Efficient Implementations of Heuristics for Routing and Wavelength Assignment

Thiago F. Noronha[1], Mauricio G.C. Resende[2], and Celso C. Ribeiro[3]

[1] Department of Computer Science, Catholic University of Rio de Janeiro
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ 22453-900, Brazil
tfn@inf.puc-rio.br
[2] Algorithms and Optimization Research Department, AT&T Labs Research
Florham Park, NJ 07932-0971, United States
mgcr@research.att.com
[3] Department of Computer Science, Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, Niterói, RJ 24210-240, Brazil
celso@ic.uff.br

Abstract. The problem of Routing and Wavelength Assignment in Wavelength Division Multiplexing (WDM) optical networks consists in routing a set of lightpaths and assigning a wavelength to each of them, such that lightpaths whose routes share a common fiber are assigned to different wavelengths. When the objective is to minimize the total number of wavelengths used, this problem is NP-hard. The current state-of-the-art heuristics were proposed in 2007 by Skorin-Kapov. The solutions provided by these heuristics were near-optimal. However, the associated running times reported were high. In this paper, we propose efficient implementations of these heuristics and reevaluate them on a broader set of testbed instances.

1 Introduction

Information in optical networks is transmitted through optical fibers as optical signals. Each link operates at a speed of the order of terabits per second, which is much faster than the currently available electronic devices for signal reception and transmission. *Wavelength Division Multiplexing* (WDM) technology allows more efficient use of the huge capacity of optical fibers, as far as it permits the simultaneous transmission of different channels along the same fiber, each of them using a different *wavelength*. An all-optical point-to-point connection between two nodes is called a *lightpath*. It is characterized by its route and the wavelength in which it is multiplexed. Two lightpaths may use the same wavelength, provided they do not share any common fiber. Such networks require a large number of available wavelengths, especially when wavelength conversion is not available.

Given an optical network and a set of lightpath requests, the problem of *Routing and Wavelength Assignment* (RWA) in WDM optical networks consists

C.C. McGeoch (Ed.): WEA 2008, LNCS 5038, pp. 169–180, 2008.

in routing the set of lightpaths and assigning a wavelength to each of them, such that lightpaths whose routes share a common fiber are assigned to different wavelengths. Variants of RWA are characterized by different optimization criteria and traffic patterns, see e.g. [3,13]. We consider the min-RWA offline variant, in which all lightpath requests are known beforehand. No wavelength conversion is available, i.e. a lightpath must be assigned the same wavelength on all fibers in its route. The objective is to minimize the total number of wavelengths used. This problem is also known as the *Path Coloring Problem*. Erlebach and Jansen [4] showed that min-RWA is NP-hard.

State-of-the-art heuristics for min-RWA are discussed in the next section. Implementation issues are discussed and new heuristics are proposed in Section 3. Computational experiments illustrating the efficiency of the new implementations on a broad set of test instances are reported in Section 4. Concluding remarks are drawn in the last section.

2 Related Work

Different heuristics have been proposed for solving min-RWA. Some approaches decompose the problem into two subproblems: the routing subproblem and the wavelength assignment subproblem [2,5,7,9], while others tackle the two subproblems simultaneously [8,12]. A functional classification of RWA heuristics can be found in [3].

The current state-of-art heuristics for min-RWA were proposed by Skorin-Kapov [12]. Each wavelength is represented by a different copy of a bidirected graph $G = (V, A)$ that represents the physical topology of the optical network. Vertices in V and arcs in A represent network nodes and fibers, respectively. Lightpaths arc-disjointly routed in the same copy of G are assigned the same wavelength. The copies of G are associated with the bins and the lightpaths with the items of a bin packing problem [1]. Problem min-RWA is reformulated as that of packing the lightpaths using a minimum number of bins.

The *size* of a lightpath is defined as the hop-count shortest path between its endnodes in G. We notice that lightpaths are not necessarily routed on shortest paths. Whenever a lightpath is placed in a bin (i.e., a copy of G), all arcs in its route are deleted from the corresponding copy of G to avoid that other lightpaths use them. Therefore, the next lightpaths packed in that bin might not be able to be routed on a shortest path.

Four min-RWA heuristics were developed based on classical bin packing heuristics: (i) FF-RWA, based on the First Fit heuristic, (ii) BF-RWA, based on the Best Fit heuristic, (iii) FFD-RWA, based on the First Fit Decreasing heuristic, and (iv) BFD-RWA, based on the Best Fit Decreasing heuristic. The first is equivalent to the Greedy-EDP-RWA [8] heuristic, except for the order in which some steps are executed [12].

The pseudo-codes of FF-RWA, BF-RWA, FFD-RWA, BFD-RWA are similar. They are summarized in Figure 1. The inputs are the graph G, the set τ of

```
begin heuristic(G, τ, d)
1.  Let t be a permutation of the lightpaths in τ;
2.  Set Ω ← ∅ and S ← ∅;
3.  for i = 1, ..., |t| do
4.      Find the bin ω ∈ Ω where the shortest path of t_i in ω has less than d arcs;
5.      if no such a bin exists then do
7.          ω ← new copy of G;
8.          Ω ← Ω ∪ {ω};
9.      end if
10.     Let p_i be the shortest path between the endnodes of t_i in ω;
11.     S ← S ∪ (p_i, ω);
12.     Delete edges in path p_i from ω;
13. end-for;
14. return S;
end
```

Fig. 1. Pseudo-code of heuristics FF-RWA, BF-RWA, FFD-RWA, and BFD-RWA

lightpath requests, and the value d of the maximum number of links in each route. As suggested in [12], d is set to be the maximum of the square root of the number of links in the network and the diameter of G (i.e., the maximum value of a shortest path between two nodes in the network). The output is a set S of tuples (p_i, ω_i), for $i = 1, \ldots, |\tau|$, where p_i is the route followed by lightpath t_i and ω_i is the wavelength with which it is multiplexed. A permutation t of lightpaths in τ is built in line 1. In FF-RWA and BF-RWA, lightpaths are randomly distributed in t, while in FFD-RWA and BFD-RWA, they are sorted in non-increasing order of their sizes. In line 2, the set S and the set Ω of copies of G are initialized. The lightpaths are routed and assigned a wavelength in lines 3 to 13, one at a time, according to their order in t. A bin $\omega \in \Omega$ in which lightpath t_i can be routed with less than d arcs is sought in line 4. FF-RWA and FFD-RWA stop at the first bin found, while BF-RWA and BFD-RWA scan all bins in Ω and select that in which t_i fits with the smallest number of arcs (since the arcs in each copy of G are not necessarily the same). Let p_i be the shortest path between the endnodes of t_i in ω. If there is no bin in Ω where p_i fits with less than d arcs, then t_i is routed on a new copy of G that is created in line 7 and added to set Ω in line 8. The tuple (p_i, ω) is added to the solution in line 11, and all arcs in p_i are deleted from ω in line 12 to avoid that other lightpaths are routed on those arcs in this copy of G.

Numerical results in [12] showed that FFD-RWA and BFD-RWA outperformed Greedy-EDP-RWA [8], one of the best heuristic in the literature for min-RWA. However, the running times reported in [12] were very high. On the largest instances, running times of up to 8 minutes (Pentium IV 2.8 GHz) were reported. In the next section, we propose five different implementation strategies for FF-RWA, BF-RWA, FFD-RWA, and BFD-RWA and evaluate them in a broad set of test instances in Section 4.

3 Implementation Issues

Let $n = |V|$, $m = |A|$, and $l = |\tau|$. Furthermore, let $c^{sp}(m,n)$ be the computational complexity of a one-to-all shortest path algorithm applied to G, and $c^{del}(m,n)$ be the complexity of deleting an arc from G. The worst case complexity $T(n,m,l)$ of FF-RWA, BF-RWA, FFD-RWA, and BFD-RWA is calculated as follows. For sake of simplicity, we assume that $n < l$, which holds for all real and artificial networks in the literature. First, all-to-all shortest paths are calculated in time $O(n \cdot c^{sp}(m,n))$ in line 1, and sets Ω and S are initialized in constant time in line 2. Next, lines 3 to 13 are repeated for each lightpath t_i, with $i = 1, \ldots, l$. In line 4, a bin where t_i fits with less than d arcs is found in time $O(l \cdot c^{sp}(m,n))$. A new copy of G is created in time $O(m)$ in line 7 and added to set Ω in constant time in line 8. Finally, the set S is updated in constant time in line 11, while the arcs in p_i are deleted from w in time $O(d \cdot c^{del})$ in line 12. Therefore, the worst case complexity of these heuristics is

$$\begin{aligned} T(n,m,l) &= O(n \cdot c^{sp}(m,n)) + O(1) + O(l \cdot (l \cdot c^{sp}(m,n) + m + 1 + d \cdot c^{del}(m,n))) \\ &= O(n \cdot c^{sp}(m,n) + l^2 \cdot c^{sp}(m,n) + l \cdot m + l \cdot d \cdot c^{del}(m,n)). \end{aligned}$$

The efficiency of the heuristics depends on how fast a shortest path (SP, for short) query is performed and how fast an arc is removed from G. The traditional implementations using Dijkstra's algorithm and single linked lists to represent the adjacency matrix of G lead to $c^{sp}(m,n) = O(n \cdot \log n + m)$ and $c^{del}(m,n) = O(n)$. Therefore,

$$\begin{aligned} T(n,m,l) &= O(n \cdot (n \cdot \log n + m) + l^2 \cdot (n \cdot \log n + m) + l \cdot m + l \cdot d \cdot n) \\ &= O(l^2 \cdot (n \cdot \log n + m)). \end{aligned}$$

However, the hop-count shortest paths can be calculated using Breadth First Search (BFS) in time $O(m)$. In addition, any arc can be deleted in time $O(1)$ using the representation of G by adjacency lists M as follows. For each node $i \in V$, we keep a doubly linked list whose cells correspond to the arcs having i as their origin. Furthermore, we keep an array P pointing to the address of each cell in M. Whenever an arc (i,j) is to be removed, we use P to obtain the address of its corresponding cell in constant time. Since the adjacency list of node i is doubly linked, the cell corresponding to arc (i,j) can be deleted in time O(1). This data structure and the BFS algorithm were used in our standard implementation `STD` of the four heuristics. Therefore, the complexity of the min-RWA heuristics using `STD` is

$$T(n,m,l) = O(n \cdot m + l^2 \cdot m + l \cdot m + l \cdot d) = O(l^2 \cdot m).$$

The most expensive operation of the min-RWA heuristics appears in line 4 of Figure 1, where a SP query is performed in at most l bins for each of the l lightpaths in τ. At this point, only the value of the shortest path is required. Therefore, we propose another implementation based on an $n \times n$ distance matrix in which each entry is the value of the shortest path between the two corresponding nodes in G. It is initialized in $O(n \cdot m)$ in line 1 and instantiated for each

new bin created in $O(n^2)$ in line 7. As long as arcs are deleted from a bin, the shortest paths on that bin may change and the corresponding distance matrix must be updated.

The new data structure allows SP queries to be performed in constant time. However, the efficiency of the heuristics depends on how fast the updates are performed. Given the graph $G = (V, A)$ and a node $v \in V$, the shortest path (SP, for short) graph of v is a subgraph $G^v = (V, A^v)$ of G, with $A^v \subseteq A$, such that the path from any vertex i to v in G^v corresponds to the shortest path from i to v in G. If the graph is acyclic, it is called a shortest path tree. We experimented with four algorithms for updating the distance matrix: `RRg`, `RRt`, `NRRg`, and `NRRt`. The first two are based on the work of Ramalingam and Reps [10] for dynamically updating SP graphs and SP trees, respectively, while the last two are adaptations of the former two algorithms.

Given a node $v \in V$ and the SP graph G^v, the algorithm of Ramalingam and Reps [10] for dynamically updating SP graphs is based on the observation that when the weight of an arc $a \in A$ increases, the shortest paths from v to many vertices in V might not change and do not need to be recalculated. Arcs are deleted by increasing their weights to infinity. If $a \notin A^v$, no update is necessary; otherwise a is removed from G^v. Next, if G^v remains connected, the algorithm stops. Otherwise, the set U of vertices whose shortest paths to v have changed is identified and removed from G^v. Then, each vertex $u \in U$ is inserted into a priority queue Q with a key equal to the weight of the least cost arc from u to one of the vertices that remained in G^v. Finally, the algorithm proceeds in a Dijkstra-like fashion.

A variant of this algorithm is that of Ramalingam and Reps [10] for dynamically updating SP trees. The algorithm is similar to the one described above. However, the identification of the vertices in U and the shortest path updates are performed more efficiently in SP trees. Every time an arc $a \in A^v$ is deleted, the data structure has to be updated. Using a Fibonacci heap to implement Q, the worst case time complexity of both algorithms is $O(n \cdot \log n + m)$. However, since only deletions are performed and the arcs have unit cost, it can be implemented in time $O(m)$ by using a bucket to implement Q.

Algorithm `RRg` keeps one SP graph for each vertex in V. The SP graphs are initialized in $O(n \cdot m)$ in line 1 of Figure 1 and instantiated for each new bin created in $O(n \cdot m)$ in line 7. After each arc in p_i is deleted from ω in line 12, `RRg` checks if any SP graph of ω must be updated. If so, the algorithm of Ramalingam and Reps [10] for SP graphs is used, and the distance matrix is updated. In the worst case scenario, n SP graphs are updated in $O(n \cdot m)$. Therefore, the complexity of the min-RWA heuristics using `RRg` is

$$T(n, m, l) = O(n \cdot m + l^2 + l \cdot n \cdot m + l \cdot d \cdot (n \cdot m)) = O(l^2 + l \cdot d \cdot n \cdot m).$$

Algorithm `RRt` keeps one SP tree for each vertex in V. The SP trees are initialized in $O(n \cdot m)$ in line 1 of Figure 1 and instantiated for each new bin created in time $O(n^2)$ in line 7. As before, after each arc deletion, `RRt` checks if any SP tree of ω must be updated. If so, the algorithm of Ramalingam and

Reps [10] for trees of shortest paths is used, and the distance matrix is updated. Therefore, the complexity of the min-RWA heuristics using `RRt` is

$$T(n, m, l) = O(n \cdot m + l^2 + l \cdot n^2 + l \cdot d \cdot (n \cdot m)) = O(l^2 + l \cdot d \cdot n \cdot m).$$

Algorithm `RRg` (resp. `RRt`) might not be efficient, since the number of SP graphs (resp. SP trees) to be updated after each lightpath is assigned a bin may be very high. To remedy this, we propose a compromise implementation. Algorithm `NRRg` (resp. `NRRt`) uses the same data structure as algorithm `RRg` (resp. `NRRt`), but without updating the latter as soon as an arc is deleted. Therefore, the distance matrix gives a lower bound to the shortest path between any two nodes in time $O(1)$, since the shortest paths can only increase after an arc deletion. If the lower bound is larger than d, the correct distance is not needed. Otherwise, it can be calculated in time $O(d)$ by retrieving the shortest path in the SP graph (resp. SP tree) of the sink node of the lightpath. If no arc along the shortest path has been deleted, the value stored in the distance matrix is exact. Otherwise, algorithm `NRRg` (resp. `NRRt`) updates only the corresponding SP graph (resp. SP tree) from scratch and the entries of the distance matrix that have changed. Therefore, the worst case complexity of an SP query is $O(m)$. However, no SP graph (resp. SP tree) update is necessary in line 12 of Figure 1 and the arc deletion can be done in $O(d)$ for each lightpath. Therefore, the complexity of the min-RWA heuristics using algorithm `NRRg` is

$$T(n, m, l) = O(n \cdot m + l^2 \cdot m + l \cdot n \cdot m + l \cdot d) = O(l^2 \cdot m),$$

while using algorithm `NRRt` it is

$$T(n, m, l) = O(n \cdot m + l^2 \cdot m + l \cdot n^2 + l \cdot d) = O(l^2 \cdot m).$$

4 Computational Experiments

Four sets of testbed instances were used in the computational experiments. Set X was randomly generated as in [12]. Sets Y and Z are proposed in this paper. Finally, set W is a collection of the most studied realistic instances in the literature, together with two new instances introduced in this paper. All network topologies are connected and each link corresponds to a pair of bidirected fibers. The traffic matrices are asymmetric, i.e. there might be a lightpath request from a node i to a node j and not from j to i. A description of each set is presented below.

The set X of instances was randomly generated exactly as in [12]. The instances have 100 nodes, the probability P_e that there is a link between a pair of nodes is equal to 0.03, 0.04, and 0.05, and the probability P_l that there is a connection between a pair of nodes is equal to 0.2, 0.4, 0.6, 0.8, and 1.0. The networks were randomly generated and only connected networks were considered. Fifteen groups of five instances each were created, combining each possible pair of values for P_e and P_l.

We observed that set X is mostly made up of easy instances. This is due to two structural characteristics that are present in most of its instances. First, there are nodes incident to only one link whose connections are all routed through the same link. Second, there are weakly connected components, i.e. disjoint subsets of nodes that are connected by only one link. Therefore, all connections whose endnodes are in different weakly connected components must be routed through the same link. These characteristics may imply high lower bounds on the number of wavelengths necessary to establish the set of lightpath requests. For most of the instances in set X, a solution with this number of wavelengths can be easily found.

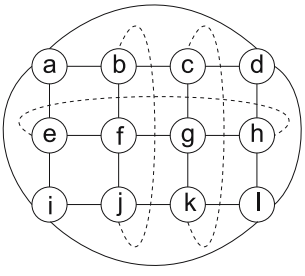

Fig. 2. Example of a 3×4 grid topology

Table 1. Description of test set W

Instance	Nodes	Links	Lightpaths	Maximum
Finland	31	51	930	1
EON	20	39	374	2
ATT	90	137	359	5
ATT2	71	175	4456	34
NSF.1	14	21	284	3
NSF.3	14	21	258	3
NSF.12	14	21	551	6
NSF.48	14	21	547	6
NSF2.1	14	22	284	3
NSF2.3	14	22	258	3
NSF2.12	14	22	551	6
NSF2.48	14	22	547	6

As an attempt to generate harder random instances, we propose the set Y. Networks in this set were randomly generated with the same number of nodes and the same values of P_e and P_l used for the instances in set X. However, we considered only networks whose node degrees are greater than or equal to 2 when P_e is equal to 0.04 and 0.05, and we restricted the diameter of the networks to 5, 6, and 7 for instances with P_e equal to 0.05, 0.04, and 0.03, respectively. As before, fifteen groups of five instances each were randomly generated, combining

Table 2. Average gaps and CPU times for implementations FF-RWA$^{\texttt{STD}}$, BF-RWA$^{\texttt{STD}}$, FFD-RWA$^{\texttt{STD}}$, and BFD-RWA$^{\texttt{STD}}$ for sets X, Y, Z, and W

	FF-RWA		FFD-RWA		BF-RWA		BFD-RWA	
Set	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
X	4.7%	0.60	1.9%	0.71	3.0%	1.33	1.2%	2.10
Y	23.1%	0.52	17.0%	0.58	13.9%	0.76	8.4%	1.07
Z	13.3%	0.89	9.7%	1.12	10.8%	1.10	7.0%	1.41
W	7.3%	0.10	6.3%	0.10	7.5%	0.11	7.1%	0.12

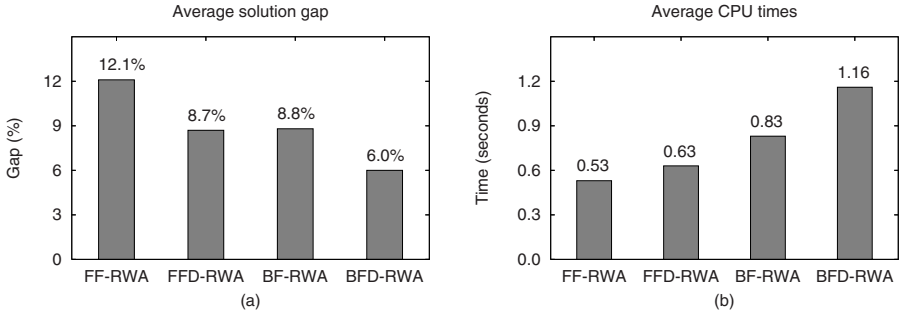

Fig. 3. (a) Average gaps and (b) CPU times for implementations FF-RWA$^{\texttt{STD}}$, BF-RWA$^{\texttt{STD}}$, FFD-RWA$^{\texttt{STD}}$, and BFD-RWA$^{\texttt{STD}}$ over all the 187 instances

each possible pair of values for P_e and P_l. The traffic matrices are the same used for the instances in set X.

Instances in test set Z are built on $n \times m$ grids embedded on the torus. Each node is connected only to its nearest four nodes. Figure 2 gives an example of a 3×4 grid. Five grid networks with approximately 100 nodes (10×10, 8×13, 6×17, 5×20, 4×25) were generated. For each of them, five traffic matrices are randomly generated with the probability P_l that there is a connection between a pair of nodes being equal to 0.2, 0.4, 0.6, 0.8, and 1.0.

Finally, set W is a collection of the most studied realistic instances in the literature, together with two new instances ATT and ATT2 whose topologies and traffic matrices resemble those of real telecommunication networks. The topology of the Finland network was obtained from [5] and its traffic matrix was the same used in [9]. Networks EON, NSF, and NSF2 and their respective traffic matrices were downloaded from [6]. The first three columns of Table 1 display the name, the number of nodes, and the number of links in each instance of set W, respectively. The total number of lightpaths and the maximum number of lightpaths starting from the same node are given in the fourth and fifth columns, respectively.

We denote by FF-RWA$^{\texttt{STD}}$, FF-RWA$^{\texttt{RRg}}$, FF-RWA$^{\texttt{RRt}}$, FF-RWA$^{\texttt{NRRg}}$, and FF-RWA$^{\texttt{NRRt}}$ the implementations of the heuristic FF-RWA using algorithms `STD`, `RRg`, `RRt`, `NRRg`, and `NRRt`, respectively. The same notation is extended to the

Table 3. Average CPU times for each heuristic using `NRRt`, `NRRg`, `RRt`, and `RRg`. Times are displayed as a percent deviation of the times using `STD`.

Heuristic	Set	`NRRt` (%)	`NRRg` (%)	`RRt` (%)	`RRg` (%)
FF-RWA	X	88.3	115.6	671.8	817.4
	Y	83.9	107.6	699.9	824.9
	Z	73.4	95.8	745.7	884.4
	W	98.4	119.0	423.8	497.6
FFD-RWA	X	84.8	112.4	562.2	690.1
	Y	81.1	103.9	626.0	739.1
	Z	62.0	79.7	595.2	699.1
	W	93.4	111.8	406.6	471.3
BF-RWA	X	50.3	67.5	254.2	333.2
	Y	67.8	88.1	400.7	496.2
	Z	63.4	83.4	568.1	685.7
	W	91.8	114.4	363.7	431.5
BFD-RWA	X	37.1	50.0	163.8	209.2
	Y	52.8	70.4	277.3	349.0
	Z	53.2	67.9	440.4	528.8
	W	88.7	107.5	344.0	407.5

other heuristics. The algorithms were coded in C++ and compiled with the GNU GCC version 4.0.3 with no compiler code optimization. The experiments were performed on a 2.8 GHz Pentium IV with 1 Gb of RAM memory using Linux Ubuntu 6.10. CPU times are reported in seconds. The quality of the heuristics is displayed as the gap (UB-LB)/LB between the cost UB of the solution provided by the heuristic and a lower bound LB for the cost of the optimal solution, which is calculated as suggested in [2].

The first experiments evaluate and compare the performance of FF-RWA$^{\texttt{STD}}$, FFD-RWA$^{\texttt{STD}}$, BF-RWA$^{\texttt{STD}}$, and BFD-RWA$^{\texttt{STD}}$ for the 187 instances in sets X, Y, Z, and W. Each heuristic was run five times with different seeds for the random number generator algorithm [11]. For each instance set, Table 2 displays the average gaps and CPU times for implementations FF-RWA$^{\texttt{STD}}$, FFD-RWA$^{\texttt{STD}}$, BF-RWA$^{\texttt{STD}}$, and BFD-RWA$^{\texttt{STD}}$, respectively. The average gaps for each heuristic over all instances are plotted in Figure 3a, while the average CPU times are plotted in Figure 3b.

The best of the five runs of BFD-RWA$^{\texttt{STD}}$ was optimal for 62 out of the 75 instances in set X and the average gap was 1.2%, which confirms the hypothesis that this set is mostly made up of easy instances. The average solution gaps for the instances in sets Y and Z proposed in this paper were greater than or equal to those in the other sets for all the heuristics, which indicates that the instances in sets Y and Z are harder than those in the literature.

Algorithm BFD-RWA$^{\texttt{STD}}$ found on average better results than the other heuristics for most of the instances tested. We notice in Figure 3a that the average gap observed for FFD-RWA (8.7%) is almost 50% larger than that corresponding to BFD-RWA (6.0%). The average gap observed for algorithm FFD-RWA$^{\texttt{STD}}$ was smaller than that for BFD-RWA$^{\texttt{STD}}$ exclusively for set W. However, this

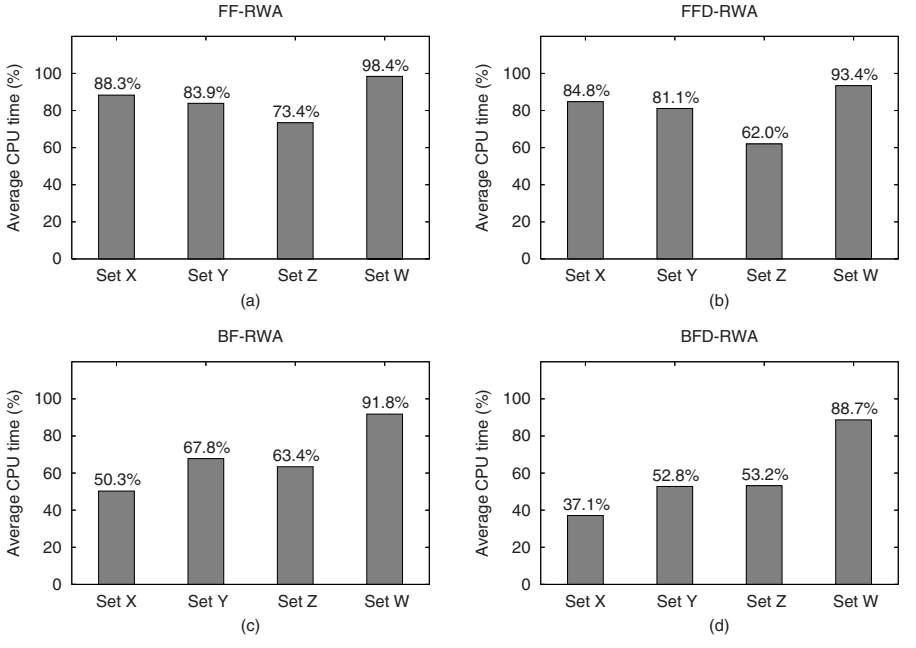

Fig. 4. Average CPU times of (a) FF-RWA$^{\texttt{NRRt}}$, (b) FFD-RWA$^{\texttt{NRRt}}$, (c) BF-RWA$^{\texttt{NRRt}}$, and (d) BFD-RWA$^{\texttt{NRRt}}$ for instance sets X, Y, Z, and W. Times are displayed as a percent deviation of the times using `STD`.

occurs because of the huge difference observed for instance ATT, where the FFD-RWA$^{\texttt{STD}}$ gap was 20.0% and the BFD-RWA$^{\texttt{STD}}$ gap was 32.0%. If we exclude this instance from set W, the average gap of FF-RWA$^{\texttt{STD}}$ would be 5.0% and that of BFD-RWA$^{\texttt{STD}}$ would be 4.9%.

As expected, CPU times of the best fit heuristics were greater than those of the first fit heuristics, because each iteration of FF-RWA$^{\texttt{STD}}$ and FFD-RWA$^{\texttt{STD}}$ stops at the first bin in which the lightpath can be routed with less than d arcs, while each iteration of BF-RWA$^{\texttt{STD}}$ and BFD-RWA$^{\texttt{STD}}$ scans all the bins looking for bins where the lightpath fits with the smallest number of arcs. Although solution gaps of BFD-RWA$^{\texttt{STD}}$ were on average smaller than those of the other heuristics, its running times were the longest. However, the maximum CPU times of BFD-RWA$^{\texttt{STD}}$ in set X was 10 seconds, which is much less than the 8 minutes reported for the implementation of [12] in the same set of instances and the same machine. The CPU times of BFD-RWA$^{\texttt{STD}}$ were always less than five seconds for instances in sets Y and Z and never greater than one second for those in set W.

The next experiments evaluate the performance of the heuristics FF-RWA, FFD-RWA, BF-RWA, and BFD-RWA using `NRRt`, `NRRg`, `RRt`, and `RRg`. The running times are compared with those using the respective standard implementation (`STD`). For each heuristic and each set of instances, Table 3 displays the average CPU times using `NRRt`, `NRRg`, `RRt`, and `RRg` as a percent deviation of the

times using STD (i.e. the times of NRRt, NRRg, RRt, and RRg are divided by the times of STD). Each version of each heuristic was run five times with different seeds for the random number generator algorithm [11]. For all heuristics and instance sets, the implementations using RRt and NRRt were faster than those using RRg and NRRg, respectively. This is due to the fact that updating of SP graphs is more expensive than updating SP trees, and the SP graphs were not dense enough to compensate the trade off. Due to the number of SP graphs that must be updated after an arc deletion, the implementations of FF-RWA, FFD-RWA, BF-RWA, and BFD-RWA using RRt and RRg were slower than their respective implementations using STD.

NRRt was the best algorithm for updating the distance matrix. The numerical results for heuristics FF-RWA$^{\texttt{NRRt}}$, FFD-RWA$^{\texttt{NRRt}}$, BF-RWA$^{\texttt{NRRt}}$, and BFD-RWA$^{\texttt{NRRt}}$ displayed in the third column of Table 3 are also plotted in Figure 4. The improvements observed in BF-RWA$^{\texttt{NRRt}}$ and BFD-RWA$^{\texttt{NRRt}}$ are greater than those observed in FF-RWA$^{\texttt{NRRt}}$ and FFD-RWA$^{\texttt{NRRt}}$, when compared with their respective standard implementations. This is due to the fact that the number of SP queries in BF-RWA and BFD-RWA is greater than in FF-RWA and FFD-RWA, while the number of updates is approximately the same. The heuristic that took more advantage of NRRt was BFD-RWA, whose times were shortened, on average, to almost one half of those of BFD-RWA$^{\texttt{STD}}$. The maximum running time of the BFD-RWA$^{\texttt{NRRt}}$ over all the 187 instances tested (including set X) was only 2.2 seconds, which is one quarter of the maximum running time of BFD-RWA$^{\texttt{STD}}$.

5 Concluding Remarks

This paper tackled the problem of routing and wavelength assignment in WDM optical networks. We proposed five different implementations of the best heuristics in the literature, as well as new testbed instances that allowed a precise comparison of the heuristics.

Computational experiments showed that BFD-RWA was the best heuristic for the instances tested. The new algorithms proposed in this paper shortened the average and maximum running times of BFD-RWA by 57% and 25%, respectively, with respect to those of the standard implementation. The maximum computation times of the best implementation of BFD-RWA was less than three seconds, while the times reported for the same heuristic in [12] were up to eight minutes on the same instances and the same Pentium IV 2.8 GHz computer.

References

1. Alvim, A.C.F., Ribeiro, C.C., Glover, F., Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. Journal of Heuristics 10, 205–229 (2004)
2. Bannerjee, D., Mukherjee, B.: Practical approach for routing and wavelength assignment in large wavelength routed optical networks. IEEE Journal on Selected Areas in Communications 14, 903–908 (1995)

3. Choi, J.S., Golmie, N., Lapeyrere, F., Mouveaux, F., Su, D.: A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case. In: Proceedings of the 7th International Conference on Optical Communication and Networks, Paris, pp. 1109–1115 (2000)
4. Erlebach, T., Jansen, K.: The complexity of path coloring and call scheduling. Theoretical Computer Science 255, 33–50 (2001)
5. Hyytiä, E., Virtamo, J.: Wavelength assignment and routing in WDM networks. In: Fourteenth Nordic Teletraffic Seminar, Copenhagen, pp. 31–40 (1998)
6. Jaumard, B.: Network and traffic data sets for optical network optimization (last visited on January 3th, 2008), `http://users.encs.concordia.ca/~bjaumard`
7. Li, G., Simha, R.: The partition coloring problem and its application to wavelength routing and assignment. In: Proceedings of the First Workshop on Optical Networks, Dallas (2000)
8. Manohar, P., Manjunath, D., Shevgaonkar, R.K.: Routing and wavelength assignment in optical networks from edge disjoint path algorithms. IEEE Communications Letters 5, 211–213 (2002)
9. Noronha, T.F., Ribeiro, C.C.: Routing and wavelength assignment by partition coloring. European Journal of Operational Research 171, 797–810 (2006)
10. Ramalingam, G., Reps, T.W.: An incremental algorithm for a generalization of the shortest-path problem. Journal of Algorithms 21, 267–305 (1996)
11. Schrage, L.: A more portable Fortran random number generator. ACM Transactions on Mathematical Software 5, 132–138 (1979)
12. Skorin-Kapov, N.: Routing and wavelength assignment in optical networks using bin packing based algorithms. European Journal of Operational Research 177, 1167–1179 (2007)
13. Zang, H., Jue, J.P., Mukherjee, B.: A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. Optical Networks Magazine 1, 47–60 (2000)