

Efficient Implementations of Heuristics for Routing and Wavelength Assignment

Thiago F. Noronha^{*}

Mauricio G. C. Resende^{**}

Celso C. Ribeiro^{***}

^{*} Dept. of Computer Science, Catholic U. of Rio de Janeiro, Brazil

^{**} Algorithms & Optimization Research Dept., AT&T Labs Research, USA

^{***} Dept. of Computer Science, U. Federal Fluminense, Brazil



at&t

Your world. Delivered.



Outline

- Problem description
- Related work
- Efficient implementations
- Computational experiments
- Conclusions



at&t

Your world. Delivered.



Routing and wavelength assignment (RWA)

- Given a graph with bidirectional links and a set of pairs of nodes that need to be connected.
- We want to route the set of connections (called lightpaths) and assign a wavelength to each of them.
- Two lightpaths may use the same wavelength, provided they do not share any common link.
- Connections whose paths share a common link in the network are assigned to different wavelengths (wavelength clash constraints).
- If no wavelength converters are available, the same wavelength must be assigned along the entire route (wavelength continuity constraints).

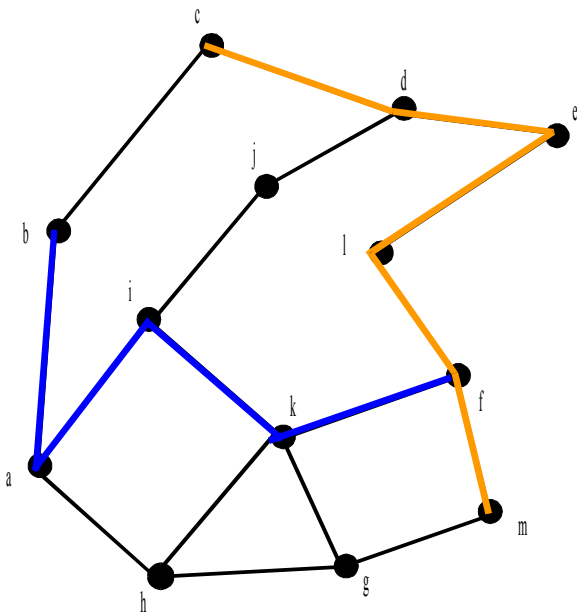


Routing and wavelength assignment (RWA)

- Variants of RWA are characterized by different optimization criteria, traffic patterns, and whether or not wavelength conversion is available.
- We consider the min-RWA offline variant:
 - Connection requirements are known beforehand.
 - No wavelength conversion is possible.
 - Objective is to minimize the number of wavelengths used for routing all connections.
 - Asymmetric traffic matrices and bidirectional links.
 - NP-hard (Erlebach and Jansen, 2001)

Routing and Wavelength Assignment (RWA)

Connections: (b ↔ f) (c ↔ m)



wavelength 1

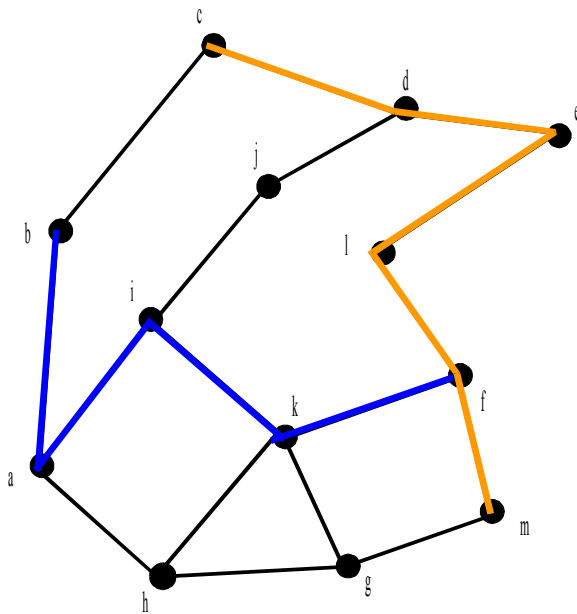


at&t

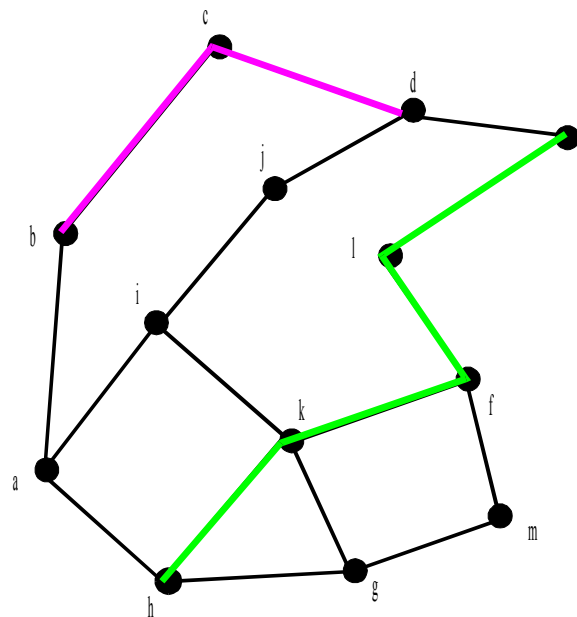
Your world. Delivered.

Routing and Wavelength Assignment (RWA)

Connections: (b ↔ f) (c ↔ m) (d ↔ b) (e ↔ h)



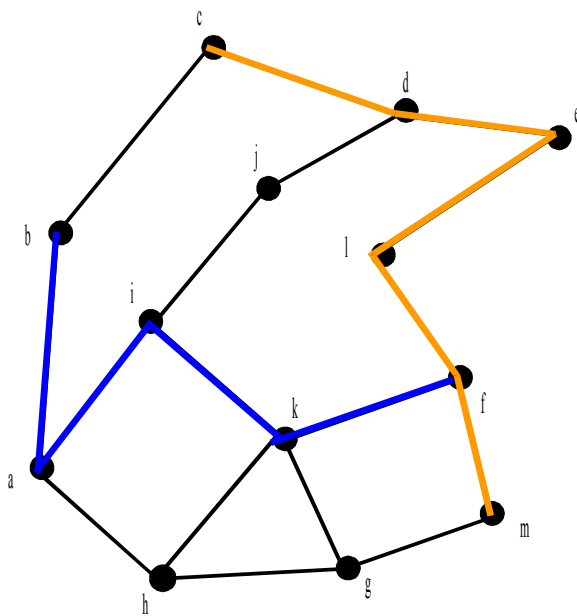
wavelength 1



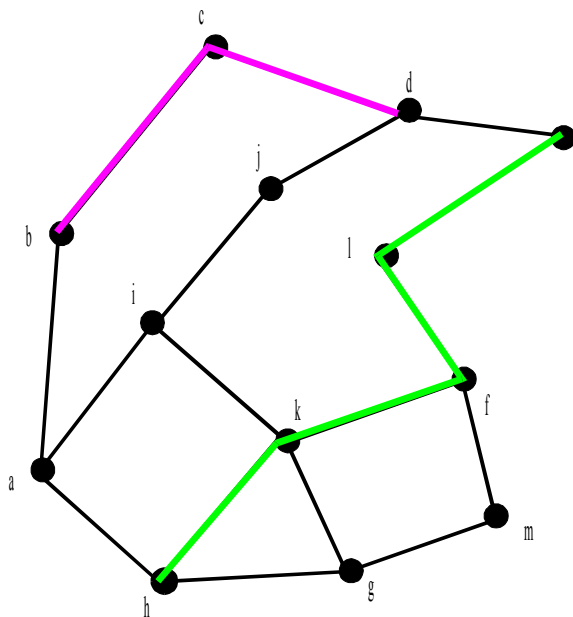
wavelength 2

Routing and Wavelength Assignment (RWA)

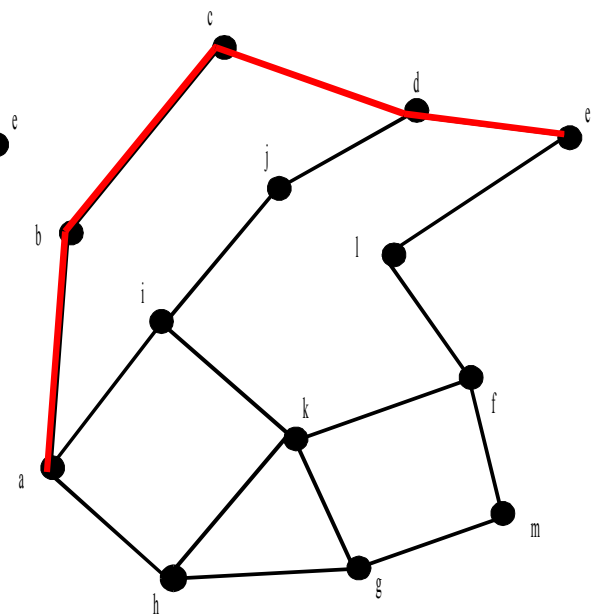
Connections: $(b \leftrightarrow f)$ $(c \leftrightarrow m)$ $(d \leftrightarrow b)$ $(e \leftrightarrow h)$ $(a \leftrightarrow e)$



wavelength 1



wavelength 2



wavelength 3



Related work (solve two subproblems separately)

- Banerjee and Mukherjee (1996)
 - Capacity-minimization multi-commodity flow formulation: lower bound.
 - Randomized rounding algorithm for routing.
 - Graph coloring for wavelength assignment.
- Hyytiä and Virtamo (1998)
 - Shortest path for routing.
 - Graph coloring for wavelength assignment.
- Li and Simha (2000)
 - K-shortest paths for routing.
 - Simple partition coloring heuristic for wavelength assignment.
- Noronha and Ribeiro (2006)
 - Edge-disjoint paths for routing.
 - Tabu search partition coloring for wavelength assignment.



Related work (solve two subproblems simultaneously)

- Manohar et al. (2002)
 - First work to handle both subproblems simultaneously.
 - Based on the BGAforEDP heuristic of Baveja and Srinivasan (2000) for the max edge disjoint path problem.
- Skorin-Kapov (2007)
 - Adaptations of bin packing heuristics.



Related work: Skorin-Kapov (2007)

- Associates the min-RWA with the bin packing problem.
 - Wavelengths correspond to bins.
 - The capacity of a bin is defined as its number of arcs.
 - The size of a connection is defined as the number of arcs in its shortest path.
- Developed RWA heuristics based on the following classical bin packing heuristics:
 - First Fit (FF)
 - Best Fit (BF)
 - First Fit Decreasing (FFD)
 - Best Fit Decreasing (BFD)



Related work: Skorin-Kapov (2007)

- FF-RWA:
 - Input:
 - A directed graph G representing the network topology.
 - A set T of connection requests.
 - The value d of the maximum number of arcs in each route. It is set to be the maximum of the square root of the number of links in the network and the diameter of G .
 - Starts with only one copy of G (called G_1).
 - Connections are selected at random and routed (with least hops) on the lowest indexed bin G_i in which there is room with less than d arcs.
 - The connection is assigned wavelength i , and the arcs along path are deleted from G_i .
 - If no existing bin can accommodate the connection, a new bin is created.



Related work: Skorin-Kapov (2007)

- BF-RWA:
 - Same input of FF-RWA.
 - Starts with only one copy of G (called G_1).
 - Connection is selected at random and routed on the bin G_i with smallest number of arcs in which the connection fits.
 - The connection is assigned wavelength i , and the arcs along path are deleted from G_i .
 - If no existing bin can accommodate the connection with less than d arcs, a new bin is created.



Related work: Skorin-Kapov (2007)

- FFD-RWA:
 - Proceeds as in FF-RWA, except that connections are selected according to non-increasing order of the lengths of their shortest paths in G .
 - Ties are randomly broken.
- BFD-RWA:
 - Proceeds as in BF-RWA, except that connections are selected according to non-increasing order of the lengths of their shortest paths in G .
 - Ties are randomly broken.



Related work: Skorin-Kapov (2007)

- FFD-RWA and BFD-RWA were the first algorithms in the literature to tackle instances with up to 100 nodes and 10,000 connection requests.
- Test set mostly consisted of “easy” instances.
 - No clear definition of the performance of the heuristics.
- Computation times reported were up to **8 minutes** (Pentium IV 2.8 Ghz) on the largest instances.



Implementation issues

- All heuristics reimplemented using the same graph data structure.
- Most important operations are **ShortestPath** and **DeleteArcs**.
- Much more shortest path queries than path/wavelength assignments.
- Standard Implementation
 - Since we are interested only in hop-count shortest paths, they can be calculated using Breadth First Search (BFS) in $O(m)$ time, where m is the number of network links.
 - Shortest path queries are performed in $O(m)$ time.
 - Arcs can be deleted in $O(1)$ time using doubly linked lists to represent the adjacency matrix.
 - Keeps **only the forward star** of each vertex for each bin in the solution.



Implementation issues

- RR_G implementation
 - Based on the algorithm of Ramalingam and Reps (1996) for **dynamic graph shortest paths**.
 - Updates the graph of shortest paths from **one vertex** after increment or decrement in the weights of the arcs.
 - Arcs can be deleted by increasing their weights to infinity.
 - After an arc deletion, it identifies the affected nodes and then proceeds as Dijkstra-like algorithm.
 - Can be implemented here in $O(m)$ amortized time using a bucket.
 - Keeps the **forward star**, the **backward star**, **the set of parents** for each vertex, and the **distance matrix** for each bin in the solution.



Implementation issues

- RR_G implementation (cont.)
 - For **each bin** in the solution, keeps n **graphs of shortest paths** starting from each node in the network.
 - Shortest path queries are performed in **constant time**.
 - Updates might be very expensive, since many shortest path graphs may change after an arc deletion.



Implementation issues

- RR_T implementation

- Based on the algorithm of Ramalingam and Reps (1996) for **dynamic trees of shortest paths**.
 - Updates a tree of shortest paths from **one vertex** after increment or decrement in the weights of the arcs.
 - Arcs can be deleted by increasing their weights to infinity.
 - After an arc deletion, it identifies the affected nodes and then proceeds as Dijkstra-like algorithm.
 - Can be implemented here in $O(m)$ amortized time using a bucket.
- Keeps similar structures as RR_G except that **only one parent** is kept for each vertex, instead of the set of all parents needed in the shortest path graph.



Implementation issues

- NRR_G and NRR_T implementations
 - They maintain the same data structure as RR_G and RR_T , respectively, but do not update them dynamically.
 - Therefore, the distance matrix gives a lower bound on the shortest path between two nodes in $O(1)$ time.
 - If the lower bound is larger than d , the correct distance is not necessary, where d is the maximum number of arcs in any shortest path (parameter of the heuristics).
 - The correct distance can be calculated by querying the shortest path.
 - If no arc is deleted in the shortest path, the distance can be retrieved from the graph (resp. tree) of the sink node in $O(d)$ time.
 - Otherwise, only one graph (resp. tree) of shortest paths is updated from scratch in $O(m)$ time and the shortest path is queried again.
 - Only the distance matrix is updated after an arc deletion.



Implementation issues

- NRR_G and NRR_T implementations
 - They maintain the same data structure as RR_G and RR_T , respectively, but do not update them dynamically.
 - Therefore, the distance matrix gives a lower bound on the shortest path between two nodes in $O(1)$ time.
 - If the lower bound is larger than d , the correct distance is not necessary, where d is the maximum number of arcs in any shortest path (parameter of the heuristics).
 - The correct distance can be calculated by querying the shortest path.
 - If no arc is deleted in the shortest path, the distance can be retrieved from the graph (resp. tree) of the sink node in $O(d)$ time.
 - Otherwise, only one graph (resp. tree) of shortest paths is updated from scratch in $O(m)$ time and the shortest path is queried again.
 - Only the distance matrix is updated after an arc deletion.

Worst case is $O(m)$ though many updates are done in $O(1)$ or $O(d)$.

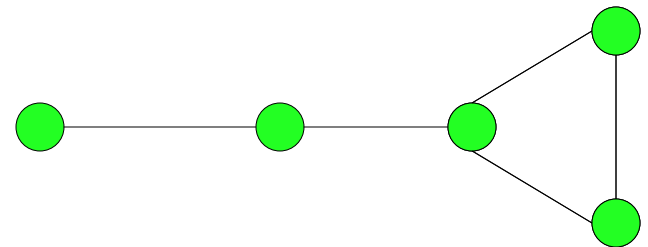
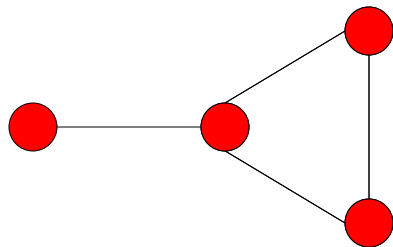
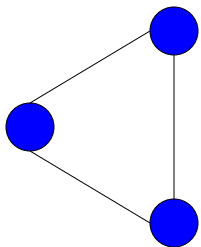


Computational experiments: conditions

- The five implementations of FF-RWA, BF-RWA, FFD-RWA, and BFD-RWA were coded in C++ 4.0.3.
- We denote by $\text{FF-RWA}^{\text{STD}}$, $\text{BF-RWA}^{\text{STD}}$, $\text{FFD-RWA}^{\text{STD}}$, and $\text{BFD-RWA}^{\text{STD}}$ the standard implementations of the four heuristic studied.
 - The same terminology was applied to RR_G , NRR_T , NRR_G , and NRR_T implementations.
- No compiling code optimization was used.
- Computational experiments were performed on a Pentium 2.8 Ghz with 1 Gbytes.
- Heuristics evaluated on 187 test instances divided into four sets.

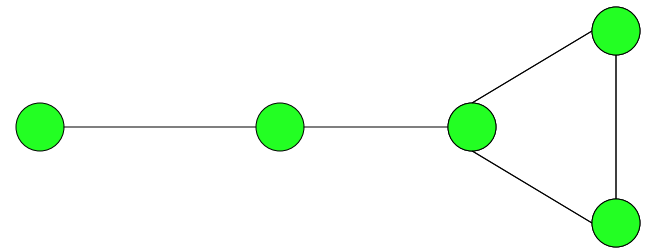
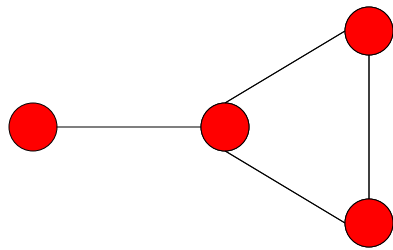
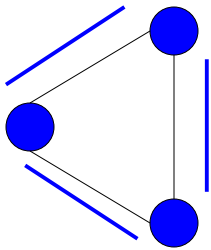
Computational experiments: test instances

- Set A: Skorin-Kapov (2007)
 - 75 instances with 100 nodes.
 - Randomly generated with edge probability equal to 0.03, 0.04, and 0.05.
 - Traffic matrices are randomly generated with probability of existing a connection between a pair of vertices equal to 0.2, 0.4, 0.6, 0.8, and 1.0.
- Mostly made up of easy instances because of network structure (nodes with degree one and components connected by single edge).



Computational experiments: test instances

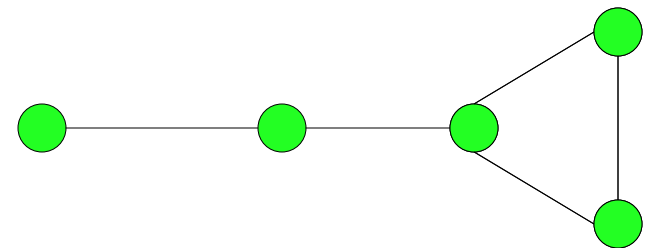
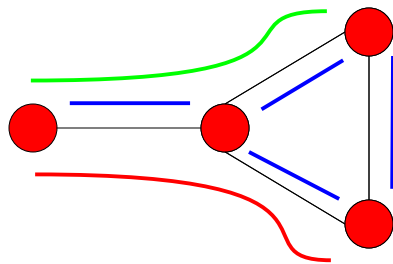
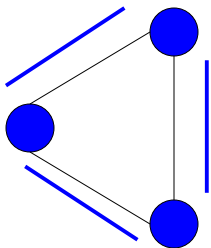
- Set A: Skorin-Kapov (2007)
 - 75 instances with 100 nodes.
 - Randomly generated with edge probability equal to 0.03, 0.04, 0.05.
 - Traffic matrices are randomly generated with probability of existing a connection between a pair of vertices equal to 0.2, 0.4, 0.6, 0.8, and 1.0.
- Mostly made up of easy instances because of network structure (nodes with degree one and components connected by single edge).



one wavelength

Computational experiments: test instances

- Set A: Skorin-Kapov (2007)
 - 75 instances with 100 nodes.
 - Randomly generated with edge probability equal to 0.03, 0.04, 0.05.
 - Traffic matrices are randomly generated with probability of existing a connection between a pair of vertices equal to 0.2, 0.4, 0.6, 0.8, and 1.0.
- Mostly made up of easy instances because of network structure (nodes with degree one and components connected by single edge).



one wavelength

three wavelengths

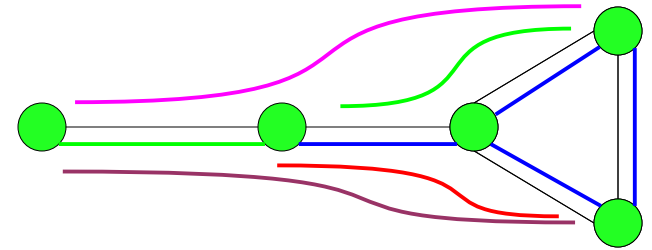
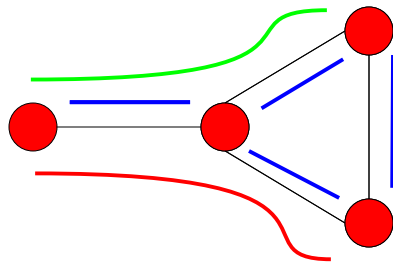
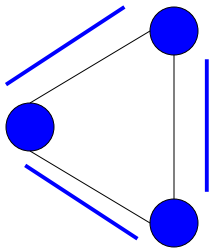


at&t

Your world. Delivered.

Computational experiments: test instances

- Set A: Skorin-Kapov (2007)
 - 75 instances with 100 nodes.
 - Randomly generated with edge probability equal to 0.03, 0.04, 0.05.
 - Traffic matrices are randomly generated with probability of existing a connection between a pair of vertices equal to 0.2, 0.4, 0.6, 0.8, and 1.0.
- Mostly made up of easy instances because of network structure (nodes with degree one and components connected by single edge).



one wavelength

three wavelengths

five wavelengths

high lower bounds: solutions easy to find



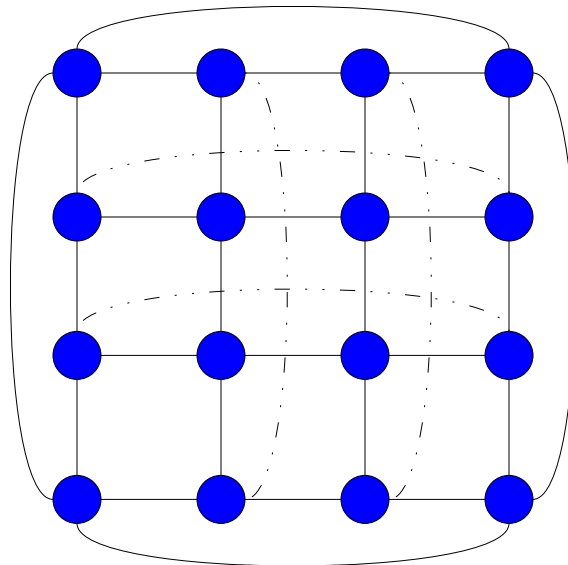
Computational experiments: test instances

- Set B: similar to set A, but harder (larger gaps)
 - 75 instances with 100 nodes.
 - Randomly generated with edge probability equal to 0.03, 0.04, and 0.05.
 - Same traffic matrices of Skorin-Kapov (2007).
 - **Vertex degrees and graph diameter are restricted:**
 - Vertices in instances with edge probability 0.04 and 0.05 have degree at least 2.
 - Diameters are restricted to 5, 6, and 7 for instances with edge probability equal to 0.05, 0.04, and 0.03, respectively.



Computational experiments: test instances

- Set C: harder, due to symmetries
 - 25 grid networks with approximately 100 nodes (10x10, 8x13, 6x17, 5x20, 4x25)
 - All nodes have degree 4.
 - Traffic matrices randomly generated with the probability of existing a connection between a pair of vertices being equal to 0.2, 0.4, 0.6, 0.8, and 1.0.





Computational experiments: test instances

- Set D: realistic instances

Instance	Nodes	Arcs	Connec.	Max connec.
Finland	31	102	930	1
EON	20	78	374	2
ATT	90	274	359	5
ATT2	71	350	2918	34
NSF.a	14	42	284	3
NSF.b	14	42	258	3
NSF.c	14	42	551	6
NSF.d	14	42	547	6
NSF2.a	14	44	284	3
NSF2.b	14	44	258	3
NSF2.c	14	44	551	6
NSF2.d	14	44	547	6



at&t

Your world. Delivered.



Computational experiments: solution quality and CPU times with the standard implementations

- Lower bounds (LB)
 - The linear programming relaxation of a min-RWA formulation is too expensive to compute because of the large number of variables.
 - Linear programming relaxation of the capacity-minimization multi-commodity flow formulation of Banerjee Mukherjee (1996) gives a good lower bound.
 - min-RWA without the wavelength continuity constraints.
- Upper bounds (UB)
 - Heuristics FF-RWA^{STD}, BF-RWA^{STD}, FFD-RWA^{STD}, and BFD-RWA^{STD}.
- Relative gaps: $(UB-LB)/LB$

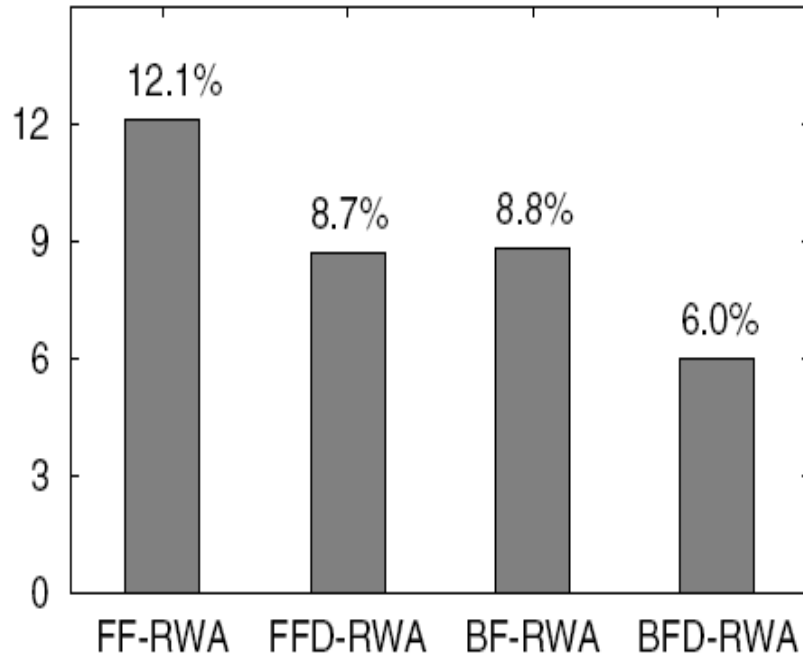


at&t

Your world. Delivered.

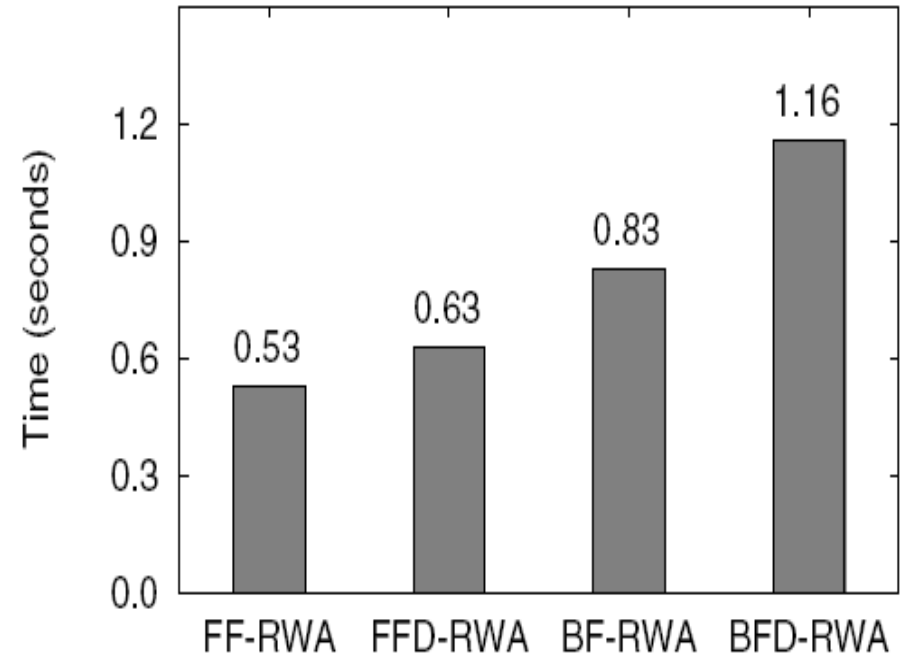
Computational experiments: solution quality and CPU times with the **standard** implementations

Average solution gap



(a)

Average CPU times



(b)

five runs of each heuristic
on each instance

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

Computational experiments: speed up with the **dynamic** implementations

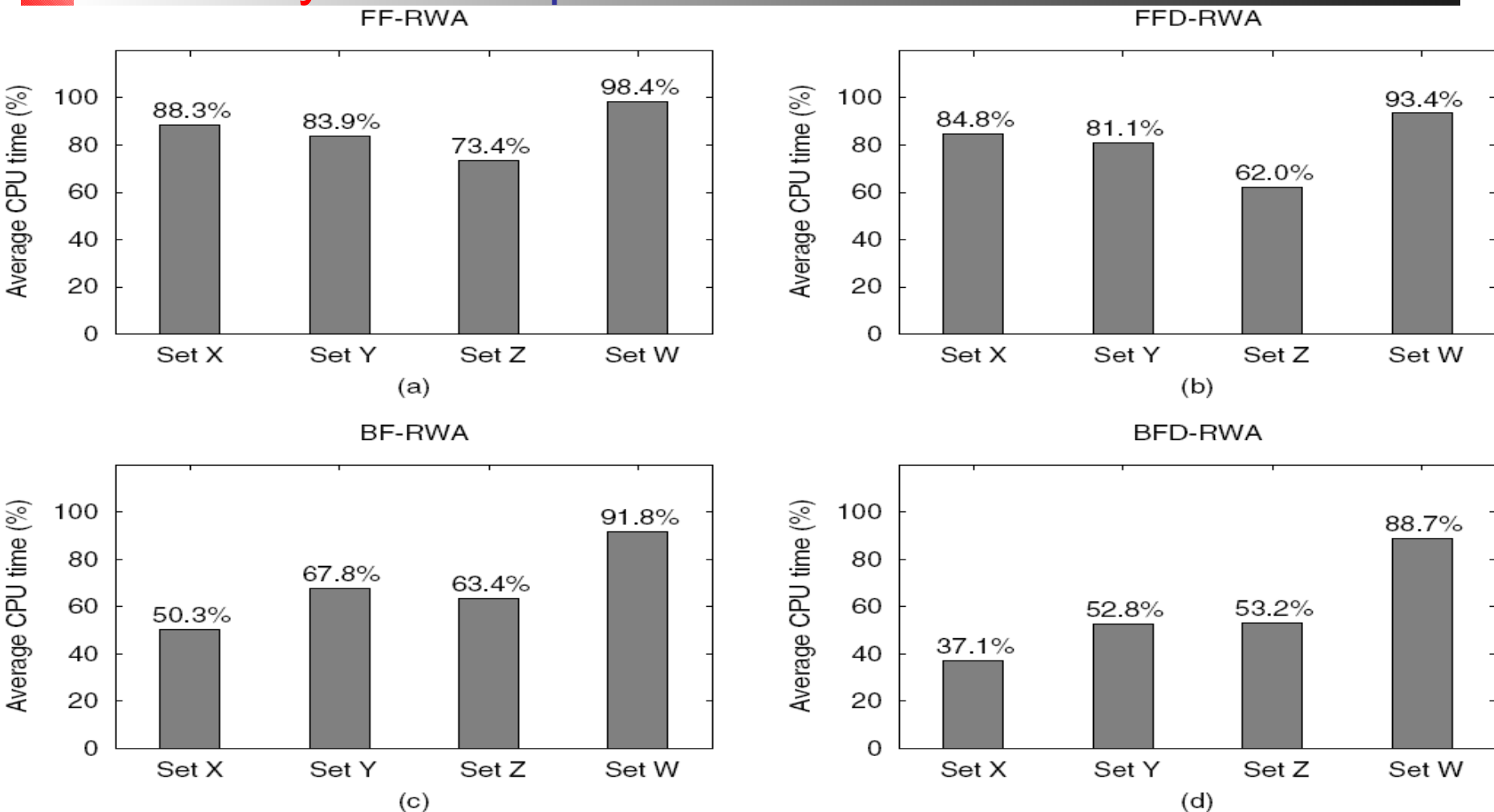


Fig. 4. Average CPU times of (a) FF-RWA^{NRRt}, (b) FFD-RWA^{NRRt}, (c) BF-RWA^{NRRt}, and (d) BFD-RWA^{NRRt} for instance sets X, Y, Z, and W. Times are displayed as a percent deviation of the times using STD.



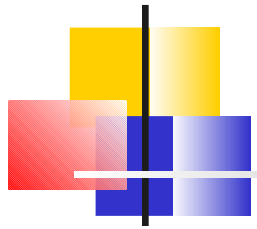
Concluding Remarks

- 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;
 - 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.



Concluding Remarks

- NRR_T implementation shortened the average and maximum running times of BFD-RWA by **57%** and **25%**, respectively, with respect to those of our standard implementation.
- The maximum computation times of the NRR_T implementation of BFD-RWA was **less than three seconds**, while the times reported for the same heuristic in Skorin-Kapov (2007) were **up to eight minutes** on the same instances and the same Pentium IV 2.8 GHz computer.



The End

These slides and all of my papers cited in this talk
can be downloaded from my homepage:
<http://mauricioresende.com>



at&t

Your world. Delivered.