European Journal of Operational Research 232 (2014) 298–306

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A post-optimization method for the routing and wavelength assignment problem applied to scheduled lightpath demands

Lucile Belgacem [a], Irène Charon [b], Olivier Hudry [b,*]

[a] FuturMaster, 696, rue Yves Kermen, 92100 Boulogne-Billancourt, France
[b] Institut Télécom, Télécom ParisTech & CNRS, LTCI UMR 5141, 46, rue Barrault, 75634 Paris Cedex 13, France

ARTICLE INFO

Article history:
Received 20 May 2009
Accepted 26 June 2013
Available online 18 July 2013

Keywords:
Combinatorial optimization
WDM optical networks
Routing and Wavelength Assignment Problem (RWA)
Scheduled Lightpath Demands (SLDs)
Post-optimization

ABSTRACT

We consider here a NP-hard problem related to the Routing and Wavelength Assignment (RWA) problem in optical networks, dealing with Scheduled Lightpath Demands (SLDs). An SLD is a connection demand between two nodes of the network, during a certain time. Given a set of SLDs, we want to assign a lightpath, i.e. a routing path and a wavelength, to each SLD, so that the total number of required wavelengths is minimized. The constraints are the following: a same wavelength must be assigned all along the edges of the routing path of any SLD; at any time, a given wavelength on a given edge of the network cannot be used to satisfy more than one SLD. To solve this problem, we design a post-optimization method improving the solutions provided by a heuristic. The experimental results show that this post-optimization method is quite efficient to reduce the number of necessary wavelengths.

1. Introduction

We consider a problem related to the Routing and Wavelength Assignment (RWA) problem in wavelength division multiplexing (WDM) optical networks; see e.g. Gagnaire, Kuri, and Koubaa (2009), Kuri, Puech, Gagnaire, Dotaro, and Douville (2003), Ramaswami and Sivarajan (2002) or Zheng and Mouftah (2004) for general references. Fibre-optic networking technology using WDM offers the potential of dividing the bandwidth of a fibre into several channels, each at a different optical wavelength, permitting to carry data in parallel. For a given network topology, represented by an undirected graph $\mathcal{G}$, the RWA problem consists in establishing a set S of traffic demands, also called connection requests, in this network. Different versions of the RWA problem can be found in the literature, depending on the performance metrics and on the traffic assumptions; see for instance (Zang, Jue, & Mukherjee, 2000). Traffic demands may be of three types: static (permanent and known in advance), scheduled (requested for a given period of time and known in advance) and dynamic (unexpected). The typical objectives of RWA can be:

- to minimize the required number of wavelengths under given connection requests,
- to minimize the blocking probability, i.e. the number of rejected traffic demands, under a given number of wavelengths and dynamic connection requests,
- to minimize the maximum number of wavelengths going through a single fibre, also called the *lightpath congestion*,
- to minimize the network load as defined by the fraction of the number of wavelengths used on the overall set of fibre links in the network.

These problems or variants of them have been extensively studied in the last decades; see, among others, (Banerjee & Mukherjee, 1996; Belgacem & Puech, 2008; Chen & Banerjee, 1996; Choi, Golmie, Lapeyere, Mouveaux, & Su, 2000; Jaumard, Meyer, & Thiongane, 2006; Krishnaswamy & Sivarajan, 2001; Kumar & Kumar, 2002; Kuri, 2003; Kuri, Puech, Gagnaire, 2003; Kuri, Puech, Gagnaire, Dotaro et al., 2003; Lee, Kang, Lee, & Park, 2002; Margara & Simon, 2000; Noronha, Resende, & Ribeiro, 2008; Noronha & Ribeiro, 2006; Ramaswami & Sivarajan, 1995; Skorin-Kapov, 2006a, 2006b, 2007, 2008; Zang et al., 2000; Zheng & Mouftah, 2004).

Many of these works consider static demands; the problem is then sometimes called the *wavelength dimensioning problem*, see for instance (Jaumard et al., 2006) where this problem is studied. In this paper, we deal with the case of a set S of Scheduled Lightpath Demands (SLDs). This is relevant because of the predictable and periodic nature of the traffic load in real transport networks, more intense during working hours, see Kuri, Puech, Gagnaire, Dotaro et al. (2003). This case is also much more difficult

* Corresponding author. Tel.: +33 1 45 81 77 63; fax: +33 1 45 81 31 19.
E-mail addresses: lucile.belgacem@futurmaster.com (L. Belgacem), charon@telecom-paristech.fr (I. Charon), hudry@telecom-paristech.fr (O. Hudry).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.06.050

than the static one, because of the time constraints which do not exist for static demands.

More precisely, an SLD s belonging to S can be represented by a quadruplet $s = (x, y, \alpha, \beta)$, where x and y are some vertices of $\mathcal{G}$ (source and destination nodes of the connection request), and where α and β denote the set-up and tear-down dates of the demand. The routing of $s = (x, y, \alpha, \beta)$ consists in setting up a lightpath (P, w) between x and y, where P is a path, also called route, between x and y in $\mathcal{G}$ and w a wavelength (we do not address here the case where a traffic demand requires several lightpaths). In order to satisfy s, this lightpath must be reserved during all the span of $[\alpha, \beta]$.

When wavelength converters exist, it is not necessary to use a same wavelength on all the links used by a lightpath. Unfortunately, this entails a lot of expense and hence changes the nature of the problem: the aim is then to determine the placement of these converters so that the overall network cost is minimized; see for instance (Chu, Li, & Chlamtac, 2002). When there are no wavelength converters in the network, as it will be assumed in this paper, the *wavelength continuity constraint* is imposed: the same wavelength must be used on all the links used by a lightpath. Moreover, at any given time, a wavelength can be used at most once on a given link; in other words, if two demands overlap in time, they can be assigned the same wavelength if and only if their routing paths are disjoint in edges. This constraint is often called the *wavelength clash constraint*.

We address in this paper the problem consisting in minimizing the number W of wavelengths required to establish all the SLDs. This problem is NP-hard, even if we do not take the time-windows into account; see Chlamtac, Ganz, and Karmi (1992). A solution of this problem is defined by specifying, for each SLD, the lightpath chosen for supporting the connection, i.e. a route and a wavelength, so that there is no conflict between any two lightpaths (let us recall that two lightpaths are in conflict if they use the same wavelength, they have at least one edge in common and the corresponding demands overlap in time). Several approximate or exact methods have been proposed in the literature to deal with this NP-hard problem for static demands or for SLDs; see Gi Ahn, Lee, Chung, and Choo (2005), Jaumard et al. (2006), Kuri, Puech, Gagnaire, Dotaro et al. (2003), Lee et al. (2002), Margara and Simon (2000), Park, Yang, and Bang (2007), Saradhi and Gurusamy (2005), Skorin-Kapov (2006a, 2006b, 2007), Wauters and Demeester (1996) and Zang et al. (2000).

The greedy method proposed by in Skorin-Kapov (2006b), which has been designed initially to deal with the case where SLDs may require several lightpaths simultaneously, gives very satisfying results in a very small amount of time and is, with this respect, among the most efficient heuristics. Its application to our problem, though we do not consider here the case where a traffic demand requires several lightpaths, will be used as a benchmark for measuring the performance of our approach. Indeed we propose in this paper a post-optimization method in order to improve the results given by other heuristics. The CPU time of the overall method will naturally increase, but it will remain acceptable to deal with SLDs: since the demands are known in advance, the allotted time to provide a solution is relatively large (unlike the case where demands are unexpected, and for which routings must be computed dynamically).

The greedy algorithm derived from Skorin-Kapov (2006b) and the post-optimization method are presented in Section 2. In Section 3, we apply these methods on different types of instances and we analyse the obtained results. Finally we conclude in Section 4.

2. Resolution methods

We describe in this section the different heuristics that we apply to deal with RWA. Let us recall that the aim consists in minimizing the number W of wavelengths required to establish a given set S of demands. We first present in Section 2.1 the greedy algorithm derived from Skorin-Kapov (2006b); we propose in Section 2.3 a slight modification of this algorithm so that it can be repeated. The post-optimization method is described in Section 2.2.

2.1. The greedy algorithm

The greedy algorithm derived from Skorin-Kapov (2006b) consists in considering the wavelengths one by one, and in trying to route as many SLDs as possible with each wavelength. For each wavelength w, the SLDs are examined following some prescribed order $\prec$; we set $s' \prec s$ if s' is examined before s.

More precisely, let w be the current wavelength and $s = (x, y, \alpha, \beta)$ be the current SLD. We consider a graph $\mathcal{H}(s)$ obtained from $\mathcal{G}$ by removing all the edges unavailable for the routing of s with w, i.e. edges that are contained in lightpaths corresponding to SLDs already routed with the wavelength w and which overlap s in time. According to this construction, any edge of $\mathcal{H}(s)$ could be used to route the demand s with wavelength w without inducing any clash with previously established SLDs. Thus, if there exists at least one path between x and y in $\mathcal{H}(s)$, we attribute the shortest possible path P_s to the SLD s as well as the wavelength w; otherwise s is put aside and will be dealt with latter using another wavelength. Then we move up to the next not yet established SLD with respect to the prescribed order.

When all the SLDs have been examined, we move up to the wavelength $w + 1$ and try to route the remaining SLDs. The algorithm stops as soon as all the SLDs have been established, and returns the current value of w.

This algorithm is given in Fig. 1 and will be referred to as G.

2.2. The post-optimization method

The post-optimization method presented in this paper aims at improving the results provided by the greedy heuristic depicted above, though it can be applied to any heuristic permitting to solve the addressed problem or even to other variants of this problem. It consists in minimizing the overall values of the wavelengths of the established lightpaths in order to try to minimize the total number of wavelengths W.

The principle of this method is the following: for any $w \in \{2, \ldots, W\}$, we try to empty the set of SLDs routed with w, at least partially; this set will be called the *layer* w in the following. This is done by trying to assign a smaller wavelength $(1, 2, \ldots, w-1)$ to the demands of the layer w, which leads us to rearrange the wavelengths assigned to the SLDs of these lower

Input: A network $\mathcal{G}$; a set S of SLDs
Output: for each $s \in S$: a lightpath (P_s, w_s); $W = \max_{s \in S}(w_s)$

$w \leftarrow 0$
$N \leftarrow 0$
$\forall s \in S$, $(P_s, w_s) \leftarrow (\emptyset, 0)$
While $N < |S|$
  $w \leftarrow w + 1$
  For $s \in S$ (with $s = (x, y, \alpha, \beta)$), if s is not established $(w_s = 0)$
    $\mathcal{H}(s) \leftarrow \mathcal{G}$
    For $s' \in S$ with $s' \prec s$ (and $s' = (x', y', \alpha', \beta')$)
      If $w_{s'} = w$ and $[\alpha, \beta] \cap [\alpha', \beta'] \neq \emptyset$
        Remove the edges of $P_{s'}$ from $\mathcal{H}(s)$
    If there exists a path between x and y in $\mathcal{H}(s)$
      Assign the lightpath (P_s, w) to s where P_s denotes
        a shortest path between x and y in $\mathcal{H}(s)$
      $N \leftarrow N + 1$
$W \leftarrow w$.

Fig. 1. Greedy algorithm G.

layers. During this operation, the layers of some SLDs may change but all of them must remain in $[1, w-1]$.

More precisely, let us assume that we want to move the demand $s = (x, y, \alpha, \beta)$ from its current layer w to a lower layer ℓ ($\ell \in [1, w-1]$). It is very likely that some of the demands belonging to the layer ℓ prevent us from routing s with this wavelength. In other words, if we delete from $\mathcal{G}$ all the edges used to establish the demands of this layer which overlap s in time, we may find no path joining x and y. So we consider a graph $\mathcal{H}(s)$, initially equal to $\mathcal{G}$, and we examine one after the other the demands s' of the layer ℓ which overlap s in time. For each such s', we remove from $\mathcal{H}(s)$ the edges of the path $P_{s'}$ supporting the connection associated with s' which are still in $\mathcal{H}(s)$. If there still exists a path in $\mathcal{H}(s)$ to set up s, we move up to the next demand s' of the layer ℓ; otherwise s' is removed from the layer ℓ and put aside in a set E, and we put back the removed edges of $P_{s'}$ inside $\mathcal{H}(s)$; of course, if some edges of $P_{s'}$ had been removed previously from $\mathcal{H}(s)$ because of former clashing SLDs, they remain removed. Thus, once all demands of the layer ℓ have been examined, it becomes possible to route s using the wavelength ℓ since all the conflicting lightpaths have been (at least temporarily) removed.

We must now deal with the demands of E: we try to place each of these demands in one of the layers $1, \ldots, w-1$, without modifying the routing of any other SLD. If a layer can be found for each demand of E, then we have finished with the demand s: s remains in the layer ℓ and the demands of E remain in their new layers, with a lightpath compatible with the ones of the other SLDs of this layer, and we move up to the following demand of the layer w. Otherwise we consider that the attempt to move s to the layer ℓ has failed: layer ℓ is restored as before the attempt to insert s inside, and we try to move s to the next layer $\ell+1$. If all the layers from 1 to $w-1$ have been examined in vain, s remains inside its current layer w, and we move up to the following demand of the layer w.

As all the demands of the current layer w are handled, the number of remaining demands on this layer may decrease, or the layer w may even become totally empty. In the later case, we shift the layers $w+1, w+2, \ldots, W$ to the layers $w, w+1, \ldots, W-1$, and we have succeeded in saving one wavelength definitively: W becomes $W-1$.

This method, summarized in Fig. 2, is referred to as the post-optimization algorithm. Let us notice that even if a rearrangement of the layers does not permit to decrease the number W of required wavelengths, it may happen that further applications of the algorithm succeed to do so, because the SLDs are not dispatched in the layers in the same manner from one application to another. In the experiments presented below, we chose to repeat the post-optimization algorithm until four consecutive runs do not decrease W. This choice is based on an experimental observation and arises from a compromise between CPU time and the quality of the computed solutions. The overall heuristic consisting of the greedy algorithm followed by the application of the post-optimization method will be denoted G+ in the rest of the paper.

Input: A network $\mathcal{G}$; a set S of SLDs;
for each $s \in S$: a lightpath (P_s, w_s); $W = \max_{s \in S}(w_s)$
Output: for each $s \in S$: a lightpath $(\tilde{P}_s, \tilde{w}_s)$; $\tilde{W} = \max_{s \in S}(\tilde{w}_s)$

```
w ← 2
While w ≤ W
  For s ∈ S (with s = (x, y, α, β)) with w_s = w
(1)   For ℓ from 1 to w − 1
        E ← ∅
        H(s) ← G
        For s' ∈ S (with s' = (x', y', α', β')) with w_s' = ℓ
          If [α, β] ∩ [α', β'] ≠ ∅
            Delete the edges of P_s' ∩ H(s) from H(s)
            If there exists no path between x and y in H(s)
              Remove s' from layer ℓ and put s' in E
              Restore the removed edges of P_s' in H(s)
        Compute a shortest path P̃_s between x and y in H(s)
        (P_s, w_s) ← (P̃_s, ℓ)
(2)     For s' ∈ E (with s' = (x', y', α', β'))
          For λ from 1 to w − 1
            H(s') ← G
            For s'' ∈ S (with s'' = (x'', y'', α'', β'')) with w_s'' = λ
              If [α', β'] ∩ [α'', β''] ≠ ∅
                Delete the edges of P_s'' ∩ H(s') from H(s')
            If there exists a shortest path P̃_s' between x' and y' in H(s')
              (P_s', w_s') ← (P̃_s', λ)
              Go to (2) and move up to the next demand s'
          If s' has not been established
            Put back all the demands of E in layer ℓ with their initial lightpaths
            Restore the initial lightpath for s with w as its wavelength
            Go to (1) and move up to the next layer (ℓ + 1)
  If the layer w has been completely emptied
    For all s ∈ S: if w_s > w, w_s ← w_s − 1
    W ← W − 1
  Else w ← w + 1
W̃ ← W
For each s ∈ S, (P̃_s, w̃_s) ← (P_s, w_s).
```

Fig. 2. Post-optimization algorithm.

2.3. Repetition of the greedy algorithm

The introduction of the post-optimization method yields a significant increase in computation time. To evaluate the post-optimization method, we will compare the results provided by the greedy heuristic with or without this post-optimization method. In order to avoid any bias, it is desirable that both methods are given the same amount of CPU time to provide a solution. Thus we propose a slight modification of G in order to make it stochastic (G is deterministic otherwise); then we will be able to compare G+ with a multi-start procedure consisting in repeating G as many times as necessary to attain the same CPU time as the one required by G+.

The principle is straightforward: we propose to consider, for each run of G, a random order for the examination of the SLDs. In Skorin-Kapov (2006b), the demands are ordered with respect to the decreasing numbers of connection requests, which is irrelevant in our context since all the SLDs are assumed to require the establishment of only one lightpath. We have performed some experiments to analyse the effect of the order of the SLDs on the performance of G. Whatever the considered order (increasing or decreasing order of the time window's width, increasing or decreasing order of the length of a shortest path between the source–destination nodes), the obtained results are not significantly different. The best way to take benefit from the allotted CPU time seems then to generate a random order of the SLDs for each run of G. Of course, the solution returned by this repetition of G will be the best one computed over the different runs of G during this repetition. This repetition of G will be denoted *RG*.

3. Experiments

In the next three subsections, we report some results that we obtained in our experiments. First, we present in Section 3.1 the detailed results obtained for three graphs. Then, in Section 3.2, we globally give the results obtained for other instances, involving different types of graphs. Thanks to the use of an analytical lower bound, we could find instances for which we know the optimal value for the necessary number of wavelengths; we study the application of our three methods to such instances in Section 3.3.

The three graphs studied in details in Section 3.1 are:

- $\mathcal{G}57$, with 57 vertices and 85 edges, extracted from the European optical transport network; see Fig. 3;
- $\mathcal{G}29$, with 29 vertices and 44 edges, representing a hypothetical North-American backbone network; see Fig. 4;
- $\mathcal{G}200$, with 200 vertices and 239 edges, simulating a large optical network.

The first two graphs are often used to illustrate RWA problems. We have added the graph $\mathcal{G}200$ in order to observe the behaviour of the methods when applied to larger networks. This graph has relatively few edges so that the number of paths that may support a given demand is limited (otherwise the number of required wavelengths is very small).

To enlarge the field of results, we study in Section 3.2 other sets of instances involving three types of graphs, called *Y*, *Z* and *W*, following the classification used in Noronha et al. (2008). Note that the connection requests considered in Noronha et al. (2008) are static ones: there is no time-window defined for such a request. Graphs of type *Y* are randomly generated, graphs of type *Z* are defined on torus, graphs of type *W* are real instances coming from the literature. More precisely, they are defined and generated as in Noronha et al. (2008):

- To define a graph of type *Y*, a number of nodes n and a probability p, i.e. a real number between 0 and 1, are assumed to be given. First, to construct a graph of type *Y* with these parameters, we connect each pair of nodes with probability p. Then, for each node x of degree less than 2, we randomly add edges incident to x in order to have x of degree 2. Finally, we restrict the diameter of the graph to at most 7, by adding edges between vertices distant of more than 7.
- The graphs of type *Z* are built on $a \times b$ grids embedded on the torus, where a and b are given integers; each node is connected only to its nearest four neighbours. So, the graph is entirely determined by its dimensions a and b; its number of vertices is equal to $a \times b$ and its number of edges is equal to $2 \times a \times b$; the degree of any vertex is equal to 4.
- As said above, the graphs of type *W* are real instances. It is already the case for the graphs $\mathcal{G}29$ and $\mathcal{G}57$ described above. We add two graphs downloaded from http://sndlib.zib.de/home.action; one, with 50 vertices and named here $\mathcal{G}50$, corresponds to an optical network in Germany; the other, with 65 vertices and named here $\mathcal{G}65$, corresponds to an optical graph in Austria.

To find instances for which we know the optimal value, as the instances of Section 3.3, we use an analytical lower bound that we define now; see Skorin-Kapov (2006b). For each vertex x of the considered graph $\mathcal{G}$, we first determine the list $L(x)$ of the SLDs involving x as their origin or destination. Then, we compute the maximum number $M(x)$ of SLDs belonging to $L(x)$ which must be routed at a same instant: the overall intersection of the time-windows of these $M(x)$ SLDs is not empty. Let $b(x)$ be the ratio of $M(x)$ divided by the degree $d(x)$ of x in $\mathcal{G}$. The number of wavelengths necessary to satisfy all the demands belonging to $L(x)$ is obviously at least equal to $b(x)$. Then the lower bound B is defined as the maximum of the ceiling of the ratios $b(x)$ over the vertices x of $\mathcal{G}$:

$$B = \max_x \left\lceil \frac{M(x)}{d(x)} \right\rceil.$$

For the instances depicted above, B is always much less than the value obtained by our methods, including when these methods are given much larger CPU times, generally less than the half. Probably, the optimum is also far from B. But, by choosing graphs of type *Y* with larger numbers of edges than before, we succeeded to find several instances with a solution reaching the lower bound: increasing the number of edges obviously increases the degrees and so B takes lower values; but the number of paths increases still faster and so the required number of wavelengths decreases; for the instances considered here, the required number of wavelengths collapses to B and thus we know that B is the optimum value. Section 3.3 reports the results for seven such instances.

The sets of SLDs are generated randomly so that the number of time-overlaps is significant but not too large (the connection requests in Noronha et al. (2008) or in Skorin-Kapov (2007) are also randomly generated but, since in these papers only static demands are considered, there are no time-windows): if they are too few, there are few clashes between the demands and therefore the addressed problem becomes too easy; on the contrary, if the time-overlaps are too numerous, the number of required wavelengths increases greatly and the problem becomes again less interesting. This can be achieved by drawing the source and destination nodes

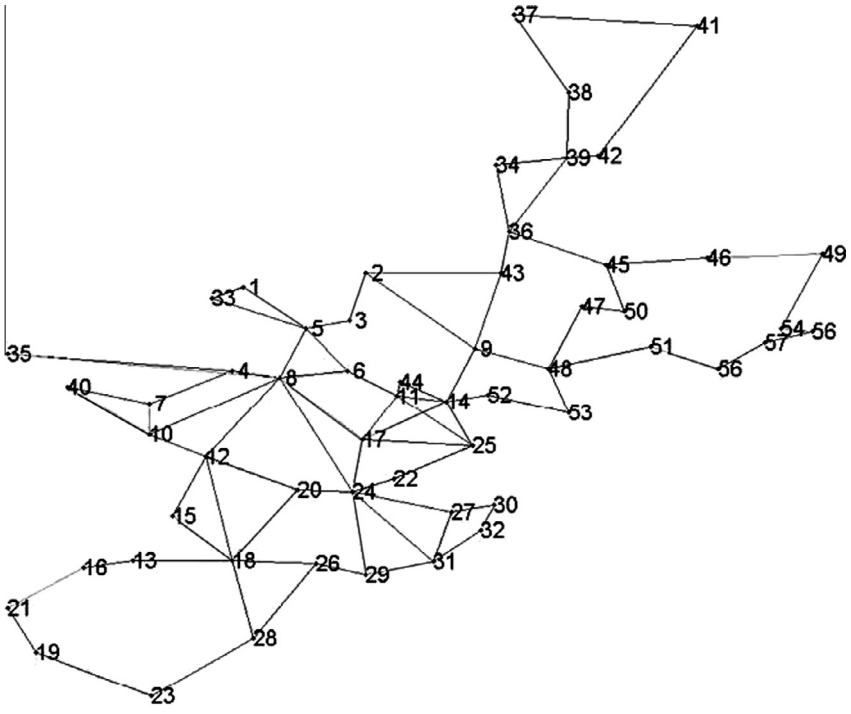

Fig. 3. The graph $\mathcal{G}57$.

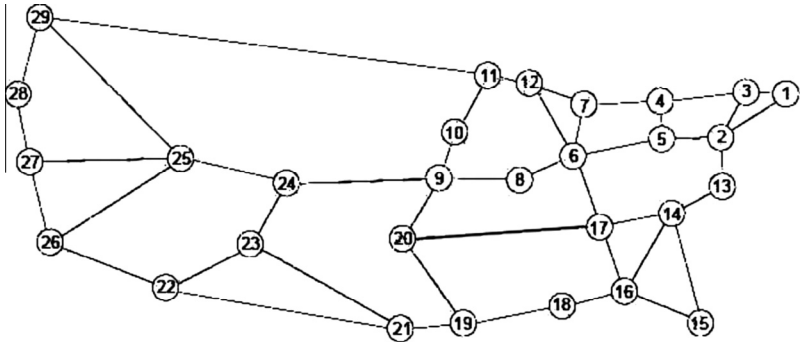

Fig. 4. The graph $\mathcal{G}29$.

of each SLD s randomly with a uniform distribution of probabilities over all vertices of the considered graph (of course the two nodes must be different). The set-up and tear-down times of s are chosen in $[0,1000]$; the centre c of the interval is a real number drawn uniformly in $[L,1000-L]$, with L chosen here between 200 and 300. The time-window of s is then of the form $[c-L\times r^{\gamma}, c+L\times r^{\gamma}]$, where r is a real number drawn uniformly between 0 and 1, while γ takes the values 2, 3 or 4. By choosing different values for these parameters, we obtain time-windows with more or less pairwise time-overlaps. To be more specific, we give in the next section the proportions of pairwise time-overlaps of the instances detailed in Section 3.1. Anyway, do not forget that a clash between two SLDs comes from both an overlap of their time-windows and from the paths chosen to route the two SLDs. As these ones are not fixed, the number of clashes cannot be fixed in advance.

The experiments reported below have been performed on Solaris Sun stations, namely Sun Ultra 20M2 AMD bicore 3 gigahertz. In order to evaluate the three heuristics G, RG and G+ when applied to our instances, we carry out 100 runs of each method for each instance.

3.1. Detailed results for nine instances involving $\mathcal{G}57, \mathcal{G}29$ and $\mathcal{G}200$

We apply the process depicted above to generate three sets of demands for each one of the three graphs $\mathcal{G}57, \mathcal{G}29$ and $\mathcal{G}200$. These sets contain respectively 500, 1000 and 3000 SLDs. Thus we obtain nine instances. The name of each one is obtained by the concatenation of the name of the graph with the number of SLDs to route; for instance, $\mathcal{G}57$–500 denotes the instance for which the graph is $\mathcal{G}57$, with 500 SLDs. The proportions of pairwise time-overlaps are the following: 66% for $\mathcal{G}57$–500, 49% for $\mathcal{G}57$–1000, 23% for $\mathcal{G}57$–3000; 65% for $\mathcal{G}29$–500, 53% for $\mathcal{G}29$–1000, 24% for $\mathcal{G}29$–3000; 40% for $\mathcal{G}200$–500, 65% for $\mathcal{G}200$–1000, 37% for $\mathcal{G}200$–3000.

In the following, we present the results obtained when applying G, RG and G+ to these nine instances. These results are given in Tables 1. For each entry of the first three lines, we specify the best, the average and the worst values of the computed numbers of wavelengths necessary to route the SLDs over the 100 runs, as well as the average CPU time in seconds; the last two lines of each table specify the ratios $(W_G - W_{G+})/W_G$ and $(W_{RG} - W_{G+})/W_{RG}$, where W_G, W_{RG} and W_{G+} denote the average numbers of wavelengths required to set up all the connections when applying 100 runs of G, RG and G+ respectively.

Let us recall that the method RG consists in repeating the greedy algorithm G as many times as required to attain the same computation time as G+. For example, for the instance $\mathcal{G}57$–500, G runs in 0.02761 second whereas G+ takes 2.68 second. Therefore RG is obtained by repeating G 97 times in average, which indeed corresponds to an overall computation time of 2.68 second. As said above, the result provided by RG is of course the smallest number of wavelengths obtained during the repetition.

For the nine instances considered here, the gain, more precisely the ratio $(W_G - W_{G+})/W_G$, yielded by the application of G+ with respect to the sole application of the greedy heuristic G exceeds generally 10% (the average gain is equal to 11%), and it reaches nearly 16% for the instance $\mathcal{G}200$–3000. Even when considering the same CPU time, the gain of G+ with respect to RG, measured similarly by $(W_{RG} - W_{G+})/W_{RG}$, remains significant: 7.4% in average, with a maximum of 13.2% for $\mathcal{G}200$–3000.

We give in Figs. 5–7 the distributions of the numbers of required wavelengths obtained over the 100 runs of each method. The x-axis represents the number of required wavelengths W, and the y-axis represents the number of times that each value has been observed.

We see that the CPU time required by G+ is significantly larger than the one required by G. In our experiments, the CPU time required to perform G+ can reach few minutes, whereas it is nearly instantaneous for G (less than one second). From a practical point of view, these computation times remain quite acceptable, especially considering the high complexity of the problem and the large sizes of the instances, since the addressed problem concerns connection requests that are known in advance. Indeed, in this case,

Table 1
Best, average and worst numbers of required wavelengths and average CPU times for $\mathcal{G}57, \mathcal{G}29$ and $\mathcal{G}200$.

$\mathcal{G}57$	$\mathcal{G}57$-500	$\mathcal{G}57$-1000	$\mathcal{G}57$-3000
G	43–44.89–47	65–67.38–70	90–91.53–95
	0.028 second	0.083 second	0.307 second
RG	42–42.95–44	64–65.24–66	88–88.82–89
	2.68 second	13.54 second	74.54 second
G+	40–40.96–42	60–61.30–63	83–84.07–86
	2.68 second	13.39 second	74.43 second
$(W_G - W_{G+})/W_G$	8.75%	9.02%	8.15%
$(W_{RG} - W_{G+})/W_{RG}$	4.63%	6.04%	5.35%
$\mathcal{G}29$	$\mathcal{G}29$-500	$\mathcal{G}29$-1000	$\mathcal{G}29$-3000
G	42–43.58–46	63–65.18–68	76–78.61–81
	0.018 second	0.056 second	0.198 second
RG	40–41.44–42	62–62.84–63	75–76.08–77
	2.16 second	7.65 second	49.85 second
G+	38–38.75–40	60–60.44–62	69–70.59–72
	2.15 second	7.63 second	49.71 second
$(W_G - W_{G+})/W_G$	11.08%	7.27%	10.2%
$(W_{RG} - W_{G+})/W_{RG}$	6.49%	3.82%	7.22%
$\mathcal{G}200$	$\mathcal{G}200$-500	$\mathcal{G}200$-1000	$\mathcal{G}200$-3000
G	18–19.68–22	49–50.35–52	78–80.19–82
	0.031 second	0.170 second	0.887 second
RG	18–18.55–19	47–48.40–49	77–77.89–78
	1.68 second	26.76 second	292.30 second
G+	16–16.84–18	41–43.14–45	66–67.59–72
	1.67 second	26.62 second	292.15 second
$(W_G - W_{G+})/W_G$	14.43%	14.34%	15.71%
$(W_{RG} - W_{G+})/W_{RG}$	9.22%	10.87%	13.22%

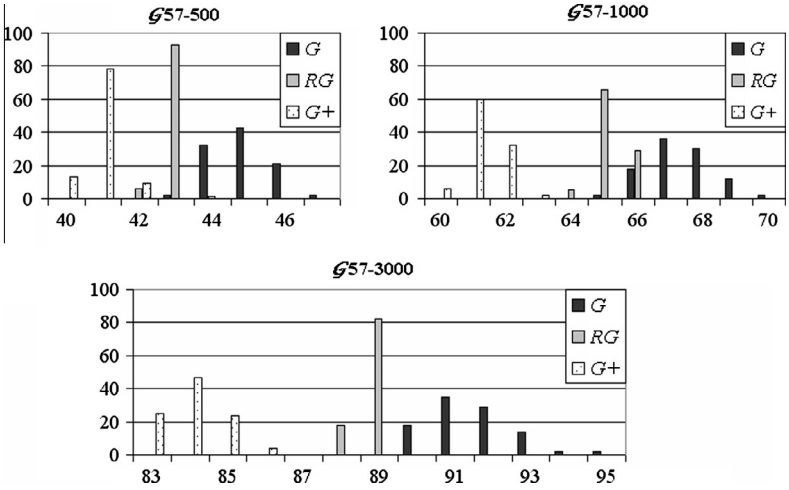

Fig. 5. Distributions of W for $\mathcal{G}57$; the x-axis represents the number of required wavelengths W, and the y-axis represents the number of times that each value has been observed during the 100 runs.

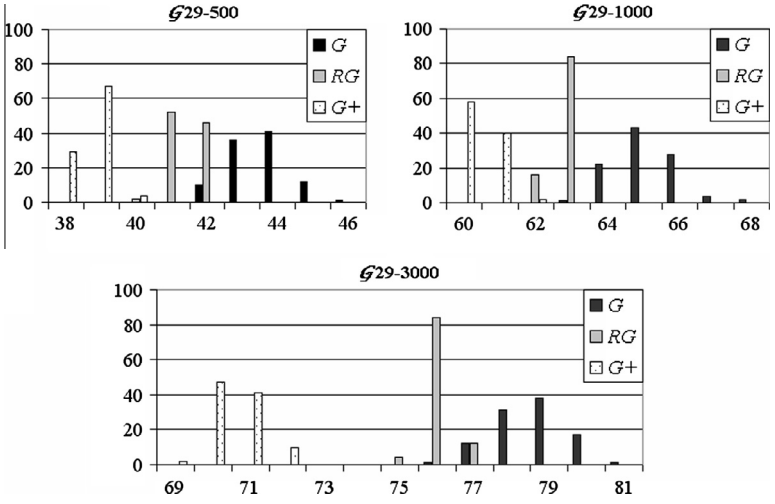

Fig. 6. Distributions of W for $\mathcal{G}29$; the x-axis represents the number of required wavelengths W, and the y-axis represents the number of times that each value has been observed during the 100 runs.

a telecommunications operator can easily afford to spend the time required by the application of the post-optimization method in order to save some wavelengths, that will be available to establish further connection requests, for instance unexpected demands.

However, in order to reduce the required time for the application of G+, we may modify the way to deal with the layers during the post-optimization method: instead of examining all the layers in order to empty them as much as possible, we propose to deal with only the layers corresponding to large values of wavelengths. These layers are indeed more likely to yield a gain in the total number of required wavelengths. More precisely, we introduce a parameter i varying from 0 to W; then we examine only the i layers of values $W - i + 1, \ldots, W - 1, W$.

Figs. 8 and 9 give respectively the average number of required wavelengths and the CPU time in seconds with respect to the value of the parameter i for the instance $\mathcal{G}57$–500 (results turned out to be similar for the other instances). The label N (none) on the x-axis corresponds to the result given by G, whereas the label A (all) corresponds to G+ when all the layers have been rearranged ($i = W$).

We observe that the CPU time varies almost linearly with respect to the number of rearranged layers i and that the quality of the solution increases when the post-optimization method is applied to more layers. However, this gain is larger for the high layers than for the low layers (the slope of the graph decreases when i increases), as expected. For the considered instance, we see that we could have dealt with only one third, or even one quarter, of the layers, reducing thus the CPU time in the same proportion, without loosing much in terms of quality of the provided solutions. Thus we can choose the number of rearranged layers with respect to the available time. Anyway, in order not to add an extra parameter, we do not adopt this possibility in the following.

3.2. Global results for twelve sets of one hundred instances each

We report now the results concerning 12 sets of 100 instances: 4 sets, called $\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3$ and $\mathcal{Y}4$, contain graphs of type Y; 3 sets, called $\mathcal{Z}1, \mathcal{Z}2$ and $\mathcal{Z}3$, contain graphs of type Z; the other sets contain graphs of type W. For each instance of each set, we apply the methods G and G+, repeating them in order to reach a chosen duration; in the following, RG still denotes the repetition of G. Of course, the result provided at the end is the best value found during the repetition. In these experiments, because of the gap between the CPU times of one application of G and one application of G+

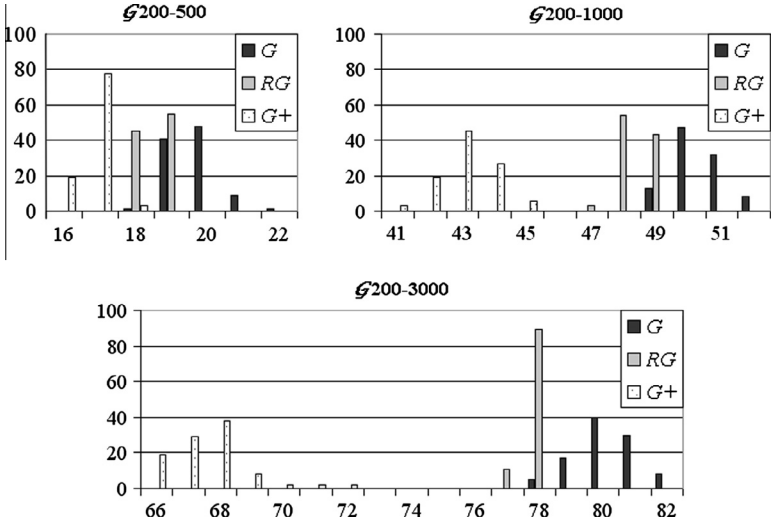

Fig. 7. Distributions of W for $\mathcal{G}200$; the x-axis represents the number of required wavelengths W, and the y-axis represents the number of times that each value has been observed during the 100 runs.

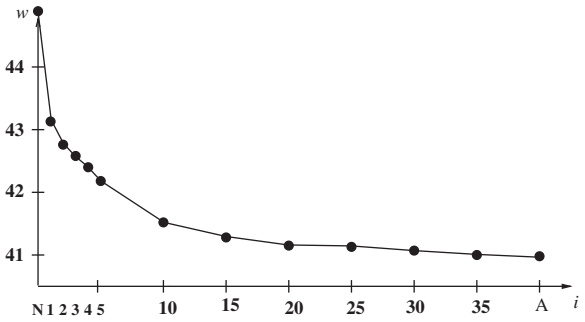

Fig. 8. Number of wavelengths according to the number of rearranged layers for $\mathcal{G}57$-500.

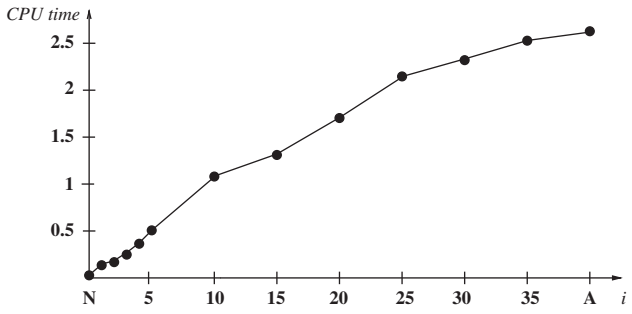

Fig. 9. CPU time, in seconds, according to the number of rearranged layers for $\mathcal{G}57$-500.

(see Section 3.1), G is repeated a large number of times and the method $G+$ a small number. In the following, ρ_{RG} and ρ_{G+} will respectively denote the averages of the number of repetitions of RG and of $G+$ during the given CPU time.

Remember that any instance of RWA is made of a graph and a set of SLDs. We first explain how the graph of each instance is generated (see above for the specification of the types Y, Z and W).

- For each instance of $\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3$ and $\mathcal{Y}4$, we randomly choose a graph of type Y with the following parameters (note that the probabilities p applied here are similar to the ones in Noronha et al. (2008), which range between 3% and 5%):
  - for $\mathcal{Y}1$: 150 vertices and a probability equal to 2%;
  - for $\mathcal{Y}2$: 100 vertices and a probability equal to 3%;
  - for $\mathcal{Y}3$: 70 vertices and a probability equal to 5%;
  - for $\mathcal{Y}4$: 40 vertices and a probability equal to 7%.
- For $\mathcal{Z}1, \mathcal{Z}2$ and $\mathcal{Z}3$, the considered graphs are of type Z (they are grids embedded on a torus). We choose the dimensions of the grids so that the number of vertices is about 100, as in Noronha et al. (2008). More precisely:
  - for the instances of the first set $\mathcal{Z}1$, the graph is the grid 10×10;
  - for the instances of the second set $\mathcal{Z}2$, the graph is the grid 17×6;
  - for the instances of the third set $\mathcal{Z}3$, the graph is the grid 25×4.

So the instances belonging to a given set $\mathcal{Z}1, \mathcal{Z}2$ or $\mathcal{Z}3$ all share the same graph, but of course the set of SLDs will not be the same.

- The last five sets involve graphs of type W. For the first of these five sets, the graph is $\mathcal{G}29$; $\mathcal{G}57$ is the graph associated with the following two sets; the graph of the fourth set is $\mathcal{G}50$; while $\mathcal{G}65$ is the graph of the last set; see their characteristics above. For simplicity, we call $\mathcal{G}29, \mathcal{G}57, \mathcal{G}50$ and $\mathcal{G}65$ the sets of instances involving respectively the graphs $\mathcal{G}29, \mathcal{G}57, \mathcal{G}50$ and $\mathcal{G}65$.

Then for each set of instances, we randomly generate 100 sets of SLDs. For each instance, each SLD of each set of SLDs is randomly generated thanks to the process described above, at the beginning of Section 3. So, the 100 instances of any set of instances are all different. The number of SLDs depends on the studied set. The numbers of SLDs are the following: 1500 SLDs for the sets $\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3$ or $\mathcal{Y}4$; 2000 SLDs for the sets $\mathcal{Z}1, \mathcal{Z}2$ or $\mathcal{Z}3$; 2000 SLDs for the set $\mathcal{G}29$; 3000 SLDs for the first set $\mathcal{G}57$; 1000 SLDs for the second set $\mathcal{G}57$; 1500 SLDs for the two sets $\mathcal{G}50$ and $\mathcal{G}65$.

For each set, we also choose a CPU time (see below). Note that the CPU time allocated to the repetition of each method can be very large. For example, the CPU time devoted to each instance of the first set $\mathcal{G}57$ is 30 minute. It means that it required 100 hour of CPU time to deal with the application of both methods to the 100 instances of this set. Because of that, it is impossible to try long CPU times for each instance. Moreover, some other experiments that we performed with other CPU times show that the qualitative conclusions reported below remain the same for these other CPU times. So, instead of reporting all these experiments, we choose to report some of them, with different CPU times ranging from

Table 2
Average results for 12 series of 100 instances.

Set	n	m	nb_SLD	$time$ (s)	ρ_{RG}	ρ_{G+}	W_{RG}	W_{G+}	$(W_{RG} - W_{G+})/W_{RG}$ (%)
$\mathcal{Y}1$	150	234.92	1500	60	501.09	1.37	30.22	26.64	11.85
$\mathcal{Y}2$	100	152.88	1500	60	1008.15	4.25	23.59	20.90	11.40
$\mathcal{Y}3$	70	130.83	1500	60	1032.45	4.70	24.42	22.25	8.89
$\mathcal{Y}4$	40	57.81	1500	60	1595.17	3.66	43.90	38.35	12.64
$\mathcal{Z}1$	100	200	2000	60	464.22	1.01	31.71	28.43	10.34
$\mathcal{Z}2$	102	204	2000	60	403.90	1.06	38.97	36.62	6.03
$\mathcal{Z}3$	100	200	2000	300	1837.89	2.96	54.92	50.71	7.67
$\mathcal{G}29$	29	44	2000	120	2565.99	4.49	59.49	53.34	10.34
$\mathcal{G}57$	57	85	3000	1800	10310.42	13.47	93.31	85.39	8.49
$\mathcal{G}57$	57	85	1000	900	31052.61	106.61	44.00	40.23	8.57
$\mathcal{G}50$	50	88	1500	60	1209.27	3.55	34.30	30.57	10.87
$\mathcal{G}65$	65	108	1500	120	2071.78	6.76	34.76	31.85	8.37

Table 3
Results for seven instances with known optimum.

n	m	$nb\text{-}SLD$	ρ_{RG}	ρ_{G+}	W_{min}	RG				$G+$		W_{RG}	W_{G+}	$\frac{W_{RG}-W_{G+}}{W_{RG}}$ (%)
						nb_0	nb_1	nb_2	nb_3	nb_0	nb_1			
30	85	1500	1275.56	8.63	22	0	29	71	0	100	0	23.71	22	7.21
30	82	2000	750.82	3.50	28	0	0	80	20	53	47	30.20	28.47	5.73
40	214	2000	620.12	8.97	13	5	95	0	0	84	16	13.95	13.16	5.66
40	139	2000	643.34	5.41	19	0	100	0	0	81	19	20	19.19	4.05
40	119	2000	655.39	3.94	22	0	86	14	0	99	1	23.14	22.01	4.88
50	142	2500	374.73	1.83	24	0	0	79	21	30	70	26.21	24.70	5.76
50	155	2000	583.02	4.47	18	0	100	0	0	99	1	19	18.01	5.21

60 second to 1800 second according to the considered instance. We prefer to specify the results for different CPU times because it brings diversity and thus a more complete lighting on the behaviour of our method. More precisely, the CPU times are the following for each instance of each set: 60 second for $\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3$ or $\mathcal{Y}4$; 60 second for $\mathcal{Z}1$ or $\mathcal{Z}2$; 300 second for $\mathcal{Z}3$; 120 second for $\mathcal{G}29$; 1800 second for each instance of the first set $\mathcal{G}57$; 900 second for each instance of the second set $\mathcal{G}57$; 60 second for $\mathcal{G}50$; 120 second for $\mathcal{G}65$.

In Table 2, we successively specify for each set of instances: the set of instances, what specifies also the type of the graph involved in these instances; the number n of vertices of the graphs, according to the set; the number m of edges or the average of the number of edges of these graphs; the number nb_SLD of SLDs to route; the given CPU time, in seconds; the averages ρ_{RG} and ρ_{G+} of the number of repetitions of G and $G+$ during the given CPU time; the averages of the computed numbers of wavelengths W_{RG} and W_{G+}; the average of the relative gain $(W_{RG} - W_{G+})/W_{RG}$.

These results confirm the ones of Section 3.1. The gains provided by the post-optimization method with respect to RG range here between 6% and almost 13%, with an average of about 10%. For the instances involving $\mathcal{G}57$ as their graphs and with 1000 or 3000 SLDs, we may compare the results obtained when the CPU time is large with the results reported in Section 3.1. In Section 3.1, one application of $G+$ provided an average gain of 6.04% for $\mathcal{G}57$-1000 and of 5.35% for $\mathcal{G}57$-3000 with respect to RG. For $\mathcal{G}57$-1000, RG was repeated 163 times in average; for $\mathcal{G}57$-3000, it was repeated 243 times in average. It is sometimes considered that multi-start procedures as RG have a slow convergence and that it is necessary to run them a long time. We observe here that, even with larger numbers of repetitions, namely more than 31,000 times for 1000 SLDs and more than 10,000 times for 3000 SLDs, RG does not provide better results than $G+$ within a same CPU time. In fact, the gain of $G+$ with respect to RG is even larger when the CPU time increases: it becomes equal to 8.57% in average for 1000 SLDs instead of 6.04% and to 8.49% for 3000 SLDs instead of 5.35%. We may explain this phenomenon by the fact that, within the same CPU time (15 minute for 1000 SLDs and 30 minute for 3000 SLDs), the repetition of $G+$ ($G+$ is repeated about 107 times for 1000 SLDs and about 13 times for 3000 SLDs) is more beneficial than the one of RG. With this respect, we may consider that the convergence of RG is reached in our experiments. The results of Table 2 show that repeating $G+$ may improve the results provided by $G+$, what is not surprising, if greater CPU times are available.

3.3. Global results for seven instances with known optimum

As explained above, the use of the bound B defined at the beginning of Section 3 allowed us to find instances for which we know the optimal value. It is the case for the seven instances of which we report the results now.

These graphs are of type Y. Two have 30 vertices, three have 40 vertices and two have 50 vertices. For each instance, we perform 100 trials. For each trial, we give a CPU time equal to 60 second. In Table 3, we give for each instance:

- the number n of nodes;
- the number m of edges of the graph;
- the number $nb\text{-}SLD$ of SLDs to route;
- the number of repetititions ρ_{RG} of RG and ρ_{G+} of $G+$ during the 60 second;
- the minimum number W_{min} of necessary wavelengths;
- nb_i denotes, for $0 \leqslant i \leqslant 3$, the number of times that the best solution provided by the considered method is equal to $W_{min} + i$ (for $G+$, the values nb_i which are not specified are egal to zero);
- the following two columns give the average of the numbers W_{RG} and W_{G+} of wavelengths computed by RG and by $G+$ over the 100 trials;
- the last column gives the relative gain of $G+$ with respect to RG.

In a sense, these instances may seem easier than the previous ones. But, even for these easier instances, the post-optimization provides an improvement with respect to RG. The probability to reach the optimum is much larger for $G+$ than for RG: $G+$ found the optimum in 78% of the 700 trials summarized in Table 3, versus only 0.7% for RG.

3.4. Further comments

We may observe that the relative gain provided by *G+* with respect to *RG* is always positive in our experiments. More precisely, if we consider only the results reported above, the gain, with the same CPU time, is between about 4% and more than 13%. Another important asset of *G+* is illustrated by the histograms of Figs. 5–7 and by Table 3: *G+* succeeds in finding some values of *W* that neither *G* nor *RG* can reach during the 100 trials. When the number of SLDs increase, the gap between the histograms of *G+* in the one hand and those of *G* and *RG* on the other hand becomes larger.

Moreover for heavy loads of traffic demands (1000 or 3000 SLDs according to the considered network), the histograms for *G+* become apart completely from the ones of *G* and *RG*: the worst solution provided by *G+* remains better than the best solution found by *G* or *RG*. The same can be observed for several instances of Section 3.3 and many instances of Section 3.2.

4. Conclusion

In this study, we considered a Routing and Wavelength Assignment (RWA) problem in wavelength division multiplexing (WDM) optical networks. More precisely, we considered the problem consisting in minimizing the number of wavelengths required to establish all the Scheduled Lightpath Demands (SLDs). In this aim, we designed a post-optimization method in order to improve a greedy heuristic which is already very efficient.

According to our experimental results, the post-optimization method appears as improving significantly the results given by this greedy heuristic, in a reasonable CPU time, while it is known in combinatorial optimization that reducing the gap between the computed solutions and the optimal ones becomes more and more difficult when going closer to the optimum. Moreover, this method can be applied to any other heuristics to deal with the routing and the wavelength assignment of SLDs, and even to other problems related to RWA in optical transport networks. It will be the topic of our next studies.

References

Banerjee, D., & Mukherjee, B. (1996). A practical approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Networks. *IEEE Journal on Selected Areas in Communications, 14*(5), 903–908.

Belgacem, L., & Puech, N. (2008). Solving large size instances of the RWA problem using graph partitioning. In *Proceedings of the 12th conference on optical network design and modelling* (pp. 1–6). Vilanova i la Geltr, Spain.

Chen, C., & Banerjee, S. (1996). A new model for optimal routing and wavelength assignment in wavelength division multiplexed optical networks. In *Proceedings IEEE INFOCOM'96* (Vol. 1, pp. 164–171).

Chlamtac, I., Ganz, A., & Karmi, G. (1992). Lightpath communications: an approach to high-bandwidth optical WANs. *IEEE Transactions on Communications, 40*, 1171–1182.

Choi, J. S., Golmie, N., Lapeyere, F., Mouveaux, F., & Su, D. (2000). A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case. In *Proc. VII Int. Conf. on optical communication and networks*. Nagoya, Japan, January 2000, doi: 10.1.1.96.9004.

Chu, X., Li, B., & Chlamtac, I. (2002). On the wavelength converter placement for different RWA algorithms in wavelength-routed all-optical networks. In N. Ghani, & K. M. Sivalingam (Eds.), *Proc. SPIE, OptiComm 2002: Optical networking and communications* (Vol. 4874, pp. 186–197).

Gagnaire, M., Kuri, J., & Koubaa, M. (2009). *From network planning to traffic engineering in translucent optical WDM networks*. Springer.

Gi Ahn, H., Lee, T.-J., Chung, M. Y., Choo, H. (2005). RWA on scheduled lightpath demands in WDM optical transport networks with time disjoint paths. In *Proc. international conference on information networking* (pp. 342–351).

Jaumard, B., Meyer, C., & Thiongane, B. (2006). ILP formulations for the routing and wavelength assignment problem: Symmetric systems. In M. G. C. Resende & P. M. Pardalos (Eds.), *Handbook of optimization in telecommunications* (pp. 637–677). Springer.

Krishnaswamy, R. M., & Sivarajan, K. N. (2001). Algorithms for routing and wavelength assignment based on solutions of LP-relaxation. *IEEE Communications Letters, 5*(10), 435–437.

Kumar, M. S., & Kumar, P. S. (2002). Static lightpath establishment in WDM networks – New ILP formulations and heuristic algorithms. *Computer Communications, 25*, 109–114.

Kuri, J. (2003). Optimization problems in WDM optical transport networks with scheduled lightpath demands. PhD thesis. ENST, Paris.

Kuri, J., Puech, N., & Gagnaire, M. (2003). Diverse routing of scheduled lightpath demands in an optical transport network. In *Proc. Design of Reliable Communications Networks (DCRN2003)* (pp. 69–76).

Kuri, J., Puech, N., Gagnaire, M., Dotaro, E., & Douville, R. (2003). Routing and wavelength assignment of scheduled lightpath demands. *IEEE Journal on Selected Areas in Communications, 21*(8), 1231–1240.

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 Journal, 24*(2), 131–141.

Margara, L., & Simon, J. (2000). Wavelength assignment problem on all-optical networks with k fibres per link. In *27th international colloquium automata, languages and programming, ICALP 2000*. *LNCS* (Vol. 1853, pp. 768–779). Springer.

Noronha, T. F., Resende, M. G. C., & Ribeiro, C. C. (2008). *Efficient implementations of routing and wavelength assignment heuristics*. *LNCS* (Vol. 5038). Springer.

Noronha, T. F., & Ribeiro, C. C. (2006). Routing and wavelength assignment by partition coloring. *European Journal of Operational Reseach, 171*(3), 797–810.

Park, S., Yang, J. S., & Bang, Y.-C. (2007). On RWA algorithms for scheduled lightpath demands. *International Journal of Computer Science and Network Security, 7*(3), 144–150.

Ramaswami, R., & Sivarajan, K. N. (1995). Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking, 3*(5), 489–500.

Ramaswami, R., & Sivarajan, K. N. (2002). *Optical networks – A practical perspective* (2nd ed.). Morgan Kaufmann.

Saradhi, C.V., & Gurusamy, M. (2005). Graph theoretic approaches for routing and wavelength assignment of scheduled lightpath demands in WDM optical networks. In *Proc. 2nd international conference on broadband networks* (Vol. 2).

Skorin-Kapov, N. (2006). Heuristic algorithms for virtual topology design and routing and wavelength assignment in WDM networks. PhD thesis, University of Zagreb, Croatia.

Skorin-Kapov, N. (2006b). Heuristic algorithms for the routing and wavelength assignment of scheduled lightpath demands in optical networks. *IEEE Journal on Selected Areas in Communications, 24*(8), 2–15.

Skorin-Kapov, N. (2007). Routing and wavelength assignment in optical networks using bin packing based algorithms. *European Journal of Operational Research, 177*(2), 1167–1179.

Skorin-Kapov, N. (2008). WDM optical networks planning using greedy algorithms. In W. Bednorz (Ed.), *Greedy algorithms* (pp. 569–586). InTech Publishers.

Wauters, N., & Demeester, P. (1996). Design of the optical path layer in multiwavelength cross-connected networks. *IEEE Journal on Selected Areas in Communications, 14*(5), 881–892.

Zang, H., Jue, J. P., & Mukherjee, B. (2000). A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *SPIE Optical Networks Magazine, 1*(1), 47–60.

Zheng, J., & Mouftah, H. (2004). *Optical WDM networks*. Wiley Interscience.