Arc Perturbation Algorithms
for Optical Network Design

Zbigniew R. Bogdanowicz

Armament Research, Development and Engineering Center
Building 95, Picatinny
New Jersey 07806, U.S.A.
E-mail: zbogdan@pica.army.mil

Abstract

Today’s network planners are faced with the challenge to design highly complex optical networks. In this paper we describe and study the performance of two new combinatorial algorithms based on the local perturbation framework to optimize the real-world optical networks. These new heuristic algorithms were implemented in an optical network design tool as a commercial product, and they are named $S$-arc and $Sd$-arc respectively.

Mathematics Subject Classification: 05C85

Keywords: Graph algorithms, optical network, optimization, heuristics.

1 Introduction

In this paper we introduce and study new combinatorial algorithms based on the perturbation framework to optimize real-world optical networks. Optical networks based on the wavelength division multiplexing (WDM) technology [5] are increasingly being deployed in military and commercial carrier networks. In a generic mesh topology, an optical network consists of optical switches interconnected by WDM fiber links.

An optical switch can switch an entire wavelength channel from input ports to output ports. In general optical switches can be purely optical, or electronic, or a combination of optical and electronic depending on the degree to which signals remain in the optical or electronic domains within the switch. In this paper it is assumed that the optical switches allow for full wavelength conversion, i.e., any wavelength channel on an input port can be switched to any wavelength channels on output ports.
A WDM fiber link consists of a wavelength multiplexer/demultiplexer (MUX) that either multiplexes or demultiplexes individual wavelength channels onto the fiber. Optical amplifiers (AMPs) along the fiber route amplify the optical signals on all the wavelengths. A regenerator (REGEN) along the fiber route performs the regeneration (i.e., regeneration, retiming, and reshaping) of an optical signal.

Optical network provides the basic service by provisioning the wavelength circuits called lightpaths between the client network elements (NEs). An NE is a device such as an internet protocol router, an asynchronous transfer mode switch, or an add-drop multiplexer that connects to the optical switches via drop ports of the optical switch. A lightpath is an optical channel that originates at an access port of an ingress optical switch, traverses multiple optical switches and fiber links, and terminates at an access port at an egress optical switch. Optical signals are typically SONET/SDH formatted, of OC-48 (2.5 Gigabits per second) or OC-192 (10 Gigabits per second) rates, and OC-768 (40 Gigabits per second) are expected to be used in the future.

In this work we consider the following three types of lightpaths: a) unprotected, b) 1+1 protected, and c) mesh-restored [8]. An unprotected lightpath is not protected upon a failure of any resource (e.g., optical switch, fiber-link, transceivers, etc.) along the lightpath route. A 1+1 protected lightpath has a working route and a diversely routed backup route. The wavelength channels for the working and diversely routed backup route are dedicated for a 1+1 protected lightpath. A 1+1 protected lightpath can recover from any single resource failure on its working route. A mesh-restored lightpath has a working route and a diversely routed backup route. The wavelength channels on the working route of the mesh-restored lightpath are dedicated for that lightpath and carry user traffic under normal operating conditions. The wavelength channels on the backup route for the mesh-restored lightpath are shared among different mesh-restored lightpaths. Wavelength channels are shared in such a way that every mesh-restored lightpath can recover from any single resource failure on its working route [2].

A carrier deploying an optical network typically has fiber routes between the different locations (nodes) in the network. The designer has to decide how to efficiently allocate capacity in the network in terms of the number of wavelength channels on links, number of ports on optical switches, etc. The designer also has a forecasted demand in terms of the number of lightpaths of each type and bandwidth between each pair of cities. So, the challenge for the designer is to find an efficient way to route these forecasted lightpath demands.
2 Problem Description

The problem of designing an optical network can be described based on the given lightpath demands, existing optical infrastructure, and the objective of optimization. A lightpath demand is described by the following parameters.

1. Source node
2. Destination node
3. Rate (e.g., OC-48, OC-192)
4. Protection type (unprotected, 1+1, mesh-restored)

Each lightpath demand is routed in the optical network from a source node to a destination node. For protected demands, a backup route is decided as well. The routes for lightpaths (working and backup) are constrained by the engineering rules that arise as a result of the characteristics of switches and WDM fiber links. Because of these constraints, and because of diversity requirements for backup routes, the working routes of the lightpaths are not always the shortest in the optimized network.

The problem of designing an optical network can be formulated as follows. 
Given the lightpath demands and existing fiber infrastructure/topology (i.e., WDM fiber links interconnecting fiber nodes), find a feasible set of the corresponding lightpath routes that minimizes the total cost of network-in-use. In most cases all lightpaths can be routed. However, in a few extreme cases when it’s not possible, we want to achieve the following:

1. Route as many lightpaths as possible, and
2. Obtain the least cost solution.

Network-in-use can be determined as follows. Given the routes for each of the lightpaths, we can determine the number of channels required on each link, and the number of ports required at each node. The capacity requirement for a WDM fiber link is the number of lightpath demands (working and backup) that are routed over that link. The number of ports required at a node can also be computed from the number of lightpaths (working and backup) that are routed through that node. Given a certain configuration of the optical network in terms of the deployed equipment, the capital cost of that network configuration can be calculated from the capital costs of the different utilized components that comprise that network configuration. Such an assignment of costs to used equipment is called a cost model.

There are two variations of the optical network design problem. The first case is design from scratch, or greenfield design, and the second case is incremental design. In greenfield design, either the existing lightpath routes are
discarded or the optical network is not yet deployed. In incremental design, the optical network is already deployed, and it is carrying the lightpath demands. In this case, the existing lightpath routes are preserved and only new lightpath demands are routed.

The solution to the network design problem consists of routes (working and backup) for the lightpath demands, and inventories at nodes and links. As mentioned before, the routes for lightpaths are restricted by practical constraints imposed by the characteristics of the equipment. For example, the choice of the type of an optical switch limits the number of ports supported by the switch. This in turn restricts the number of lightpaths that can be routed on a WDM fiber link. So, a solution is derived from the capacitated network design problem.

We also distinguish between the restricted-link (RL) and unrestricted-link (UL) optimization (Tables I). In the RL optimization, an algorithm utilizes the existing fiber routes. In the UL optimization, an optimization algorithm allows to add a certain number of new fiber routes to the topology from a given candidate set of fiber routes. These new fiber routes usually correspond to a different service provider.

3 Arc Perturbation Algorithms

It is well known that the capacitated optimization problem described in Section 2 is related to the capacitated multicommodity flow problem, which is \textit{NP-complete} [6,7]. It’s true even for the case of all unprotected lightpaths. Therefore, we use intelligent heuristic approach to solve it, which is based on a general perturbation framework.

In the perturbation framework, we iteratively evaluate different network designs. In each iteration of the algorithm, given lightpath demands are routed using the shortest path routing algorithms. Selection of a shortest path algorithm is done per lightpath, and it depends on a lightpath type. As many lightpath demands are routed as possible. Then the cost of the design is evaluated using the cost model described in previous section. If the cost of the design is the best found for given number of routed lightpaths, then it is saved. In the subsequent iterations of the algorithm, the network is perturbed slightly by changing the characteristics of the topology of the network. Such perturbations are designed to allow the algorithm to escape from the local optima of the solution space, and converge towards the global optimal solution.

The routing algorithms route each demand one-by-one in a sequence. For unprotected lightpaths, we use Dijkstra’s algorithm [1]. For 1+1 protected lightpaths, we use a version of the Suurballe’s algorithm based on Bellman-Ford algorithm [2]. For mesh-restored lightpaths we use a simple heuristic algorithm.
Arc Perturbation Algorithms

The perturbation framework is general in the sense that it allows any routing algorithm to be used. Once the lightpaths are routed, the cost of the routing is determined using a cost model. A solution is deemed to be better than the best found if it has lesser number of unrouted lightpaths. If the number of unrouted lightpaths is the same, then a solution is deemed better if it has a lower cost.

We model optical infrastructure with digraph $G$ on $m$ arcs and $n$ vertices. Each WDM fiber link corresponds to a pair of opposite arcs in $G$. The cost of arcs in $G$ might correspond to a network cost model initially, but it doesn’t have to. In particular, the cost of arcs in $G$ is used by the routing algorithms (such as Dijkstra’s) to route the lightpaths over the shortest paths. We allow multiple pairs of opposite arcs between the vertices, but we do not allow loops in $G$. The pairs of opposite arcs correspond to the existing fiber links as well as to the potential new fiber links that we might consider to add later. Let $L$ be a set of $d$ lightpath demands, and $C$ be a cost matrix defining arc costs. The input to all our heuristic algorithms presented in upcoming Sections 3.1, 3.2 and 3.3 consists of $G$, $L$, $C$, and a cost model.

The basic perturbation step of all the algorithms presented in this paper consists of either encouraging or discouraging an arc (or two adjacent arcs) in $G$ from being included in a lightpath route. For example in Single-arc perturbation (Section 3.1), in each iteration an arc is set to either desirable or undesirable state. In Double-arc perturbation (Section 3.2), in each iteration the two arcs that are incident at a vertex are set to all combinations of states, i.e., (desirable, desirable), (desirable, undesirable), (undesirable, desirable), (undesirable, undesirable). Even though the lightpath paths are bidirectional, every shortest path algorithm searches for a shortest path from its source to its destination in directional manner. Hence, changing a cost of arc $(a, b)$ rather than $(b, a)$ between vertices $a$, $b$ in $G$ has different impact on finding the shortest paths.

To present our optimization algorithms we also need the following two definitions. Let low cost (i.e., desirable state of arc) be defined as $c_L = \min_{i,j} \frac{c_{i,j}}{n}$, and high cost (i.e., undesirable state of arc) be defined as $c_H = \max_{i,j} c_{i,j}n$.

During the routing of the mesh-restored lightpaths the arc costs (initialized from $C = [c_{i,j}]$) are considered. We propose to initialize these costs for optimization to the same amount for each arc and equal $n + 1$, i.e., $c_{i,j} = n + 1$. The arc costs will be changing, however, as the iterations of the optimization progresses. During the network-in-use cost evaluation (after each iteration, which corresponds to the routing of all lightpath demands) the ”real costs” of links (i.e., pairs of arcs) are considered based on a given cost model. This cost model will remain unchanged during the optimization. Our algorithms are built of two components: (1) Single-arc, and (2) Double-arc.
3.1 **Single-arc Algorithm**

Single-arc optimization can be executed in 12 steps as follows.

---

Step 1. Route all lightpath demands over the shortest paths.
Step 2. Set best-solution = current-solution.
Step 3. For every arc \( a \) in \( G \) execute Steps 4 through 7, and then go to Step 8.
Step 4. Save cost of \( a \).
Step 5. Set cost of \( a \) to \( c_L \).
Step 6. Route all lightpath demands over shortest cost paths.
Step 7. If best-solution not worse than current solution then restore saved cost of current arc \( a \).
   Otherwise, set best-solution = current-solution.
Step 8. For every arc \( a \) in \( G \) execute Steps 9 through 12, and then STOP.
Step 9. Save cost of \( a \).
Step 10. Set cost of \( a \) to \( c_H \).
Step 11. Route all lightpath demands over shortest cost paths.
Step 12. If best-solution not worse than current-solution then restore cost of current arc \( a \).
   Otherwise, set best-solution = current-solution.
---

At each iteration of Single-arc algorithm an arc is set to either desirable or undesirable state, and then all lightpaths are routed over the shortest paths. The execution time of routing of a single \( i \)’th lightpath over a shortest path will vary depending on its type (e.g., Dijkstra’s algorithm for unprotected lightpath could be executed in \( O(m + n \log n) \) [1]), but it’s always polynomial, say \( T_i(n,m) \). Let \( T(n,m) \) be the worst time \( T_i(n,m) \). Since every arc in \( G \) is set first to the desirable and then to undesirable state, then the worst time complexity of single-arc algorithm is \( O(mdT(n,m)) \).

3.2 **Double-arc Algorithm**

Double-arc optimization can be executed in 19 steps as follows.

---

Step 1. Route all lightpath demands over shortest paths.
Step 2. Set best-solution = current-solution.
Step 3. For every two adjacent arcs \( a, b \) in \( G \) execute steps 4 through 19, and then STOP.
Step 4. Save current cost of both arcs \( a, b \).
Step 5. Set cost of both arcs $a$, $b$ to cost $c_L$.
Step 6. Route lightpath demands over the shortest paths.
Step 7. If $best\text{-}solution$ not worse than $current\text{-}solution$ then restore saved costs of both arcs $a$, $b$.
    Otherwise, set $best\text{-}solution = current\text{-}solution$.
Step 8. Save current cost of both arcs $a$, $b$.
Step 9. Set cost of the first arc $a$ to $c_L$ and of the second arc $b$ to $c_H$.
Step 10. Route lightpath demands over the shortest paths.
Step 11. If $best\text{-}solution$ not worse than $current\text{-}solution$ then restore saved costs of both arcs $a$, $b$.
    Otherwise, set $best\text{-}solution = current\text{-}solution$.
Step 12. Save current cost of both arcs $a$, $b$.
Step 13. Set cost of the first arc $a$ to $c_H$ and of the second arc $b$ to $c_L$.
Step 14. Route lightpath demands over the shortest paths.
Step 15. If $best\text{-}solution$ not worse than $current\text{-}solution$ then restore saved costs of both arcs $a$, $b$.
    Otherwise, set $best\text{-}solution = current\text{-}solution$.
Step 16. Save current cost of both arcs $a$, $b$.
Step 17. Set cost of both arcs $a$, $b$ to $c_H$.
Step 18. Route lightpath demands over the shortest paths.
Step 19. If $best\text{-}solution$ not worse than $current\text{-}solution$ then restore saved costs of both arcs $a$, $b$.
    Otherwise, set $best\text{-}solution = current\text{-}solution$.

At each iteration of Double-arc algorithm, two adjacent arcs are set to every combination of desirable/undesirable states, and then all lightpaths are routed over the shortest paths. Because of that, the execution time does not depend strictly on $m$, $d$, $T(n, m)$ as was the case for Single-arc algorithm, but it also depends on a specific topology of $G$. However, in a typical optical infrastructure the average number of WDM fiber links incident to an optical switch is between two and four [8], which corresponds to four through eight arcs per vertex. In particular, for a regular graph $G$ of order 3 (i.e., with six arcs incident to each vertex), there are $\frac{18}{6}n$ possible arc pairs, with four favorable/unfavorable state combinations per pair. Hence, for such sparse optical networks the worst time complexity of Double-arc algorithm is $O(n^2dT(n, m))$.

3.3 Optimization Based on Single-arc and Double-arc

When a cost of any arc changes after a Single-arc execution, it might affect the routing of all lightpath demands. Hence, to assure the best outcome of Single-arc we might continue executing it in the loop as long as $best\text{-}solution$
continues to be improved by Single-arc. This method we name S-arc, and it is as follows.

\begin{itemize}
  \item Step 1. Set \texttt{best-solution} = \texttt{worst-solution}.
  \item Step 2. Execute Single-arc algorithm.
  \item Step 3. If \texttt{best-solution} improved in Step 2 then return to Step 2. Otherwise, STOP.
\end{itemize}

Similarly, the solution can be further improved by continuing optimization with \textit{Double-arc} as long as the \texttt{best-solution} keeps improving. We named this method \textit{Sd-arc}, and it is as follows.

\begin{itemize}
  \item Step 1. Set \texttt{best-solution} = \texttt{worst-solution}.
  \item Step 2. Execute S-arc algorithm.
  \item Step 3. Execute Double-arc algorithm.
  \item Step 4. If \texttt{best-solution} improved in Step 3 then return to Step 3. Otherwise, STOP.
\end{itemize}

4 Computational Results and Conclusions

We implemented S-arc and Sd-arc algorithms on a Pentium III 733 MHz Windows XP PC and executed optimization for twelve test networks (Table I). We provided enough bandwidth capacity in all cases, so all lightpath demands could be routed for fair cost comparison. In particular, networks 2, 4, 7, 9, 11 were derived from networks 1, 3, 6, 8, 10 respectively by adding approximately 10\% additional WDM fiber links, and keeping original lightpath demands unchanged. For these derived networks UL optimization was executed, and in all cases it improved the optimization result in respect to the corresponding RL executions. This in turn suggests the good quality of the S-arc and Sd-arc solutions. Furthermore, for several special cases of small and moderate size networks, we actually were able to compare S-arc and Sd-arc algorithms with the optimal solutions based on the results in [3].

Based on these comparisons we estimate that the average optimization cost after S-arc execution is within 5\% of optimal cost. For Sd-arc optimization the average optimization cost should be even better. These conclusions are supported by the results in Table I due to the fact that the Sd-arc algorithm
improves the optimization from $S$-arc by no more than 3%, and that as described above $UL$ produced better results than $RL$ in all test cases.

Note, $S$-arc alone can improve the initial cost of network by up to 30% (Net8 and Net 9 in Table I)), potentially saving millions of dollars in a typical optical network. For $S$-arc algorithm the running times ranged between several seconds and a few hours. For $Sd$-arc algorithm the running times ranged between tens of seconds and tens of hours.

<table>
<thead>
<tr>
<th>Case</th>
<th>$n$</th>
<th>$m$</th>
<th>$d$</th>
<th>Opt. Type</th>
<th>Initial Cost</th>
<th>Cost $S$-arc</th>
<th>% Imp.</th>
<th>Cost $Sd$-arc</th>
<th>% Imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net1</td>
<td>10</td>
<td>30</td>
<td>50</td>
<td>$RL$</td>
<td>16,078</td>
<td>14,413</td>
<td>10.3</td>
<td>14,389</td>
<td>10.5</td>
</tr>
<tr>
<td>Net2</td>
<td>10</td>
<td>34</td>
<td>50</td>
<td>$UL$</td>
<td>16,034</td>
<td>14,056</td>
<td>12.3</td>
<td>13,913</td>
<td>13.5</td>
</tr>
<tr>
<td>Net3</td>
<td>15</td>
<td>42</td>
<td>110</td>
<td>$RL$</td>
<td>26,195</td>
<td>24,672</td>
<td>5.8</td>
<td>24,084</td>
<td>8.1</td>
</tr>
<tr>
<td>Net4</td>
<td>15</td>
<td>46</td>
<td>110</td>
<td>$UL$</td>
<td>26,152</td>
<td>24,629</td>
<td>5.8</td>
<td>24,031</td>
<td>8.1</td>
</tr>
<tr>
<td>Net5</td>
<td>17</td>
<td>48</td>
<td>68</td>
<td>$RL$</td>
<td>15,295</td>
<td>14,681</td>
<td>4.0</td>
<td>14,457</td>
<td>5.4</td>
</tr>
<tr>
<td>Net6</td>
<td>20</td>
<td>60</td>
<td>200</td>
<td>$RL$</td>
<td>19,564</td>
<td>18,483</td>
<td>5.5</td>
<td>18,245</td>
<td>6.7</td>
</tr>
<tr>
<td>Net7</td>
<td>20</td>
<td>66</td>
<td>200</td>
<td>$UL$</td>
<td>19,201</td>
<td>18,070</td>
<td>5.9</td>
<td>17,893</td>
<td>6.8</td>
</tr>
<tr>
<td>Net8</td>
<td>30</td>
<td>86</td>
<td>500</td>
<td>$RL$</td>
<td>27,374</td>
<td>19,467</td>
<td>28.9</td>
<td>19,063</td>
<td>29.4</td>
</tr>
<tr>
<td>Net9</td>
<td>30</td>
<td>96</td>
<td>500</td>
<td>$UL$</td>
<td>27,011</td>
<td>19,210</td>
<td>28.9</td>
<td>18,620</td>
<td>32.0</td>
</tr>
<tr>
<td>Net10</td>
<td>40</td>
<td>120</td>
<td>3,000</td>
<td>$RL$</td>
<td>54,382</td>
<td>47,476</td>
<td>12.7</td>
<td>46,871</td>
<td>13.8</td>
</tr>
<tr>
<td>Net11</td>
<td>40</td>
<td>132</td>
<td>3,000</td>
<td>$UL$</td>
<td>53,971</td>
<td>47,225</td>
<td>12.5</td>
<td>46,823</td>
<td>13.2</td>
</tr>
<tr>
<td>Net12</td>
<td>50</td>
<td>160</td>
<td>8,000</td>
<td>$RL$</td>
<td>87,281</td>
<td>82,386</td>
<td>5.7</td>
<td>80,910</td>
<td>7.3</td>
</tr>
</tbody>
</table>

Table I - Computational results for $S$-arc and $Sd$-arc algorithms.

Finally, the generic nature of the presented algorithms allows them to support unprotected, 1+1 protected, and mesh-restored lightpaths. This in turn makes them potentially attractive for military applications, where flexibility and survivability are critical issues.

ACKNOWLEDGEMENTS. I would like to acknowledge former software developers from Tellium, Inc. who implemented Single-arc and Double-arc algorithms in a commercial software product. The computational results were based on that implementation.

References


Received: August 25, 2006