	100.00	**107**	2.88	111.58	20.00
Y.3.100.1	131	143	9.16	141	7.63	299.65	141	7.63	221.13	80.00	**140**	6.87	214.13	40.00
Y.3.100.2	146	146*	0.00	146*	0.00	5.82	146*	0.00	3.16	100.00	146*	0.00	2.66	100.00
Y.3.100.3	146	146*	0.00	146*	0.00	29.27	146*	0.00	25.59	100.00	146*	0.00	19.11	100.00
Y.3.100.4	131	132	0.76	132	0.76	165.10	**131***	0.00	122.35	100.00	**131***	0.00	100.82	100.00
Y.3.100.5	129	136	5.43	136	5.43	177.49	135	4.65	129.53	100.00	**134**	3.88	181.82	80.00
Average			3.29		3.28	110.38		2.91	77.84	98.40		2.38	80.35	82.40

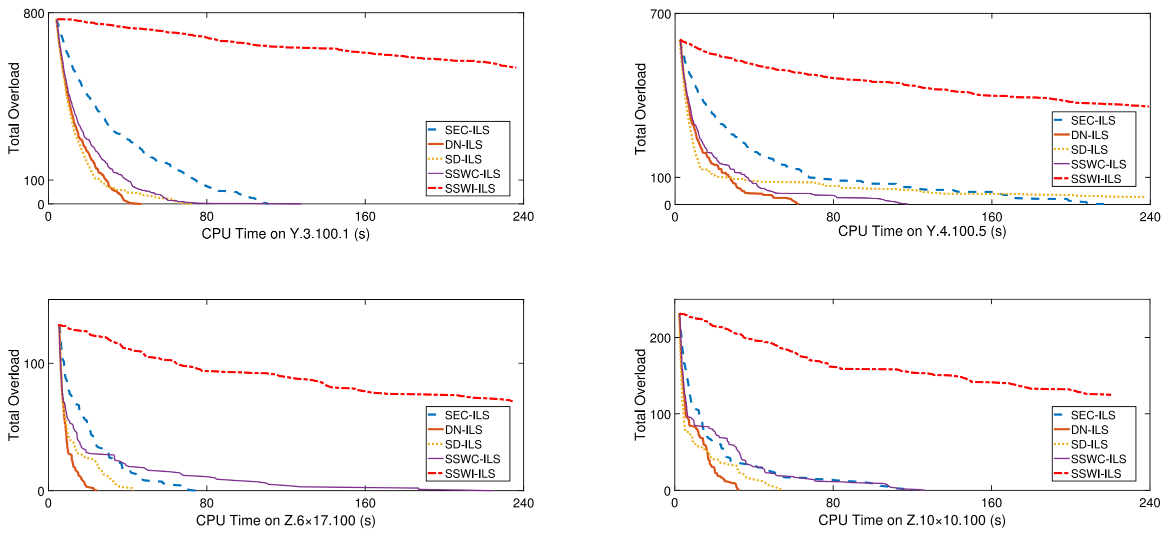


Fig. 5. Evolution of the total overload by DN-ILS and its variants on four largest instances. Point (x, y) means that the best total overload is y at x seconds.

Table 4
Computational results and comparisons on set Y4 instances.

Instance	LB	VND		MN-ITS			SAS-ILS				DN-ILS			
		Obj	Gap (%)	Obj	Gap (%)	CPU (s)	Obj	Gap (%)	CPU (s)	Succ (%)	Obj	Gap (%)	CPU (s)	Succ (%)
Y.4.20.1	17	19	11.76	19	11.76	12.82	19	11.76	2.29	100.00	**18**	5.88	89.88	60.00
Y.4.20.2	28	_28*_	0.00	_28*_	0.00	0.69	_28*_	0.00	0.45	100.00	_28*_	0.00	0.52	100.00
Y.4.20.3	23	_23*_	0.00	_23*_	0.00	0.18	_23*_	0.00	0.12	100.00	_23*_	0.00	0.11	100.00
Y.4.20.4	19	_19*_	0.00	_19*_	0.00	6.30	_19*_	0.00	5.24	100.00	_19*_	0.00	3.04	100.00
Y.4.20.5	17	_19_	11.76	_19_	11.76	5.08	_19_	11.76	3.82	100.00	_19_	11.76	5.86	100.00
Y.4.40.1	31	_35_	12.90	_35_	12.90	20.51	_35_	12.90	15.09	100.00	_35_	12.90	33.09	20.00
Y.4.40.2	57	_57*_	0.00	_57*_	0.00	0.94	_57*_	0.00	0.67	100.00	_57*_	0.00	0.57	100.00
Y.4.40.3	43	_43*_	0.00	_43*_	0.00	0.86	_43*_	0.00	0.46	100.00	_43*_	0.00	0.39	100.00
Y.4.40.4	38	_38*_	0.00	_38*_	0.00	7.98	_38*_	0.00	2.64	100.00	_38*_	0.00	2.26	100.00
Y.4.40.5	33	37	12.12	37	12.12	14.27	37	12.12	9.05	100.00	**36**	9.09	106.08	100.00
Y.4.60.1	47	53	12.77	53	12.77	88.82	**52**	10.64	53.07	100.00	**52**	10.64	42.17	100.00
Y.4.60.2	86	_86*_	0.00	_86*_	0.00	24.76	_86*_	0.00	19.90	100.00	_86*_	0.00	19.27	100.00
Y.4.60.3	64	_64*_	0.00	_64*_	0.00	3.19	_64*_	0.00	2.36	100.00	_64*_	0.00	2.33	100.00
Y.4.60.4	58	_58*_	0.00	_58*_	0.00	1.33	_58*_	0.00	0.81	100.00	_58*_	0.00	0.62	100.00
Y.4.60.5	49	55	12.24	55	12.24	52.86	**54**	10.20	52.45	100.00	**54**	10.20	43.15	100.00
Y.4.80.1	62	70	12.90	69	11.29	299.74	69	11.29	180.28	100.00	**68**	9.68	125.54	100.00
Y.4.80.2	118	_118*_	0.00	_118*_	0.00	17.36	_118*_	0.00	8.80	100.00	_118*_	0.00	11.15	100.00
Y.4.80.3	81	_81*_	0.00	_81*_	0.00	4.88	_81*_	0.00	3.53	100.00	_81*_	0.00	3.24	100.00
Y.4.80.4	78	_78*_	0.00	_78*_	0.00	1.62	_78*_	0.00	1.17	100.00	_78*_	0.00	1.01	100.00
Y.4.80.5	65	72	10.77	71	9.23	87.10	71	9.23	45.28	100.00	**70**	7.69	126.08	100.00
Y.4.100.1	76	86	13.16	86	13.16	93.44	85	11.84	63.75	100.00	**84**	10.53	143.83	100.00
Y.4.100.2	146	_146*_	0.00	_146*_	0.00	25.55	_146*_	0.00	16.02	100.00	_146*_	0.00	38.17	100.00
Y.4.100.3	98	_98*_	0.00	_98*_	0.00	31.89	_98*_	0.00	17.31	100.00	_98*_	0.00	8.27	100.00
Y.4.100.4	98	_98*_	0.00	_98*_	0.00	75.13	_98*_	0.00	15.26	100.00	_98*_	0.00	11.57	100.00
Y.4.100.5	80	89	11.25	89	11.25	228.83	**87**	8.75	150.35	100.00	**87**	8.75	98.19	100.00
Average			4.87		4.74	44.24		4.42	26.80	100.00		3.89	36.66	95.20

Table 5
Computational results and comparisons on set Y5 instances.

Instance	LB	VND		MN-ITS			SAS-ILS				DN-ILS			
		Obj	Gap (%)	Obj	Gap (%)	CPU (s)	Obj	Gap (%)	CPU (s)	Succ (%)	Obj	Gap (%)	$CPU(s)$	Succ (%)
Y.5.20.1	13	_13*_	0.00	_13*_	0.00	4.62	_13*_	0.00	2.52	100.00	_13*_	0.00	2.22	100.00
Y.5.20.2	17	_17*_	0.00	_17*_	0.00	17.39	_17*_	0.00	15.48	100.00	_17*_	0.00	1.30	100.00
Y.5.20.3	12	_12*_	0.00	13	8.33	102.08	_12*_	0.00	76.58	100.00	_12*_	0.00	74.35	80.00
Y.5.20.4	17	_17*_	0.00	_17*_	0.00	0.35	_17*_	0.00	0.27	100.00	_17*_	0.00	0.28	100.00
Y.5.20.5	15	_15*_	0.00	_15*_	0.00	0.16	_15*_	0.00	0.08	100.00	_15*_	0.00	0.09	100.00
Y.5.40.1	24	_24*_	0.00	_24*_	0.00	9.66	_24*_	0.00	7.13	100.00	_24*_	0.00	3.71	100.00
Y.5.40.2	31	_31*_	0.00	_31*_	0.00	3.77	_31*_	0.00	2.75	100.00	_31*_	0.00	1.47	100.00
Y.5.40.3	22	_23_	4.55	_23_	4.55	13.41	_23_	4.55	6.87	100.00	_23_	4.55	5.90	100.00
Y.5.40.4	33	_33*_	0.00	_33*_	0.00	5.20	_33*_	0.00	2.91	100.00	_33*_	0.00	2.06	100.00
Y.5.40.5	28	_28*_	0.00	_28*_	0.00	0.98	_28*_	0.00	0.69	100.00	_28*_	0.00	0.69	100.00
Y.5.60.1	33	_35_	6.06	_35_	6.06	29.99	_35_	6.06	28.76	100.00	_35_	6.06	34.44	100.00
Y.5.60.2	45	_45*_	0.00	_45*_	0.00	10.92	_45*_	0.00	7.86	100.00	_45*_	0.00	3.29	100.00
Y.5.60.3	34	_34*_	0.00	_34*_	0.00	14.41	_34*_	0.00	7.64	100.00	_34*_	0.00	7.18	100.00
Y.5.60.4	48	_48*_	0.00	_48*_	0.00	50.06	_48*_	0.00	34.00	100.00	_48*_	0.00	13.89	100.00
Y.5.60.5	40	_40*_	0.00	_40*_	0.00	4.55	_40*_	0.00	2.41	100.00	_40*_	0.00	2.22	100.00
Y.5.80.1	43	46	6.98	47	9.30	26.79	46	6.98	26.04	100.00	**45**	4.65	60.20	20.00
Y.5.80.2	59	_59*_	0.00	_59*_	0.00	27.14	_59*_	0.00	25.86	100.00	_59*_	0.00	10.84	100.00
Y.5.80.3	43	_44_	2.33	45	4.65	38.67	_44_	2.33	19.40	100.00	_44_	2.33	17.01	100.00
Y.5.80.4	63	_63*_	0.00	_63*_	0.00	146.78	_63*_	0.00	73.57	100.00	_63*_	0.00	41.15	100.00
Y.5.80.5	53	_53*_	0.00	_53*_	0.00	5.01	_53*_	0.00	4.26	100.00	_53*_	0.00	3.93	100.00
Y.5.100.1	55	57	3.64	57	3.64	59.60	57	3.64	33.30	100.00	**56**	1.82	118.53	60.00
Y.5.100.2	73	_73*_	0.00	74	1.37	58.37	_73*_	0.00	48.54	100.00	_73*_	0.00	29.82	100.00
Y.5.100.3	53	_54_	1.89	55	3.77	32.80	_54_	1.89	51.68	100.00	_54_	1.89	39.76	100.00
Y.5.100.4	77	_77*_	0.00	_77*_	0.00	180.49	_77*_	0.00	103.90	100.00	_77*_	0.00	50.72	100.00
Y.5.100.5	66	_66*_	0.00	_66*_	0.00	11.24	_66*_	0.00	6.60	100.00	_66*_	0.00	6.15	100.00
Average			1.02		1.67	34.18		1.02	23.56	100.00		0.85	21.25	94.40

calculates the incremental evaluation objective by calling GreedyInsert or ECS.

Fig. 5 shows that although SD-ILS quickly improves the solution quality in the early stage, DN-ILS outperforms SD-ILS by taking less computation time to obtain a feasible solution. The reason for this phenomenon might be that without the shaking procedure, the routing of the lightpaths in both bins is not sufficiently optimized to reach a local optimum. That is to say, good candidate high-level shift-moves may be overlooked due to an inferior evaluation when only using the greedy heuristic. It is noteworthy that DN-ILS is much faster than SEC-ILS in all cases, demonstrating the importance of the combined neighborhood evaluation method. When the incremental evaluation is disabled, SSWI-ILS converges slowly such that the objective value is still high when the time limit is reached, justifying the importance of the incremental evaluation technique. In addition, when the caching technique is disabled, SSWC-ILS is much slower than DN-ILS in all cases, demonstrating the importance of this technique.

5. Conclusion

Two novel neighborhoods, called shift-shaking and swap-shaking, are proposed to tackle the classic min-RWA problem. In detail, our

Table 6
Computational results and comparisons on set Z instances.

Instance	LB	VND		MN-ITS			SAS-ILS				DN-ILS			
		Obj	Gap (%)	Obj	Gap (%)	CPU (s)	Obj	Gap (%)	CPU (s)	Succ (%)	Obj	Gap (%)	CPU (s)	Succ (%)
Z.4 × 25.20	66	68	3.03	*66**	0.00	299.53	*66**	0.00	211.35	20.00	*66**	0.00	122.00	100.00
Z.4 × 25.40	126	129	2.38	*127*	0.79	34.71	*127*	0.79	31.71	100.00	*127*	0.79	27.41	100.00
Z.4 × 25.60	192	195	1.56	*193*	0.52	40.58	*193*	0.52	21.69	100.00	**192***	0.00	53.69	40.00
Z.4 × 25.80	257	261	1.56	*258*	0.39	101.22	*258*	0.39	65.05	100.00	*258*	0.39	104.51	100.00
Z.4 × 25.100	312	319	2.24	317	1.60	234.76	315	0.96	180.09	100.00	**312***	0.00	136.89	60.00
Z.5 × 20.20	54	55	1.85	*54**	0.00	0.86	*54**	0.00	0.86	100.00	*54**	0.00	0.69	100.00
Z.5 × 20.40	101	104	2.97	*101**	0.00	115.50	*101**	0.00	107.18	100.00	*101**	0.00	76.85	100.00
Z.5 × 20.60	154	158	2.60	*154**	0.00	299.61	*154**	0.00	203.93	40.00	*154**	0.00	146.55	100.00
Z.5 × 20.80	205	209	1.95	206	0.49	103.39	**205***	0.00	56.88	100.00	**205***	0.00	81.09	100.00
Z.5 × 20.100	250	256	2.40	253	1.20	80.08	252	0.80	59.40	100.00	**251**	0.40	134.70	60.00
Z.6 × 17.20	44	46	4.55	*44**	0.00	40.52	*44**	0.00	32.30	100.00	*44**	0.00	80.73	100.00
Z.6 × 17.40	84	87	3.57	*85*	1.19	10.70	*85*	1.19	9.34	100.00	*85*	1.19	6.89	100.00
Z.6 × 17.60	128	133	3.91	*129*	0.78	15.55	*129*	0.78	13.48	100.00	*129*	0.78	20.02	100.00
Z.6 × 17.80	171	176	2.92	*171**	0.00	165.17	*171**	0.00	119.83	100.00	*171**	0.00	169.33	100.00
Z.6 × 17.100	216	222	2.78	220	1.85	93.25	217	0.46	89.66	100.00	**216***	0.00	128.97	100.00
Z.8 × 13.20	33	35	6.06	34	3.03	72.05	**33***	0.00	60.35	100.00	**33***	0.00	16.61	100.00
Z.8 × 13.40	63	67	6.35	*64*	1.59	78.87	*64*	1.59	45.49	100.00	*64*	1.59	27.06	100.00
Z.8 × 13.60	96	101	5.21	98	2.08	109.98	**97**	1.04	79.52	100.00	**97**	1.04	100.15	100.00
Z.8 × 13.80	129	134	3.88	*130*	0.78	52.78	*130*	0.78	42.17	100.00	*130*	0.78	36.45	100.00
Z.8 × 13.100	168	175	4.17	173	2.98	150.49	**169**	0.60	176.50	100.00	**169**	0.60	37.03	100.00
Z.10 × 10.20	27	31	14.81	29	7.41	141.37	**28**	3.70	112.46	100.00	**28**	3.70	47.94	100.00
Z.10 × 10.40	51	59	15.69	55	7.84	171.94	**54**	5.88	130.57	100.00	**54**	5.88	33.11	100.00
Z.10 × 10.60	77	88	14.29	84	9.09	124.33	82	6.49	70.22	100.00	**81**	5.19	58.60	40.00
Z.10 × 10.80	103	116	12.62	112	8.74	54.35	109	5.83	45.92	100.00	**108**	4.85	75.03	100.00
Z.10 × 10.100	125	142	13.60	139	11.20	84.61	134	7.20	54.02	100.00	**133**	6.40	99.51	100.00
Average			5.48		2.54	107.05		1.56	80.80	94.40		1.34	72.87	92.00

local search-based algorithm incorporates high-level shift and swap moves to change the wavelengths of lightpaths and two low-level shaking procedures to further optimize the routing of the associated lightpaths. Our approach additionally introduces several strategies to accelerate the neighborhood evaluation. Computational experiments on 113 commonly used benchmark instances show that DN-ILS obtains the optimal solutions on 8 instances for the first time, improves the best known results for 35 instances and matches the best known results for the remaining 78 ones within short computation time. Although our SS and SWS neighborhoods are specially designed for the k-RWA problem, their success inspires us to adapt their underlying ideas to tackle other multi-level optimization problems with strong constraints.

CRediT authorship contribution statement

Zhipeng Lü: Conceptualization, Methodology, Writing – review & editing, Supervision. **Yuan Fang:** Methodology, Software, Validation, Formal analysis, Writing – original draft. **Zhouxing Su:** Methodology, Visualization, Writing – original draft, Writing – review & editing. **Yang Wang:** Investigation, Writing – review & editing. **Xinyun Wu:** Methodology, Writing – review & editing. **Fred Glover:** Methodology, Writing – review & editing.

Data availability

The data is already publicly available and the code is available at https://github.com/HUST-Smart/DNILS-RWA.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 72101094 and the Special Project for Knowledge Innovation of Hubei Province under Grant 2022013301015175.

References

Banerjee, D., Mukherjee, B., 1996. A practical approach for routing and wavelength assignment in large wavelength-routed optical networks. IEEE J. Sel. Areas Commun. 14 (5), 903–908.
Cai, S., Su, K., Luo, C., Sattar, A., 2013. NuMVC: An efficient local search algorithm for minimum vertex cover. J. Artificial Intelligence Res. 46, 687–716.
Carello, G., Croce, F.D., Ghirardi, M., Tadei, R., 2004. Solving the Hub location problem in telecommunication network design: A local search approach. Networks 44 (2), 94–105.
Chlamtac, I., Ganz, A., Karmi, G., 1992. Lightpath communications: An approach to high bandwidth optical WAN's. IEEE Trans. Commun. 40 (7), 1171–1182.
Dawande, M., Gupta, R., Naranpanawe, Sriskandarajah, C., 2007. A traffic-grooming algorithm for wavelength-routed optical networks. INFORMS J. Comput. 19 (4), 565–574.
Dutta, R., Rouskas, G.N., et al., 2000. A survey of virtual topology design algorithms for wavelength routed optical networks. Opt. Netw. Mag. 1 (1), 73–89.
Erlebach, T., Jansen, K., 2001. The complexity of path coloring and call scheduling. Theoret. Comput. Sci. 255 (1–2), 33–50.
Fang, Y., Lü, Z., Su, Z., Wang, Y., Zhang, T., Zhang, Q., 2020. Local search based on a new neighborhood for routing and wavelength assignment. In: 2020 IEEE Int. Conf. Syst. Man Cybern.. SMC, pp. 1123–1128.
Glover, F., 1996. Ejection chains, reference structures and alternating path methods for traveling salesman problems. Discrete Appl. Math. 65 (1–3), 223–253.
Jaumard, B., Meyer, C., Thiongane, B., 2009. On column generation formulations for the RWA problem. Discrete Appl. Math. 157 (6), 1291–1308.
Katayama, K., Narihisa, H., 1999. Iterated local search approach using genetic transformation to the traveling salesman problem. In: Proc. Genetic Evol. Computation Conf., Vol. 1. Morgan Kaufmann Publishers Inc., pp. 321–328.
Krishnaswamy, R.M., Sivarajan, K.N., 2001. Algorithms for routing and wavelength assignment based on solutions of LP-relaxations. IEEE Trans. Commun. 5 (10), 435–437.
Lee, K., Kang, K.C., Lee, T., Park, S., 2002. An optimization approach to routing and wavelength assignment in WDM all-optical mesh networks without wavelength conversion. ETRI J. 24 (2), 131–141.
Li, G., Simha, R., 2000. The partition coloring problem and its application to wavelength routing and assignment. In: Proc. 1st Workshop Opt. Networks. Citeseer, p. 1.
Lü, Z., Hao, J., 2010a. A memetic algorithm for graph coloring. European J. Oper. Res. 203 (1), 241–250.
Lü, Z., Hao, J.-K., 2010b. A memetic algorithm for graph coloring. European J. Oper. Res. 203 (1), 241–250.
Manohar, P., Manjunath, D., Shevgaonkar, R., 2002. Routing and wavelength assignment in optical networks from edge disjoint path algorithms. IEEE Trans. Commun. 6 (5), 211–213.

Martins, A.X., Duhamel, C., Mahey, P., Saldanha, R.R., de Souza, M.C., 2012. Variable neighborhood descent with iterated local search for routing and wavelength assignment. Comput. Oper. Res. 39 (9), 2133–2141.

Noronha, T.F., Resende, M.G., Ribeiro, C.C., 2008. Efficient implementations of heuristics for routing and wavelength assignment. In: Exp. Algorithms 7th Int. Workshop, WEA 2008. Springer, pp. 169–180.

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

Noronha, T.F., Ribeiro, C.C., 2006. Routing and wavelength assignment by partition colouring. European J. Oper. Res. 171 (3), 797–810.

Peng, B., Liu, M., Lü, Z., Kochengber, G., Wang, H., 2016. An ejection chain approach for the quadratic multiple knapsack problem. European J. Oper. Res. 253 (2), 328–336.

Ramaswami, R., Sivarajan, K.N., 1995. Routing and wavelength assignment in all-optical networks. IEEE/ACM Trans. Netw. 3 (5), 489–500.

Rego, C., Glover, F.W., 2010. Ejection chain and filter-and-fan methods in combinatorial optimization. Ann. Oper. Res. 175 (1), 77–105.

Rego, C., Roucairol, C., 1996. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In: Meta-Heuristics. Springer, pp. 661–675.

Skorin-Kapov, N., 2007. Routing and wavelength assignment in optical networks using bin packing based algorithms. European J. Oper. Res. 177 (2), 1167–1179.

Wei, L., Oon, W.-C., Zhu, W., Lim, A., 2011. A skyline heuristic for the 2D rectangular packing and strip packing problems. European J. Oper. Res. 215 (2), 337–346.

Wu, X., Lü, Z., Glover, F., 2020. A matheuristic for a telecommunication network design problem with traffic grooming. Omega 90, 102003.

Wu, X., Lü, Z., Guo, Q., Ye, T., 2015. Two-level iterated local search for WDM network design problem with traffic grooming. Appl. Soft Comput. 37, 715–724.

Wu, X., Yan, S., Wan, X., Lü, Z., 2016. Multi-neighborhood based iterated tabu search for routing and wavelength assignment problem. J. Comb. Optim. 32 (2), 445–468.

Yagiura, M., Ibaraki, T., Glover, F.W., 2006. A path relinking approach with ejection chains for the generalized assignment problem. European J. Oper. Res. 169 (2), 548–569.

Zang, H., Jue, J.P., Mukherjee, B., et al., 2000. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. Opt. Netw. Mag. 1 (1), 47–60.

Zhang, Q., Lü, Z., Su, Z., Li, C., Fang, Y., Ma, F., 2020. Vertex weighting-based tabu search for p-center problem. In: Bessiere, C. (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. International Joint Conferences on Artificial Intelligence Organization, pp. 1481–1487, Main track.