International Journal of Mathematics, Statistics and Operations Research
Vol. 1, No. 1, 2021 : pp. 75-99

ARF INDIA
Academic Open Access Publishing
www.arfjournals.com

MAXIMIZING THE NUMBER OF SCHEDULED LIGHTPATH DEMANDS IN OPTICAL NETWORKS BY CONFLICT GRAPHS

OLIVIER HUDRY[1]

Article History

Received : 16 June 2021
Revised : 24 June 2021
Accepted : 7 July 2021
Published : 1 August 2021

ABSTRACT: In an optical network, a Scheduled Lightpath Demand (SLD) is a connection demand between two nodes, during a certain time and with a certain wavelength. We consider the following NP-hard Routing and Wavelength Assignment (RWA) problem dealing with SLDs: given a set of SLDs and a number W of wavelengths, maximize the number of SLDs to which we can assign a lightpath (*i.e.* a routing path and a wavelength) without exceeding the number W of available wavelengths. The constraints are: a same wavelength must be assigned all along 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 study an approach stating the problem as the successive searches of independent sets in some conflict graphs. Moreover, we improve this approach thanks to a post-optimization method. The experimental results show that this model and the post-optimization method are quite efficient to provide a large number of routed SLDs.

Keywords: WDM Optical Networks, Routing and Wavelength Assignment (RWA) Problem, Scheduled Lightpath Demands (SLD), Combinatorial Optimization, Conflict Graphs, Independent Sets, Post-Optimization.

1. Introduction

In this paper we deal with a problem related to the routing and wavelength assignment problem (RWA) in wavelength division multiplexing (WDM) optical networks (see for instance Mukherjee (2006) or, for more recent references on these networks, Chadha (2019); see Cheng *et al.* (2006) or Resende and Pardalos (2006) for references on optimization problems in telecommunications). Fiber-optic 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. More precisely, a WDM optical network is represented by an undirected graph $\mathcal{G}$, and a number W of available wavelengths. We consider a set S of n scheduled lightpath demands (SLD) to be established in this network (see Kuri *et al.* (2003)): an SLD s belonging to S is characterized by a quadruplet $s = (x, y, \alpha, \beta)$ where:

[1] Télécom-Paris & LTCI, Palaiseau, France. E-mail: *olivier.hudry@telecom-paris.fr*

To cite this article:

Olivier Hudry. Maximizing the Number of Scheduled Lightpath Demands in Optical Networks by Conflict Graphs. *International Journal of Mathematics, Statistics and Operations Research,* 2021. 1(1): 75-99

- x and y are two nodes (or vertices) of $\mathcal{G}$,
- α and β denote the set-up and tear-down dates of the demand.

In order to satisfy an SLD s, we must find a path P_s between x and y in $\mathcal{G}$, and reserve a wavelength w_s along all the edges of this path and during all the timespan $[\alpha, \beta]$. The constraints related to the optical network are the following:

- The same wavelength must be used on all the edges used by a lightpath (otherwise wavelength converters are required, which involves large expenses and 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 *et al.* (2002)).
- At any given time and for any edge of $\mathcal{G}$, a same wavelength cannot be used to established several connection demands; 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.

A usual problem dealing with SLDs (see Kuri *et al.* (2003) or Zang *et al.* (2000); see Nazir and Arora (2019) for a recent survey on RWA) consists in looking for the minimum number of wavelengths necessary to establish all the SLDs. This problem and several variants are NP-hard (see Chlamtac *et al.* (1992) and Erlebach and Jansen (2001)). Chlamtac *et al.* (1992) proposed a first greedy heuristic to solve this problem. Then many other methods have been designed in the 1990's and after in order to provide approximate solutions (see for instance Choi *et al.* (2000), Dutta and Rouskas (2000), Zang *et al.* (2000), Kuri *et al.* (2003) or still Mukherjee (2006)). In 2006, Skorin-Kapov (2006) proposed a very efficient heuristic for this problem; our study is partially based on her heuristic (see Section 2). Integer linear programming and exact solutions have also been provided (see for instance Kumar and Kumar (2002), Ozdaglar and Bertsekas (2003), Jaumard *et al.* (2006), Mukherjee (2006) or Skorin-Kapov (2006)...). Several mathematical formulations have also been developed (see Ramaswami and Sivarajan (1995), Krishnaswamy and Sivarajan (2001), Lee *et al.* (2002), Jaumard *et al.* (2009)...).

In this paper, we consider another problem, anyway related to the usual one: given the graph, the set S of SLDs and the number W of available wavelengths, we want to determine a lightpath (a path and a wavelength) for as many SDLs of S as possible. Observe that this maximization problem allows also solving the usual one: indeed, if we want to determine the minimum number of wavelengths necessary to route all the SLD's, it is sufficient to vary the number W of wavelengths until this number is great enough to route all the SLDs of S.

In Section 2, we describe a greedy algorithm in order to try to maximize the number of SLDs that we can route with a given number of wavelengths.

Then we study in Section 3 a way to solve this problem formulated as the successive search of independent sets in conflict graphs. These independent sets are computed by means of an iterative improvement method (or descent algorithm). These two heuristics (the greedy one and the one using independent sets) are improved by a post-optimization method (Section 4). After having specified the experimental framework, we perform a preliminary study on a graph and an SLDs set in order to tune some parameters and to select the most efficient methods. Then we extend these results to other data sets (Section 5). Conclusions are finally presented in Section 6.

2. Greedy Method

In this paper, we consider first a greedy algorithm adapted from the method proposed by N. Skorin-Kapov (2006). Her method was originally designed to handle a more sophisticated version of the RWA of SLDs in which a connection may require several wavelengths to be established.

In our case, the greedy method works as follows. Let w be the current wavelength: we start with $w = 1$ and if $w > 1$, we assume that some SLDs have already been established using previous wavelengths. Then we determine the SLDs to which w will be assigned. We consider in turn the non-already established SLDs (following some predefined order). Let $s = (x, y, \alpha, \beta)$ be the current SLD. We consider a graph $\mathcal{H}(s)$ obtained from $\mathcal{G}$ by removing all the edges of the lightpaths already established with wavelength w and such that the associated SLDs overlap in time with s. Therefore $\mathcal{H}(s)$ contains only edges that can be used to route s without generating any conflict with previously established SLDs. So we just search for a shortest path between x and y in $\mathcal{H}(s)$; if such a path exists, it is assigned to s with wavelength w, otherwise s remains non-established for the time being. When all the SLDs have been considered with w, we start again the process with wavelength $w + 1$. The algorithm stops when all the SLDs have been established or when there is no remaining available wavelength ($w = W$).

This method will be referred to as G in the remainder. Its pseudo-code is given in Figure 1.

The order in which the SLDs are considered can be very important. In Skorin-Kapov (2006), the author proposes to sort the SLDs in decreasing order with respect to the number of required wavelengths (as mentioned previously, she deals with a version of the RWA of SLDs in which a connection may require several wavelengths to be established). The experimental results presented in Skorin-Kapov (2006) about this problem show that her algorithm provides generally very good results, very often optimal or nearly optimal.

For our problem, we can for instance sort the SLDs with respect to:

Input: A network $\mathcal{G}$; a set S of SLDs; a number W of wavelengths $1, ..., W$.
Output: For each $s \in S$: a lightpath (P_s, w_s) with $P_s = \emptyset$ if $w_s = W + 1$

For any $s \in S$, do: $(P_s, w_s) \leftarrow (\emptyset, W + 1)$
$w \leftarrow 0$
$S\_satisfied \leftarrow \emptyset$
While $(w < W)$ and $|S\_satisfied| < |S|$
  $w \leftarrow w + 1$
  For $s \in S \setminus S\_satisfied$ with $s = (x, y, \alpha, \beta)$
    $\mathcal{H}(s) \leftarrow \mathcal{G}$
    For $s' \in S\_satisfied$ with $s' = (x', y', \alpha', \beta')$
      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 shortest path P_s between x and y in $\mathcal{H}(s)$
      Assign the lightpath (P_s, w) to s
      $S\_satisfied \leftarrow S\_satisfied \cup \{s\}$

Figure 1: Greedy algorithm *G*

- a random order,
- the increasing or the decreasing order of the width of time intervals,
- the increasing or decreasing order of the distance between the source-destination nodes of the SLD (measured by the number of edges of a shortest path).

We found that the results obtained with these different orders are similar in average. So we chose to consider the SLDs in a random order, which permits to introduce a stochastic aspect (two runs of the method may provide different solutions). This will allow a fair comparison between the methods, since we could repeat them as many times as necessary to take advantage of a given amount of CPU time. Of course, the solution returned by such a repetition of a given method will be the best one computed over the different runs.

3. Heuristic Based on the Search of Independent Sets in Conflict Graphs

We describe in this section a model of the RWA problem which amounts to determine successively some independent sets of maximum cardinality in appropriate graphs referred to as *conflict graphs* (Section 3.1). Then we propose in Section 3.2 a heuristic based on a *descent* approach (also called *iterative improvement method*) in order to compute some large independent sets in these graphs.

3.1 Description of the Conflict Graphs

In order to formulate the RWA of SLDs as the determinination of independent sets, we start by choosing a number k between 1 and a few units (for instance 5; the experiments–see Section 5.3–show that greater values for k require more CPU time without improving the quality of the computed solutions). Then we compute, for each SLD s to be established, the k shortest paths between the origin and destination nodes x and y of s in $\mathcal{G}$ (we can use for instance Yen's algorithm; see Yen (1971); other algorithms of smaller complexity but more difficult to implement exist–see Eppstein (1998) for instance–but, since the number k is quite small, it is not worth considering them here). Now that we have determined k shortest paths for each SLD (or less if there exist less than k paths from x to y in $\mathcal{G}$), we build a new graph $\mathcal{C}_k$ (which will be the conflict graph mentioned above) with $N_k = n \times k$ vertices. To any pair (s, p) consisting of an SLD s and one of its corresponding k shortest path p, we associate a vertex $v(s, p)$ in the conflict graph $\mathcal{C}_k$. Two vertices $v(s_1, p_1)$ and $v(s_2, p_2)$ will be joined by an edge if:

- $s_1 = s_2$ and $p_1 \neq p_2$; in other words, the k vertices associated with the same SLD form a clique;
- $s_1 \neq s_2$, and the timespan of s_1 and s_2 overlap, and the paths p_1 and p_2 have at least one edge in common; that means that the connection requests s_1 and s_2 cannot be routed with p_1 and p_2 and be assigned the same wavelength.

Once we have built the conflict graph $\mathcal{C}_k$, we wish to determine an independent set in $\mathcal{C}_k$ of maximum cardinality. Since the conflict graph $\mathcal{C}_k$ may contain many vertices, and since the determination of a maximum independent set is an NP-hard problem (see Garey and Johnson (1979)), we do not try to solve this problem exactly, we just apply a heuristic (see below).

After obtaining a first independent set (of cardinality as large as possible) in $\mathcal{C}_k$, we establish, for each vertex $v(s, p)$ of this set, the corresponding SLD s using the wavelength 1 and the path p. Then we remove from the conflict graph all the vertices corresponding to the established SLDs (for each SLD, we remove its associated clique). We obtain therefore a new conflict graph associated with the SLDs which are not yet established, and we repeat the same process with the following wavelength until all the SLDs are established or all the wavelengths have been considered. Observe that we do not seek to partition the set of vertices of $\mathcal{C}_k$ into independent sets, since we just need to choose one vertex of each clique associated to an SLD in order to establish this one; so our model does not amount to a colouring problem, as in Noronha and Ribeiro (2006).

3.2 Descent algorithm for the computation of an independent set in the conflict graph

In order to determine some independent sets as large as possible in the conflict graph, we chose to apply a descent algorithm. First we examined some more sophisticated metaheuristics such as the simulated annealing, but the high computation time required to make profitable such a method, as well as the difficulty relative to the tuning of the parameters, led us to give up this type of tough approaches. The objective of the algorithm is to find a large independent set, though not necessarily of maximum cardinality, in the conflict graph.

The principle of the method is the following. We start from an independent set I_1 of cardinality 1 (by choosing any vertex), and we try to determine an independent set I_2 of cardinality 2, then an independent set I_3 of cardinality 3, and so on, until such a set cannot be found for some cardinality α; then the independent set $I_{\alpha-1}$ of cardinality $\alpha - 1$ is retained.

During this process, when an independent set I_α of cardinality α has been obtained, we try to build the independent set $I_{\alpha+1}$ by adding to I_α a vertex drawn randomly. We obtain a set T of vertices; generally there exist some edges between the added vertex and some vertices of I_α, and therefore T is not an independent set (otherwise $I_{\alpha+1} = T$ and we move to the next step). Then we modify T by applying some *elementary transformations* (defined in the following section) that transform T into other sets of the same cardinality. Hence the determination of an independent set of cardinality α amounts to the minimization of an objective function defined as the number of edges that join two vertices of T: T will be an independent set if this number is equal to zero. This minimization problem is solved by a descent algorithm described below.

3.2.1 Description of the descent method

As we have just mentioned, the application of a descent to our problem requires the definition of an *elementary* or *local transformation* (see for instance Glover and Kochenberger (2003) or Dreo *et al.* (2006)) permitting to move from the current set T to another set of the same cardinality. The chosen elementary transformation consists in replacing one of the vertices of T by another vertex of the conflict graph which is not in T. The new set T' thus obtained from T is a "neighbour" of T. If the cardinality of T is α, T has $\alpha \times (N_k - \alpha)$ distinct neighbours, where N_k still denotes the number of vertices of the conflict graph C_k, forming the neighbourhood of T.

We say that an elementary transformation from T to T' is *favourable* if it does not worsen the objective function, *i.e.* if the number of edges between the vertices of T' is not greater than the number of edges between the vertices of T. More precisely, let t and z be the exchanged vertices, with $t \in T$, $z \notin T$, $t \notin T'$ and $z \in T'$. Let q (respectively q') be the number of edges between t (resp. z) and the other vertices of T (resp. T'). The variation $\Delta(t, z)$ of the objective function generated by the elementary transformation from T to T' is equal to $\Delta(t, z) = q' - q$. The transformation is favourable if $\Delta(t, z) \leq 0$, or equivalently if $q' \leq q$.

The principle of the descent algorithm is the following. We start from an initial set T of α vertices of the conflict graph. Then we undertake the exploration of the neighbourhood of T by drawing randomly two vertices t (in T) and z (not in T) following a uniform probability distribution. As soon as we have found one elementary transformation that is favourable, we apply it to T and so we move from T to one of its neighbours. We repeat this process until either the objective function becomes equal to zero, which means that an independent set of cardinality α has been found, or all neighbours of T are discarded.

This last stop condition is defined by a maximum number of neighbours to be examined. The role of this number is twofold; first it permits to end the algorithm when none of the trials has been favourable and the objective function is still positive. In this case the method fails to provide an independent set. Second, since an elementary transformation that involves no variation of the objective function is considered as favourable, and so accepted, we may otherwise get stuck inside a loop, going from one set to one of its neighbours and then coming back to the initial set. The definition of a maximum number of neighbours to be examined prevents us from such a phenomenon. We set this number proportional to the size of the neighbourhood of T, *i.e.* as $\psi \times \alpha \times (N_k - \alpha)$. The proportionality coefficient ψ is defined as a parameter of the method; its value must be fixed beforehand. Some preliminary experiments have been performed on a given instance (see Section 5.2) in order to determine the most suitable value of ψ; this value has been kept for all the experimental studies carried out in this paper.

Some variants of the descent method have been tested, in particular the one where only strictly improving transformations are accepted, or variants including other ways to explore the neighbourhoods. We obtained similar results, no better than the results provided by the method described above, so we chose not to present those variants.

Input: A conflict graph $\mathcal{C}_k$ on N_k vertices; a set T of α vertices of $\mathcal{C}_k$;
Output: If possible, an independent set T of α vertices of $\mathcal{C}_k$; otherwise, a set T of α vertices of $\mathcal{C}_k$ in which we tried to minimize the number of edges.

$i \leftarrow 1$
While $i \leq \psi \times \alpha \times (N_k - \alpha)$ and T is not an independent set
  Draw a vertex t of $\mathcal{C}_k$ belonging to T with a uniform probability
  Draw a vertex z of $\mathcal{C}_k$ not belonging to T with a uniform probability
  Compute the number q of edges between t and $T \setminus \{t\}$
  Compute the number q' of edges between z and $T \setminus \{t\}$
  If $q' - q \leq 0$, then $T \leftarrow (T \setminus \{t\}) \cup \{z\}$
  $i \leftarrow i + 1$

Figure 2: The descent algorithm for the search of an independent set with a given cardinality

When the descent is over, either the current set T is an independent set of cardinality α, or it remains some edges in T. In the first case, we add a new vertex to T and we repeat the descent algorithm in order to find an independent set of cardinality $\alpha + 1$. In the second case, the process ends and we consider the last obtained independent set (of cardinality $\alpha - 1$) as the final solution given by the method.

The pseudo-code of the descent algorithm for the search of an independent set with a given cardinality is presented on Figure 2, and the *pseudo-code* concerning the whole method to compute an independent set with a large cardinality is given in Figure 3.

3.2.2 Improvement of the descent method thanks to the greedy heuristic

For each wavelength w, after applying the descent algorithm, we try to improve the obtained solution thanks to the greedy method described in Section 2. More precisely, we consider the SLDs still unsatisfied after the application of the descent, and we look for a path permitting to route each one of them with the wavelength w without changing the already-established SLDs. This is possible if there exists a path between the source and destination

Input: A conflict graph $\mathcal{C}_k$ on N_k vertices.
Output: An independent set T^* of $\mathcal{C}_k$ with a cardinality as large as possible.

$T \leftarrow \{x\}$ where x denotes any vertex of $\mathcal{C}_k$
While T is an independent set of $\mathcal{C}_k$
  $T^* \leftarrow T$
  $T \leftarrow T \cup \{y\}$ where y denotes any vertex of $\mathcal{C}_k$ which does belong to T
  Apply the descent of Figure 2 to T

Figure 3: The descent-based algorithm to compute an independent set witha large cardinality

nodes of the considered SLD s which does not contain any edge belonging to the already-established lightpaths associated with w and that overlap in time with s. This may arise since we considered only a small number k of shortest paths between each source-destination pair when building the conflict graph. So even if there was no possibility to route s when considering the conflict graph, there may still be some suitable path in the initial network $\mathcal{G}$.

The pseudo-code of the global method (repeated descent methods with the greedy improvement) is presented in Figure 4. Since this method is based on the conflict graphs, that are obtained with respect to the value k of the number of considered shortest paths, it will be denoted as D_k. When k varies, we obtain as many descent methods noted $D1$, $D2$, $D3$, etc.

4. A Post-optimization Method

The post-optimization method presented in this section aims at improving the results provided by the greedy heuristic or by the descent depicted above, though it can be applied to any heuristic permitting to solve the addressed problem or other variants of this problem, as done with success in Belgacem *et al.* (2014) for the usual problem of computing the minimum number of wavelengths necessary to satisfy all the SLDs.

Input: A network $\mathcal{G}$; a set S of SLDs; an integer k;
a number W of available wavelengths $1, ..., W$.

Output: For each $s \in S$: a lightpath (P_s, w_s) with $P_s = \emptyset$ if $w_s = W + 1$

For any $s \in S$, do $(P_s, w_s) \leftarrow (\emptyset, W + 1)$
$S_{unsatisfied} \leftarrow S$
For w varying from 1 to W with a step of 1
  Build the conflict graph $\mathcal{C}_k$ associated with $\mathcal{G}$, k and $S_{unsatisfied}$
  Compute an independent set T^* of $\mathcal{C}_k$ as large as possible
    by applying the method of Figure 3
  For any vertex (s, p) of T^*
    $(P_s, w_s) \leftarrow (p, w)$
    $S_{unsatisfied} \leftarrow S_{unsatisfied} \setminus \{s\}$
  For $s \in S_{unsatisfied}$ with $s = (x, y, \alpha, \beta)$
    $\mathcal{H}(s) \leftarrow \mathcal{G}$
    For s' satisfied with $s' = (x', y', \alpha', \beta')$
      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 shortest path p between x and y in $\mathcal{H}(s)$
      $(P_s, w_s) \leftarrow (p, w)$
      $S_{unsatisfied} \leftarrow S_{unsatisfied} \setminus \{s\}$

Figure 4: The descent-based algorithm D_k for the computation of successive independent sets in order to satisfy as many SLDs as possible, with the improvement brought by the greedy algorithm

The principle of this method is the following. First, we create an extra wavelength $W+1$ which is fictitiously associated with all the unsatisfied SLDs (there is no path associated with these SLDs); notice that it is what we did for the greedy heuristic G or for the descent methods D_k above. For any $w \in \{1, 2, ..., W, W+1\}$, let the *layer* $L(w)$ be the set of the SLDs with w as their wavelengths. We try to empty $L(w)$, at least partially, the ultimate aim being to empty $L(W+1)$ as much as possible. This is done, for $w > 1$, by trying to assign a smaller wavelength $(1, 2, ..., w-1)$ to the demands belonging to $L(w)$, which leads us to rearrange the wavelengths assigned to the SLDs of these lower layers. So, during this operation, the layers of some SLDs change but all of them must remain inside the interval $[1, w-1]$. To empty $L(w)$, even only partially, may allow us to route some unsatisfied SLDs when we will try to empty $L(W+1)$ in its turn and, globally, this operation will increase the number of routed SLDs.

More precisely, let $s = (x, y, \alpha, \beta)$ be the demand that we would like to move from its current layer $L(w)$ to a lower layer $L(\ell)$ ($\ell \in [1, w-1]$). It is very likely that some of the demands belonging to $L(\ell)$ 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 that overlap s in time, we may find no path joining x and y.

To avoid this, we remove some SLDs from $L(\ell)$. More precisely, we consider the SLDs of $L(\ell)$ greedily in any prescribed order: $s_1, s_2, ..., s_{|L(\ell)|}$. For each such SLD $s_i (1 \leq i \leq |L(\ell)|)$, we build a graph $\mathcal{H}_i$ as follows. We start with $\mathcal{G}$ and se set $\mathcal{H}_0 = \mathcal{G}$. When dealing with s_i, let P_i be the path associated with s_i. If the deletion of the edges of P_i from $\mathcal{H}_{i-1}$ does not prevent us from finding a path between x and y, then we delete them in order to obtain $\mathcal{H}_i$: $\mathcal{H}_i = \mathcal{H}_{i-1} \setminus P_i$; in this case, s_i remains inside $L(\ell)$. Otherwise, we remove s_i from $L(\ell)$, we put it aside in a set E and we do not change the graph $\mathcal{H}$: $\mathcal{H}_i = \mathcal{H}_{i-1}$. Thus, once all the demands of the layer $L(\ell)$ 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 $1, ..., w-1$, 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 $L(\ell)$, 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 $L(w)$. Otherwise we consider that the attempt to move s to the layer $L(\ell)$ has failed, and we try to move it to the next layer $L(\ell+1)$. If all the layers from $L(1)$ to $L(w-1)$ have been examined in vain, s remains inside the layer $L(w)$, and we move up to the following demand of $L(w)$.

This method, summarized in Figure 5, 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 of unsatisfied SLDs, it may happen that repeated 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 two consecutive runs do not increase the number of routed SLDs. This choice is based on an experimental observation and arises from a compromise between CPU time and the quality of the computed solutions.

In the following, if M denotes a method, $M+$ will denote the application of M followed by the application of the post-optimization algorithm to the results provided by M. Thus we obtain the methods $G+$, $D1+$, $D2+$, etc.

5. Experimental Results

5.1 Experimental Framework

We present below the results obtained for 18 instances. These instances are based on two graphs, which are often used to illustrate RWA problems:

- $\mathcal{G}57$ (57 vertices and 85 edges), extracted from the European optical transport network (see Figure 6);
- $\mathcal{G}29$ (29 vertices and 44 edges), representing a hypothetical North-American backbone network (see Figure 7).

We then generate two sets of demands for each considered network, with respectively 500 and 1000 SLDs. We thus obtain four pairs (graph, set of SLDs). The name of each pair is obtained by the concatenation of the name of the network with the number of SLDs to route: $\mathcal{G}57$-500 denotes the pair for which the network is $\mathcal{G}57$, with 500 SLDs; similarly, the other three pairs are denoted $\mathcal{G}57$-1000, $\mathcal{G}29$-500, and $\mathcal{G}29$-1000. When dealing with 500 SLDs, we consider four values for W: 5, 10, 20, and 30; when dealing with 1000 SLDs, we consider five values for W: 10, 20, 30, 40, and 50. Hence a total of 18 instances ($\mathcal{G}$, S, W).

The sets of SLDs are generated randomly so that the number of time-overlaps is significant but not too large: if they are too few, there are few clashes between the demands and therefore the addressed problem becomes too easy; to the contrary if the time-overlaps are too numerous, the number of required wavelengths increases greatly and the problem becomes again less interesting. According to our experiments, this can be achieved by drawing the source and destination nodes of each SLD $s = (x, y, \alpha, \beta)$

Input: A network $\mathcal{G}$; a set S of SLDs; a number W of wavelengths $1, ..., W$
a fictitious wavelength $W+1$
for each $s \in S$: a lightpath (P_s, w_s) with $P_s = \emptyset$ if $w_s = W+1$

Output: for each $s \in S$: a lightpath $(\tilde{P}_s, \tilde{w}_s)$ with $\tilde{P}_s = \emptyset$ if $\tilde{w}_s = W+1$

$w \leftarrow 2$
While $w \leq W+1$
  For $s \in S$ with $s = (x, y, \alpha, \beta)$ and with $w_s = w$
    For ℓ from 1 to $w-1$
      $E \leftarrow \emptyset$
      $\mathcal{H} \leftarrow \mathcal{G}$
      For $s' \in S$ with $s' = (x', y', \alpha', \beta')$ and with $w_{s'} = \ell$
        If $[\alpha, \beta] \cap [\alpha', \beta'] \neq \emptyset$
          Delete the edges of $P_{s'} \cap \mathcal{H}$ from $\mathcal{H}$
          If there exists no path between x and y in $\mathcal{H}$
            Remove s' from layer $L(\ell)$ and put s' in E
            Restore the removed edges of $P_{s'}$ in $\mathcal{H}$
      Compute a shortest path p between x and y in $\mathcal{H}$
      $(P_s, w_s) \leftarrow (p, \ell)$
      For $s' \in E$ with $s' = (x', y', \alpha', \beta')$
        For λ from 1 to $w-1$
          $\mathcal{H}(s') \leftarrow \mathcal{G}$
          For $s'' \in S$ with $s'' = (x'', y'', \alpha'', \beta'')$ and with $w_{s''} = \lambda$
            If $[\alpha', \beta'] \cap [\alpha'', \beta''] \neq \emptyset$
              Delete the edges of $P_{s''}$ from $\mathcal{H}(s')$
          If there exists a shortest path p between x' and y' in $\mathcal{H}(s')$
            $(P_{s'}, w_{s'}) \leftarrow (p, \lambda)$
            Exit the λ-for loop and move up to the next demand s'
        If s' has not been established
          Put back all the demands of E in layer $L(\ell)$
                with their initial lightpaths
          Restore the initial lightpath for s
          Exit the s'-for loop and move up to the next layer $L(\ell+1)$
                to try to add s inside layer $L(\ell+1)$
  $w \leftarrow w+1$
For each $s \in S$
  If $w_s \neq W+1$ $(\tilde{P}_s, \tilde{w}_s) \leftarrow (P_s, w_s)$
  Else $(\tilde{P}_s, \tilde{w}_s) \leftarrow (\emptyset, W+1)$.

Figure 5: The Post-optimization Algorithm

randomly with a uniform distribution of probability over all the vertices of the considered graph (of course the two nodes must be different). The setup time α and tear-down time β 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 equal to 300 for 500 SLDs or to 250 for 1000 SLDs. The time-window of s is then of the form $[c - L \times r^2, c + L \times r^2]$, where r is a real number drawn uniformly between 0 and 1.

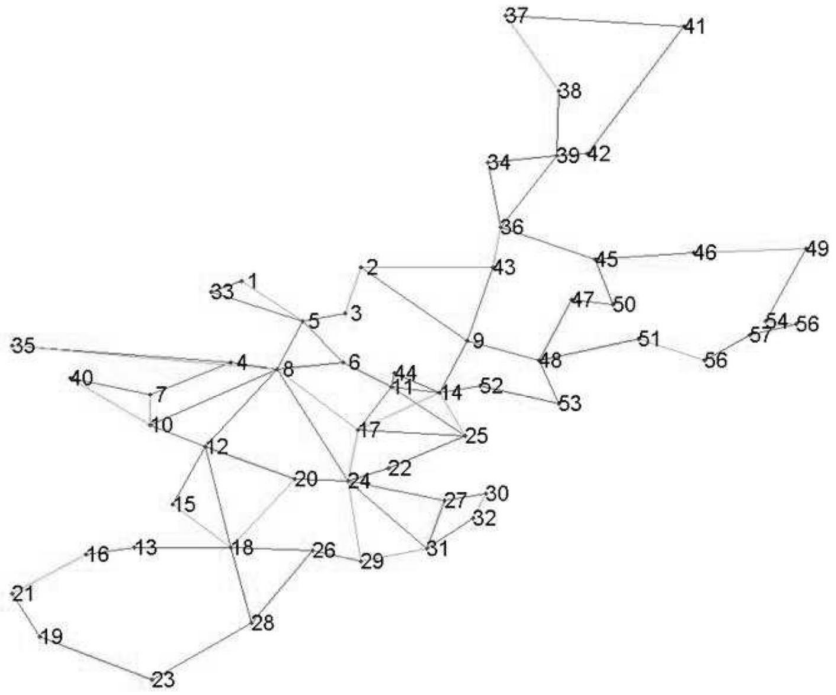

Figure 6: The network $\mathcal{G}57$

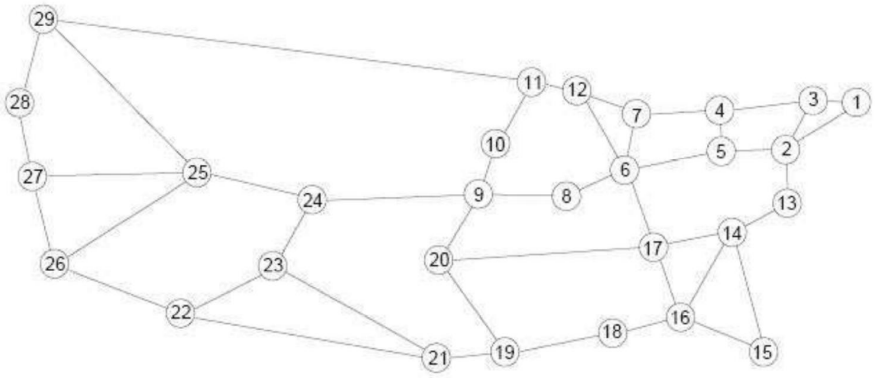

Figure 7: The network $\mathcal{G}29$

In the next sections, we present the results provided by the heuristics described above when applied to these 18 instances. We performed other experiments, dealing with other graphs, other sets of SLDs, or other numbers

of wavelengths: they lead to the same qualitative conclusions. The experiments have been performed on Solaris Sun stations (Sun Ultra 20M2 3 Ghz dual core AMD). In order to evaluate the heuristics on these instances, we have carried out 100 runs of each method for each instance and we compute, over the 100 runs:

- the average CPU time;
- the average number of SLDs that we succeeded to route;
- the maximum and minimum numbers of routed SLDs.

In order to compare two methods M and M', we will write $M < M'$ to mean that M' provides better results than M: the average number of SLDs satisfied by M' is greater than the average number of SLDs satisfied by M. Moreover, to measure the relative improvement brought by a method M with respect to G, we will consider the quantity $(N_M - N_G)/N_G$, where N_M (respectively N_G) denotes the number of SLDs satisfied by M (respectively by G).

5.2 Preliminary Study on One Instance

In this section, we consider the results obtained when applying different variants of the methods described above on the instance $\mathcal{G}57$-500, with $W = 20$ wavelengths. The objective is to identify the assets of each variant, and select the methods that will be studied more intensively in the remainder.

We start first by fixing the number of neighbours examined in each descent iteration, or more precisely, the value of ψ. For example, let us consider the method $D1$, with different values of ψ; the obtained results are presented on Table 1 (for each value of ψ, we give the average number of satisfied SLDs and the average time in seconds when considering 100 runs of $D1$). These results are displayed on Figure 8, where the x-axis corresponds to the values of ψ, and the y-axis denotes the average number of satisfied SLDs.

We can observe that the quality of the method does not increase much with ψ, whereas the required computation time increases regularly. So we chose to fix $\psi = 3$ as a compromise between a good quality and a reasonable time. Similar studies have been done with other instances and other variants of the method, and this value seemed a fair choice for every case. This setting will be used for all the experiments presented in the remainder.

Let us now estimate the interest of the post-optimization algorithm by considering the twelve following methods :

- the greedy method G;

Table 1: Tuning of ψ

ψ	1	2	3	4	5	
Av. # of routed SLDs	409.79	410.41	410.74	410.84	410.95	
CPU Time (in seconds)	0.14	0.205	0.257	0.308	0.355	
ψ	10	15	20	25	30	35
Av. # of routed SLDs	411.30	411.33	411.49	411.48	411.60	411.55
CPU Time (in seconds)	0.575	0.788	0.991	1.191	1.597	1.579

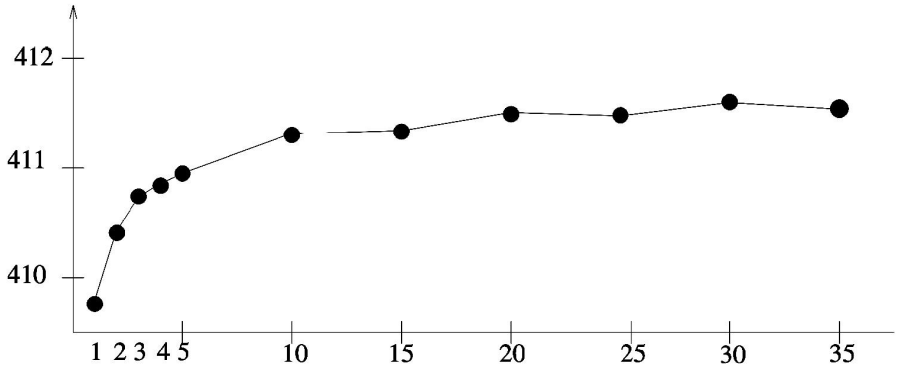

Figure 8: Tuning of ψ

- the greedy method combined with the post-optimization algorithm $G+$;
- the five descent methods $D1$, $D2$, $D3$, $D4$ and $D5$ (remember that D_k denotes the descent described above where k shortest paths are considered for the generation of the conflict graph);
- the same five descent methods combined with the post-optimization algorithm: $D1+$, $D2+$, $D3+$, $D4+$ and $D5+$.

For each case, we computed the average number of satisfied SLDs over 100 runs and we kept the worst result as well as the best one. Those results are presented in Figure 9; each value (worst, average and best) is represented by a small line. Moreover, Table 2 specifies the average number of satisfied SLDs over the 100 runs, the relative gain with respect to G (except for the method G itself), and the average computation time in seconds.

Those results show that it is clearly better to apply the post- optimization algorithm since it permits to establish several additional connection requests. The extra time required by the post-optimization is quite small (a few seconds). For the descent methods, especially for $D4$ and $D5$, this time is negligible in comparison with the time necessary to perform the descents. To the contrary, when comparing G and $G+$, almost the whole consumed time is devoted to the post-optimization method.

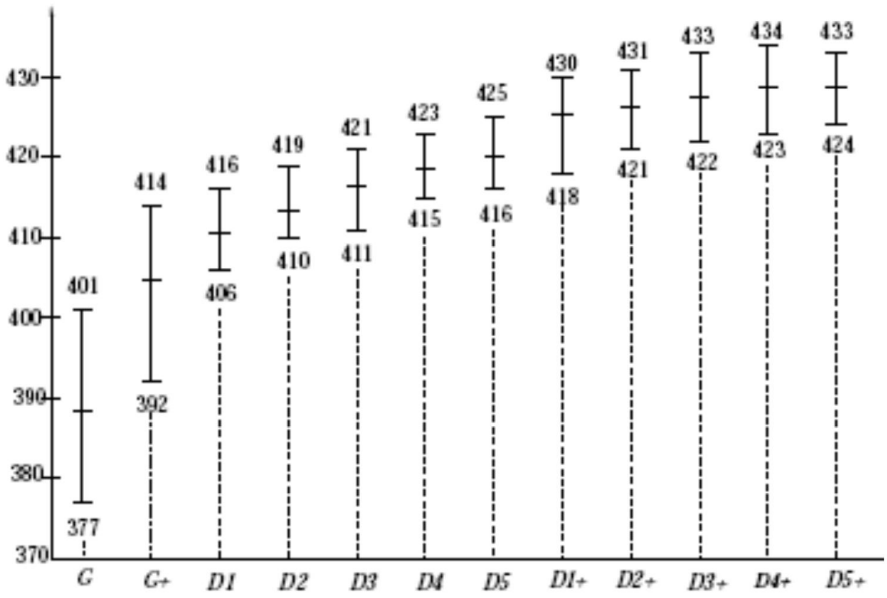

Figure 9: Results for the twelve methods applied to $\mathcal{G}$57-500 with $W = 20$

In the light of these results (corroborated by similar studies performed on other instances), we will keep for the remainder the methods G, $G+$, $D1+$, $D2+$, $D3+$, $D4+$ and $D5+$.

5.3 Other Results for $\mathcal{G}$57-500

Let us now consider the instances associated with the network $\mathcal{G}$57-500 and different values of the number of available wavelengths W: 5, 10, 20 or 30. Table 3 gives the average results when applying 100 runs of each method. The first column, labelled W, indicates the number of avalaible wavelengths;

Table 2: Average number of satisfied SLDs and average CPU time for the twelve methods applied to $\mathcal{G}$57-500 with $W = 20$

Method	*G*	*D1*	*D2*	*D3*	*D4*	*D5*
Av. # of routed SLDs	388.59	410.64	413.44	416.37	418.57	420.06
Improvement w.r.t. *G*		5.7 %	6.4 %	7.1 %	7.7 %	8.1 %
CPU time (seconds)	0.02	0.26	1.21	3.21	6.12	11.73
Method	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
Av. # of routed SLDs	404.72	425.34	426.35	427.58	428.69	429.13
Improvement w.r.t. *G*	4.2 %	9.5 %	9.7 %	10 %	10.3 %	10.4 %
CPU time (seconds)	1.91	2.28	3.28	5.11	8.04	12.27

Table 3: Results for $\mathcal{G}$57-500 and different values of W

W	G	$G+$	$D1+$	$D2+$	$D3+$	$D4+$	$D5+$
5	189.25	198.73	260.97	262.70	264.24	265.16	264.93
		5 %	37.9 %	38.8 %	39.6 %	40.1 %	40 %
	0.007	0.34	0.63	1.67	3.11	5.86	9.22
10	277.05	290.85	340.74	341.99	344.71	345.41	346.28
		5 %	23 %	23.4 %	24.4 %	24.7 %	25 %
	0.013	0.77	1.11	1.20	3.89	7.54	10.36
20	388.59	404.72	425.34	426.35	427.58	428.69	429.13
		4.2 %	9.5 %	9.7 %	10 %	10.3 %	10.4 %
	0.020	1.91	2.28	3.28	5.11	8.04	12.27
30	444.30	458.10	465.36	466.45	466.47	466.66	466.76
		3.1 %	4.7 %	5 %	5 %	5 %	5.1 %
	0.024	2.42	2.83	3.83	6.57	10.05	12.76

for each of these values, the first row specifies the average number of established SLDs, the second row gives the gain with respect to the basic greedy heuristic G, and the third row gives the CPU time in seconds.

According to these results, the ranking of the methods with respect to the number of satisfied SLDs is: $G < G+ < D1+ < D2+ < D3+ < D4+ < D5+$, with a significant gap between, on the one hand, the greedy approaches G and $G+$ and, on the other hand, the descent methods $D1+$ to $D5+$. This gap is the larger as the number of available wavelengths is small: for $W = 5$, the improvement with respect to G reaches 40 % for $D4+$ and $D5+$.

We can observe that the computation time required by the various methods are very different. In order to perform fair and easier comparisons, we propose to repeat each method until a certain amount of time (the same for all the methods) is reached.

The results obtained following this methodology for $W = 20$ are presented in Table 4, labelled *Comparison in triangle* (this representation will be used again in the remainder). The first row gives the name of the considered methods. The second row gives the time required by the greedy method G (0.02 s), and the average number of satisfied SLDs over 100 runs (388.59) for G. The third row gives first the average time T necessary to run $G+$, and then the average result (number of satisfied SLDs) obtained by repeating G until reaching the time T over 100 runs, and finally the average result for $G+$. Let us recall that when repeating a method, the returned value is the best value found over all the repetitions, so we consider the average of these best values when considering 100 runs of the repeated method. More generally, the first cell of the row i specifies the CPU time T required by one run of the

method which is in the column i of the table. Then we give the average results when repeating the methods that are faster than the current method until the time T is reached, and the last value of the row corresponds to the average result for the current method. The last row presents the average results when all the methods are repeated during 1 minute.

We observe in Table 4 that even with equivalent time, the ranking of the methods from the least efficient to the most efficient remains the same: $G < G+ < D1+ < D2+ < D3+ < D4+ < D5+$. Although the greedy approaches are repeated a great number of times, the gaps with respect to the results provided by the descent methods are still significant (the gain with respect to G is around 6 % and the gain with respect to $G+$ is around 2 %). On the other hand, for comparable computation times, the descent methods $D1+$ to $D5+$ provide average results quite similar; the results are only slightly better when the number of shortest paths considered to build the conflict graph is larger. This is why we did not consider larger values for the number of shortest paths in our experiments.

5.4 Results for $\mathcal{G}$29-500

We performed the same study as above for the graph $\mathcal{G}$29-500 with the number of wavelengths W equal to 5, 10, 20 or 30. We applied the seven methods G, $G+$, $D1+$, $D2+$, $D3+$, $D4+$, and $D5+$ to these 4 instances. The results for $W = 5$ are presented in Figure 10.

Then we compare the seven methods with equivalent computation times and still with $W = 5$, following the comparison in triangle process (see above). The results are presented in Table 5. This table gives also the time required by each method.

Finally, we give in Table 6, the results obtained with 5, 10, 20 or 30 available wavelengths.

Table 4: Comparisons in triangle of the methods G, $G+$, $D1+$, $D2+$, $D3+$, $D4+$ and $D5+$ for the instance $\mathcal{G}$57-500 with $W = 20$

Time	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
0.02 s	388.59						
1.91 s	400.86	404.63					
2.28 s	400.93	405.16	425.34				
3.28 s	401.08	405.89	426.16	426.35			
5.11 s	402.42	408.72	426.95	427.10	427.58		
8.04 s	402.85	410.39	427.42	427.75	428.28	428.69	
12.27 s	403.43	410.51	427.99	428.67	428.81	428.96	429.13
60 s	405.62	413.85	429.94	430.14	430.67	431.26	431.59

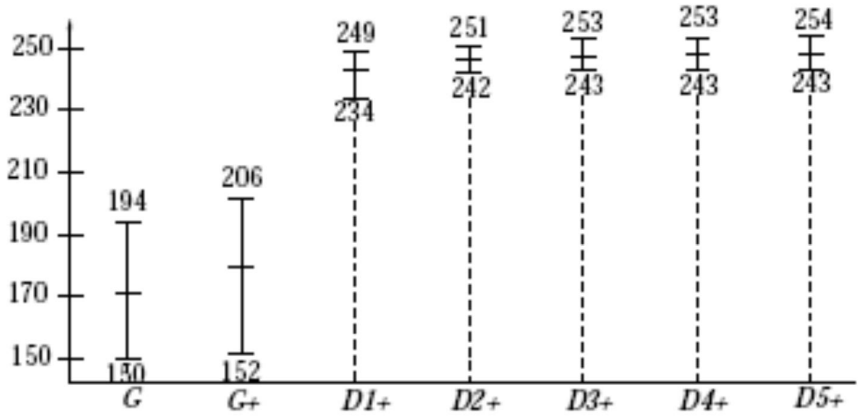

Figure 10: Results for $\mathcal{G}$29-500 with $W = 5$

Here again, the descent approaches $D1+$ to $D5+$ provide better results than the greedy methods G and $G+$. The gain of the descents with respect to G exceeds 40 % for small values of W; it reaches almost 45 % for $D5+$ with $W = 5$ available wavelengths. Even for higher values of W, for which G provides solutions closer to the optimum, the descent methods still permit to improve these solutions (gain of 7 to 8 % for $W = 30$). However the ranking between the descent methods becomes less clear: they all provide results very close, just slightly less good for $D1+$. For the same computation time (60 s, $W = 5$), the post-optimization method combined with the greedy method G yields a gain of 1.5 % whereas the gain produced when considering the descent methods with respect to the greedy method is very large, around 25 %.

5.5 Results for $\mathcal{G}$57-1000

We consider again the graph $\mathcal{G}57$ representing an optical network and we generate as previously a set of 1000 SLDs for this graph. First, we present the results obtained when considering 10 available wavelengths: Figure 11 gives the average values of the solutions provided by the various methods, and Table 7 presents the results for the same computation times. Finally, we give in Table 8 the results for various numbers of available wavelengths: 10, 20, 30, 40 and 50.

Here again, the methods $D1+$ to $D5+$ give clearly better results than G and $G+$, with a gain with respect to the greedy method G reaching almost 30 % for the small values of W. For larger numbers of available wavelengths, the gain is smaller but remains significant (around 3.7 % for $W = 50$). As

Table 5: Comparison in triangle for $\mathcal{G}$29-500 with $W = 5$

CPU time	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
0.005 s	171.5						
0.22 s	188.20	179.79					
0.43 s	190.35	190.10	243.27				
1.31 s	192.87	196.26	245.71	246.49			
3.08 s	194.76	193.71	246.57	247.57	247.22		
5.12 s	195.78	195.64	247.33	248.33	247.78	247.83	
8.97 s	197.02	197.80	247.86	248.83	248.60	248.27	248.18
60 s	200.15	203.17	249.61	251.00	251.04	250.91	250.14

Table 6: Results for G29-500 with various number of wavelengths W

W	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
5	171.50	179.79	243.27	246.49	247.22	247.83	248.18
		4.8 %	41.8 %	43.7 %	44.2 %	44.5 %	44.7 %
	0.0048	0.22	0.43	1.31	3.08	5.12	8.97
10	257.69	273.32	325.49	327.61	328.07	328.21	327.99
		6.1 %	26.3 %	27.1 %	27.3 %	27.4 %	27.3 %
	0.0078	0.485	0.74	1.78	3.27	6.09	11.50
20	365.48	384.32	415.19	417.40	417.77	417.84	418.12
		5.2 %	13.6 %	14.2 %	14.3 %	14.3 %	14.4 %
	0.012	1.39	1.49	2.71	3.99	7.02	12.55
30	437.54	459.82	469.85	471.14	470.89	471.18	470.89
		5.1 %	7.4 %	7.7 %	7.6 %	7.7 %	7.6 %
	0.015	1.98	2.19	3.05	4.80	7.69	3.50

previously, the descent methods give close results, a little less good for $D1+$. For the same time (300 s, $W = 10$), the post-optimization permits to improve G with a gain of 3 %. The gain of the descent methods relative to G remains well above, around 21 %.

5.6 Results for $\mathcal{G}$29-1000

In this section, we come back to the graph $\mathcal{G}29$, but with 1000 SLDs randomly generated. The results obtained with 30 wavelengths are displayed in Figure 12 and in Table 9; Table 9 gives the results of the comparisons in triangle, *i.e.* when the CPU times are the same for the different methods. Table 10 provides the results obtained when 10, 20, 30, 40 or 50 wavelengths are available.

As in the previous sections, the results provided by the methods $D1+$ to $D5+$ are significantly better than the ones of methods G and $G+$, with

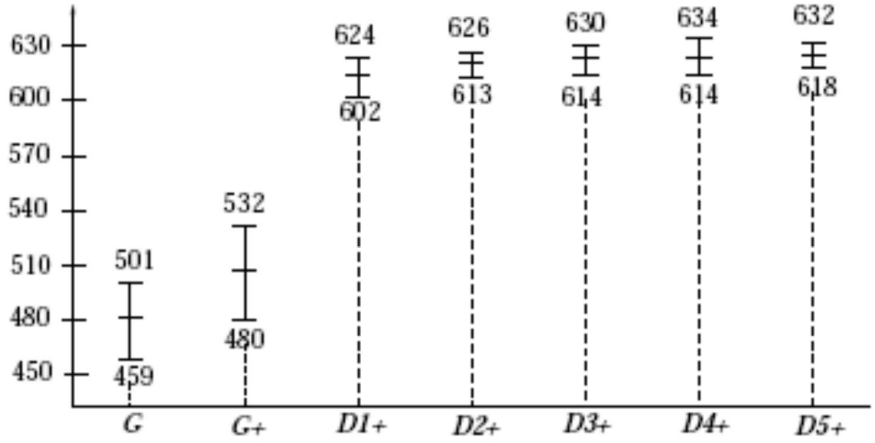

Figure 11: Results for $\mathcal{G}57$-1000 with $W = 10$

Table 7: Comparison in triangle for $\mathcal{G}57$-1000 with $W = 10$

Time	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
0.03 s	481.55						
2.28 s	503.49	506.74					
4.94 s	505.92	514.77	613.79				
10.96 s	508.07	518.83	616.13	620.54			
23.34 s	511.50	523.61	617.95	622.50	622.71		
45.23 s	513.40	526.53	619.00	623.61	623.93	623.37	
69.78 s	514.35	528.94	620.00	624.27	624.93	624.25	624.32
300 s	517.54	533.63	621.97	626.41	627.12	627.30	626.95

gains with respect to G larger than 37 % for small values of W. When W increases, the gain with respect to G decreases, but remains important, for instance about 4.4 % for $W = 50$. As above, the results computed by the descent methods are near each other. For a same CPU time (namely, 300 s with $W = 30$), the post-optimization method brings a relatively important improvement to G, about 4.4 %, while the gain due to the descent methods with respect to G is about 11.6 %.

6. Conclusion

It is well-known, in combinatorial optimization, that the most difficult, when considering heuristics, consists in reducing the gap existing between the results provided by the heuristics and the optimal values. The experiments

Table 8: Results for $\mathcal{G}57$-1000 with various number of wavelengths W

W	G	$G+$	$D1+$	$D2+$	$D3+$	$D4+$	$D5+$
10	481.55	506.74	613.79	620.54	622.71	623.37	624.32
		5.2 %	27.5 %	28.9 %	29.3 %	29.5 %	29.6 %
	0.034	2.28	4.94	10.96	23.34	45.23	69.78
20	665.08	701.37	771.77	777.16	778.68	779.87	781.50
		5.5 %	16 %	16.9 %	17.1 %	17.3 %	17.5 %
	0.047	5.99	8.88	15.98	29.29	58.35	79.63
30	788.76	820.24	863.67	867.54	868.84	869.67	870.79
		4 %	9.5 %	10 %	10.1 %	10.3 %	10.4 %
	0.060	9.94	15.38	21.13	35.15	57.54	86.72
40	870.45	901.64	924.51	926.59	928.27	928.17	928.62
		3.6 %	6.2 %	6.4 %	6.6 %	6.6 %	6.7 %
	0.069	13.64	17.86	28.74	44.71	61.00	87.92
50	931.30	960.23	965.47	965.47	965.67	965.91	965.92
		3.1 %	3.7 %	3.7 %	3.7 %	3.7 %	3.7 %
	0.076	15.68	18.29	28.60	43.94	61.73	87.23

reported in Section 5 as well as others, not reported here, show that, from this point of view, the two features presented in this article, *i.e.* the post-optimization method and the formulation of the problem as the search of independent sets, are very beneficial.

Compared to the greedy method G adapted from Skorin-Kapov (2006), these two features allow to significantly increase the number of SLDs that can be satisfied. When the number of available wavelengths is small, the

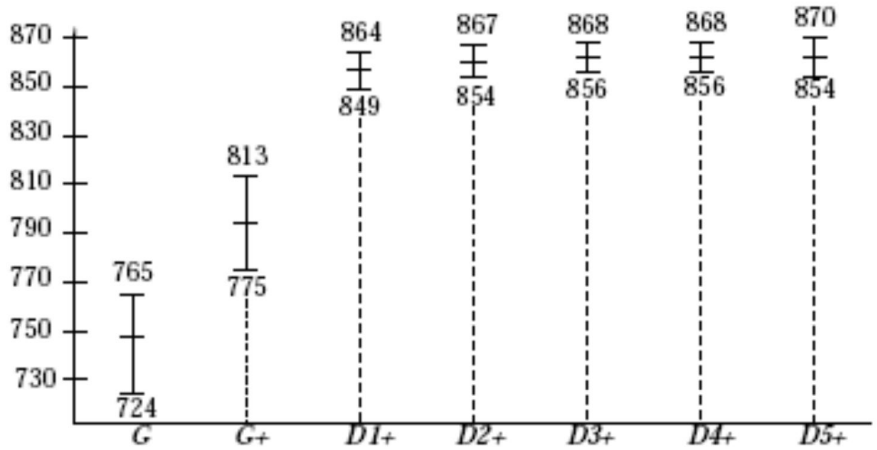

Figure 12: The seven methods G, $G+$, $D1+$ to $D5+$ applied to $\mathcal{G}29$-1000 for 30 wavelengths

Table 9: Comparison in triangle for G29-1000 with $W = 30$

Time	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
0.047 s	747.18						
7.20 s	768.24	793.87					
9.60 s	768.48	795.60	857.21				
14.67 s	769.45	797.24	857.79	861.01			
28.09 s	770.37	801.68	859.40	862.73	862.15		
48 s	771.60	804.46	860.83	863.71	863.31	862.16	
77.56 s	773.02	805.36	861.56	864.47	864.00	862.56	861.96
300 s	775.88	810.37	863.13	866.27	865.74	865.57	864.36

Table 10: Results for G29-1000 with respect to the number of wavelengths

W	*G*	*G+*	*D1+*	*D2+*	*D3+*	*D4+*	*D5+*
10	423.46	447.55	577.87	581.84	583.12	583.84	583.30
		5.7 %	36.5 %	37.4 %	37.7 %	37.9 %	37.7 %
	0.022	1.39	2.78	8.37	20.46	39.65	64.72
20	612.30	650.97	746.98	753.47	753.84	754.52	754.06
		6.3 %	22 %	23.1 %	23.1 %	23.2 %	23.2 %
	0.037	3.70	6.20	11.51	24.37	44.67	
30	747.18	793.87	857.21	861.01	862.15	862.16	861.96
		6.2 %	14.7 %	15.1 %	15.4 %	15.4 %	15.4 %
	0.047	7.20	9.60	14.67	28.09	48.00	77.56
40	848.66	898.86	930.54	931.98	932.46	932.39	932.67
		5.9 %	9.6 %	9.8 %	9.9 %	9.9 %	9.9 %
	0.055	11.34	13.92	21.61	32.45	53.08	93.87
50	926.85	967.33	968.17	968.09	968.02	967.95	967.95
		4.4 %	4.5 %	4.4 %	4.4 %	4.4 %	4.4 %
	0.060	12.78	10.73	17.29	35.46	51.63	80.89

descent methods may increase the number of satisfied SLDs by about 50 % with respect to the results given by G. Even by repeating G a large number of times, G remains far from the descent methods: the gains of the descents with respect to G may reach, or even exceed, 25 % (it is the case for instance for $\mathcal{G}$29-500 with $W = 5$ and 60 seconds).

When the number of wavelengths increases or when the CPU time increases so that it becomes possible to repeat all the methods (reaching 60 or 300 seconds in our experiments), this gain becomes less important because the results obtained by G are better and become closer to the optimal

values; but even in this case, the post-optimization method applied to G to get $G+$ and the descent methods improve the results of G by some percents and thus allow us to reduce a little more the gap between the obtained results and the optimal values.

Globally, it appears from the results mentioned above and from other results not reported here that the ranking of the methods according to their increasing efficiencies is the following, even when we give the same CPU times to all the methods: first G, which already provides good results, then $G+$, then $D1+$ and last the methods $D2+$, $D3+$, $D4+$ and $D5+$, whose results are very close and sometimes appear in different orders.

References

[1] Belgacem, L., Charon, I. and Hudry, O. (2014) "A Post-Optimization Method for the Routing and Wavelength Assignment Problem Applied to Scheduled Lightpath Demands", *European Journal of Operational Research.* 232, 298-306.

[2] Chadha, D. (2019) Optical WDM Networks: From Static to Elastic Networks, John Wiley and Sons Ltd, New York.

[3] Cheng, M.X., Li, Y. and Du, D.-Z. (2006), Combinatorial optimization in communication networks, Springer, Berlin.

[4] Chlamtac, I., Ganz, A. and Karmi *G.* (1992) "Lightpath communications: an approach to high-bandwidth optical WANs", *IEEE Trans. Commun.* 40, 1171–1182.

[5] Choi, J.S., Golmie, N., Lapeyere F., Mouveaux, F. and Su D. (2000) "A functional classification of routing and wavelength assignment schemes in DWDM networks: static case", Proc. VII Int. Conf. on Optical Communication and Networks, 2000.

[6] Chu, X., Li, B., Chlamtac, I. (2002) "On the wavelength converter placement for different RWA algorithms in wavelength-routed all-optical networks", Proc. SPIE Vol. 4874, OptiComm 2002: Optical Networking and Communications, N. Ghani and K. M. Sivalingam (eds), 186-197.

[7] Dréo, J., Petrowski, A., Taillard, É. and Siarry P. (2006), Metaheuristics for Hard Optimization, Methods and Case Studies, Springer, Berlin.

[8] Dutta R., Rouskas, *G.*N. (2000) "A survey of virtual topology design algorithms for wavelength routed optical networks", *Optical Networks Magazine.* 1(1), 73-89.

[9] Eppstein D. (1998) "Finding the k shortest paths", *SIAM J. Computing.* 28 (2), 652-673.

[10] Erlebach, T. and Jansen, K. (2001) "The complexity of path coloring and call scheduling", *Theoretical Computer Science*, 255 (1-2), 33-50.

[11] Garey, M.R. and Johnson, D.S. (1979) Computers and intractability, a guide to the theory of NP-completeness, Freeman, New York.

[12] Glover, F.W. and Kochenberger, G.A. (2003) Handbook of Metaheuristics. Kluwer Academic Publishers, New York.

[13] Jaumard, B., Meyer, C. and Thiongane B. (2006) "ILP formulations for the routing and wavelength assignment problem: symmetric systems", in Handbook of Optimization in Telecommunications, M.G.C. Resende and P.M. Pardalos (eds), Springer, Berlin, 637-677.

[14] Jaumard, B., Meyer, C. and Thiongane B. (2009) "On column generation formulations for the RWA problem" *Discrete Applied Mathematics*, 157 (6), 1291-1308.

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

[16] Kumar, M.S. and Kumar, P.S. (2002) "Static lightpath establishment in WDM networks - new ILP formulations and heuristic algorithms", *Computer Communications*, 25, 109-114.

[17] Kuri, J., Puech, N., Gagnaire, M., Dotaro, E. and Douville, R. (2003), "Routing and Wavelength Assignment of Scheduled Lightpaths Demands", *IEEE Journal on Selected Areas in Communications*, 21 (8), 1231–1240.

[18] Lee, K., Kang, K.C., Lee, T. and 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.

[19] Mukherjee, B. (2006) Optical WDM Networks, Springer, Berlin.

[20] Nazir, S. and Arora, J. (2019) "A Survey on Wavelength Assignment in WDM Systems", *International Journal of Research in Electronics and Computer Engineering*, 7(2), 3344-3348.

[21] Noronha, T. and Ribeiro, C. (2006) "Routing and wavelength assignment by partition coloring", *European Journal of Operational Research*, 171 (3), 797-810.

[22] Ozdaglar, A.E. and Bertsekas, D.P. (2003) "Routing and wavelength assignment in optical networks", *IEEE/ACM Transactions on Networking*, 11 (2), 259-272.

[23] Ramaswami, R. and Sivarajan, K.N. (1995) "Routing and Wavelength Assignment in All-Optical Networks", *IEEE/ACM Transactions on networking*, 3 (5), 489–500.

[24] Resende, M.G.C. and Pardalos, P.M. (eds) (2006) Handbook of Optimization in Telecommunications, Springer, Berlin.

[25] Skorin-Kapov, N. (2006) "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.

[26] Yen, J. Y. (1971) "Finding the K shortest loopless paths in a network", *Management Science*, 17, 712-716.

[27] Zang, H., Jue, J.P. and 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.