Augmentation Algorithms for Survivable Network Topologies in Wired and Optical Networks

Nikhil Garg Luv Kohli Rahul Simha, and Wenxun Xing

Abstract— **This paper falls in the area of survivable topology design where the goal is to design robust network topologies that anticipate link or node failures. While some survivable networks are designed from scratch, others must be built by *augmenting* existing networks with sufficient resources to enable survivability. The general topology augmentation problem addresses the general question: what additional resources are required to build upon an existing network to enhance its survivability? A typical such problem involves finding the fewest links to add to a topology to make the topology 2-edge-connected. This paper considers a cost-constrained variation of that problem: given a topology, a set of potential links each with a cost, and a limited budget, find links that can be added to enhance the topology's biconnectivity. We also consider an extension to optical networks in which fiber and wavelength resources are part of the problem. We present several different heuristics for both general wired networks and for optical networks. The main result of this work is that no single heuristic alone is effective but that a combination of them find near-optimal solutions when compared with simulated annealing.**

Index Terms— **Survivable network design, reliability, graph augmentation, topology design**

I. Introduction

Survivability is often an important goal in network design, particularly so for backbone and optical networks where a single link failure can cause vast amounts of data to be retransmitted. Accordingly, there has recently been considerable interest in designing topologies in anticipation of failures so that a single failure cannot disrupt connectivity. This interest has resulted in the study of various graph-theoretic design problems relating to topology design. Because it is unlikely that entire networks will be designed from scratch, many of these practical design problems involve building upon existing networks - the so-called *augmentation problem*. For example, the well-known biconnectivity augmentation problem as defined by [8] is defined as follows: given a topology with unit edge (link) costs, find the minimum number of edges that, if added to the edge-set, would result in two edge-disjoint paths for every source-destination pair. However, the augmentation problem in this form, forces certain unnecessary requirements on the topology that are neither practical nor always possible to achieve.

In this paper, we propose and study a new, practically useful variant of the general augmentation problem for both conventional and optical networks. In both cases, a limited budget for topology expansion must be used as effectively as possible in increasing overall network survivability. The general problem, as might be expected of this combinatorial nature, is shown to be NP-hard. We present and study several fast heuristics and compare them to solutions produced from lengthy executions of simulated annealing. Interestingly, no single heuristic alone returns best results; instead, we find that the combination of heuristics always produces near-optimal results. This is because the problem is such that both costs and topology structure play a role in determining an effective solution. Thus, the problem has a knapsack-like [10] flavor with its costs and limited budget, and yet has a path-finding flavor similar to a Hamiltonian-path problem (an observation made in [9] for augmentation problems in general). This dual nature makes it difficult for a single-strategy greedy heuristic to perform well. However, we find that a combination of heuristics, some cost-based and some path-based, appears to find near-optimal solutions as measured against solutions found through extensive use of simulated annealing.

Our new problem formulation is motivated by the limited practical impact of prior research efforts on the augmentation problem whose focus has been on *complete augmentation* where the goal is to ensure two edge-disjoint (or node-disjoint) paths for every pair of nodes. Most of these efforts, for theoretical convenience, assume unit edge costs. In contrast, we consider the following practical networking constraints.

- *Link Feasibility:* For cost, geographical or other reasons, it may not be possible to build a link between every pair of nodes. Thus, the set of potential edges that may be added to a topology graph is often a proper subset of the set of all possible edges.
- *Link Cost:* The unit cost assumption is rarely justified in practice: costs may vary drastically across the different potential edges since the cost of laying or leasing a new line may be higher in some geographical areas than others. For an optical network, the cost may also include the expenses of upgrading the switching capability of a node.
- *Partial Bi-connectivity:* Prior work has focused on complete augmentation, the requirement that every source-destination pair must have two fully edge-disjoint paths. Unfortunately, this objective may unnecessarily reduce the acceptable solution space for a given budget. We consider softening the requirement so that some alternate (backup) paths may overlap with their primary paths – this allows trading off cost against the risk of survivability. By explicitly modeling this tradeoff in the cost function, we assign a penalty to overlapped paths based on the degree of overlap, allowing a network designer to empha-

Nikhil Garg: Department of Computer Science, The George Washington University, Washington, DC 20052.

Luv Kohli: Department of Computer Science, University of North Carolina, Chapell Hill.

Rahul Simha: Department of Computer Science, The George Washington University, Washington, DC 20052. Email: simha@seas.gwu.edu. Partially supported by NSF grant ANI-9973106 and DARPA grant N66001-00-1-8949

Wenxun Xing: Department of Mathematical Sciences, Tsinghua University, Beijing, China. Email: wxing@math.tsinghua.edu.cn. Partially supported by NSFC-69904007.

size or de-emphasize this factor as desired. For optical networks, bi-connectivity is the typical protection requirement for connections. Hence, for a connection request that requires protection, bi-connectivity ensures an edge- or node-disjoint backup path, while a connection without such a strong protection requirement might share backup paths with other such connections.

- *Limited set of Connection Requests:* In contrast to other formulations, our input connection set need not be all-to-all; that is, we allow the input connection set to be a proper subset of source-destination pairs. This latter constraint is important in, for example, optical networks where connections have different protection requirements [11].
- *Limited funds:* We consider the special circumstance of optimization with a limited budget. This approach is more general because the budget can be increased to allow for complete augmentation solutions.

The above differences with past work are both practical and also expand the size of the solution space beyond those considered by other problem formulations.

The major contributions of the paper can be summarized as follows. We formulate a combinatorial optimization problem that captures the practical objectives described above. This formulation is presented as an Integer Programming (IP) formulation. The IP formulation, while time-consuming to solve optimally, proves useful in designing our heuristics, as we show. We first present a formulation for general networks and later present a similar formulation for optical networks. The formulation for conventional networks is referred to as *graph augmentation* whereas the optical version is referred to as *optical augmentation*. We describe several fast greedy heuristics for both the graph and optical augmentation problems and compare the heuristics' solutions to the solutions produced by simulated annealing. As mentioned earlier, the main contribution is an assessment of these heuristics and the finding that the combined application of the heuristics produces near-optimal results.

We now summarize some related work. A general overview of issues related to survivable network design may be found in [7], [18], [20]. Steiglitz et al [21] used local search to find the near optimal solution of a vertex-disjoint-path survivable network designing problem. Since then a number of papers have focused on the "empty network" problem in which the whole network is designed from scratch. The augmentation problem is thought to have originated in [8]. A recent excellent survey of the standard augmentation problem and some variants may be found in [9]. The additional constraint of forcing a bound on the total length of the two paths is also considered in [9]. Some authors consider both edge and node connectivity simultaneously [12]. Issues related to SONET networks are considered in [6].

Various additional constraints on the general problem have also been considered. For example, Ameur [3], Kort [13] and Monma and Shallcross [17] considered connection-length constraints and tree-shaped connection-sets. Myung et al [18] considered a richer set of alternative network topologies and analyzed their survivability in the worst-case failure involving k edges. Ball et al [4] studied the 2 edge-disjoint paths network-designing problem when the network has two component failures. Lucet et al [16] presented a decomposition method for computing the 2 edge-disjoint paths reliability of the networks with bounded pathwidth and their computational results show the efficiency of the method. Gaoxi et al [1] have proposed a two-stage cut saturation algorithm for augmenting all-optical networks by adding links, but did not consider the issue of protection (bi-connectivity), although they have considered multiple connection requests for the same node pair, but without the requirement of edge(node)-disjointness.

The rest of the paper is organized as follows. In Section II, we provide the definition and an integer programming formulation for the graph augmentation problem, which we will refer to as the *2-Path Minimal-Overlap Problem* (2-PMOP). We show that the problem is NP-hard. A relaxation formulation, its properties and algorithms is considered in Section III. Then in Section IV, we describe experimental results comparing the algorithms. We study the optical topology problem, subsequently reffered to as *Optical Augmentation Problem* (OAP), by defining it in Section V. The problem is shown to be NP-Hard in Section V-B, with the various heuristics described in Section VI. We compare the results from the various heuristics in Section VII before making concluding remarks in Section VIII.

II. Graph Augmentation Formulation

Recall that the graph augmentation problem refers to conventional wired networks. Accordingly, the network is modeled by a a connected, undirected network graph $G = (V, E)$ in which each link is assumed to be full-duplex. Later when we consider optical networks, we will also include fibers and wavelengths in the problem formulation. For the graph augmentation problem, we are also given a collection of source-destination required pairs $R = \{(s_1, d_1), \cdots, (s_k, d_k) : s_i, d_i \in V\}$ and a set of *potential edges* $F \subseteq \{(i,j) : i,j \in V \text{ and } (i,j) \notin E\}$ from which to augment the network. As usual, $V = \{1, 2, \cdots, n\}$ is the vertex set where each vertex represents a network node, E is the set of links or edges, for each of which we associate a "weight" or distance $d(i,j)$. For each $(i,j) \in F$ associate a nonzero positive cost of building an edge $c(i,j)$. The cost $c(i,j)$ is intended to model the purchase or leasing of a new link between nodes i and j.

For a source $s \in V$ and destination $d \in V$, let $P_{s,d}$ and $P'_{s,d}$ be two paths from s to d in G. Define the overlap distance of $P_{s,d}$ and $P'_{s,d}$ to be the total distance of common shared edges:

$$\alpha(P_{s,d}, P'_{s,d}) = \sum_{(i,j) \in P_{s,d} \cap P'_{s,d}} d(i,j).$$

Next, define

$$\alpha(G, E) = \sum_{(s,d) \in R} \alpha(P_{s,d}, P'_{s,d})$$

to be the total network overlap distance. We use the link length as the cost because, typically, the longer the link the higher risk of failure. However, this definition can be modified to also reflect other factors such as reliability or cost. In short, 2-PMOP is defined as follows: for a given budget B, find $T \subseteq F$ such that $\sum_{(i,j) \in T} c(i,j) \leq B$ and $\alpha(G, E \cup T)$ is minimized.

A more detailed integer-programming formulation of 2-PMOP is given next.

$$
\begin{aligned}
Z_1 \quad = \quad & \min \sum_{l=1}^{k} \sum_{(i,j) \in E \cup F} \frac{1}{2} d(i,j)(x_{ijs_l d_l} - 1) x_{ijs_l d_l} \\
s.t. \quad & \sum_{i:(j,i) \in E \cup F} x_{jis_l d_l} = 2, \; j = s_l, \; l = 1, 2, \cdots, k \\
& \sum_{i:(i,j) \in E \cup F} x_{ijs_l d_l} = 2, \; j = d_l, \; l = 1, 2, \cdots, k. \\
& \sum_{i:(i,j) \in E \cup F} x_{ijs_l d_l} = \sum_{i:(j,i) \in E \cup F} x_{jis_l d_l}, \\
& j \neq s_l, d_l, l = 1, 2, \cdots, k \\
& \sum_{l=1}^{k} x_{ijs_l d_l} \leq k * y_{ij}, \; (i,j) \in F, \\
& \sum_{(i,j) \in F} c(i,j) y_{ij} \leq B, \\
& x_{ijs_l d_l} \geq 0 \text{ and integer}, \; y_{ij} \in \{0, 1\}.
\end{aligned}
$$

In this IP formulation, $x_{ijs_l d_l} = x_{jis_l d_l}$ and $y_{ij} = y_{ji}$ are decision variables. The former is the number of paths for the source-destination pair (s_l, d_l) that traverse edge (i,j) whereas the latter is unity when edge (i,j) is selected for the solution. The objective Z_1 (a related objective Z_2 is defined later) seeks to minimize the overlap distance for any two paths of a request of R. The first two constraints represent the path constraints at the source and destination respectively. The third enforces flow balance (path integrity) at each vertex except the source and destination vertices. Note that the fourth constraint expresses the fact that there are at most k paths going through edge (i,j) when $y_{ij} = 1$; this constraint is further explained in Proposition 3. The last two constraints are budget and integer constraints respectively.

Many variations of the well-known augmentation problem are NP-hard. Because we penalize overlap instead of prohibiting it, our objective function is in some sense “softer” than the objective functions of many earlier augmentation problems. Therefore, it is natural to wonder whether the problem remains NP-hard. We show that it is NP-hard by reducing the Partition problem [10]: Given integers $c_1, c_2, \cdots, c_n$, is there a subset $S \subseteq \{1, 2, \cdots, n\}$ such that $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$

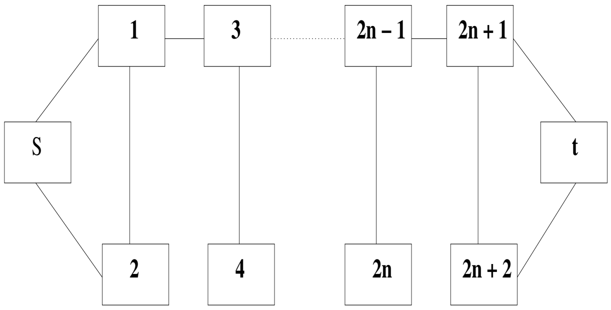

Fig. 1. 2-PMOP conversion of the partition problem

Given any instance of PARTITION, we construct a connected communication graph $G = (V, E)$ as shown in Figure 1 where $V = \{s, t, 1, 2, \cdots, 2n+2\}$ and $E = \{(s,1), (s,2), (1,2), (1,3), \cdots, (2n+1, 2n+2), (2n+1, t), (2n+2, t)\}$. The request source and destination pair is (s,t). The distance of each edge is defined as

$$
d(u,v) = \begin{cases} c_i, & \text{if } u = 2i-1, v = 2(i+1)-1, i = 1, 2, \cdots, n, \\ \infty, & \text{otherwise.} \end{cases}
$$

The cost of the potential edge is defined as

$$
c(u,v) = \begin{cases} c_i, & \text{if } u = 2i, v = 2(i+1), i = 1, 2, \cdots, n, \\ \infty, & \text{otherwise.} \end{cases}
$$

When the budget is $\frac{1}{2}\sum_{j=1}^{n} c_j$, we polynomially transform the instance of the PARTITION into a decision instance of 2-PMOP, which can be stated as: given a connected communication graph $G = (V, E)$, distances $d(u,v), (u,v) \in E$, costs $c(u,v), (u,v) \notin E$ and budget $\frac{1}{2}\sum_{j=1}^{n} c_j$, is there a solution of 2-PMOP such that $Z_1 \leq \frac{1}{2}\sum_{j=1}^{n} c_j$ and the total cost of the added edges is within the given budget?

Proposition 1: 2-PMOP is NP-hard.

Proof: It is easy to see that the common shared edges of 2-PMOP are $(2i-1, 2(i+1)-1), i = 1, 2, \cdots, n$. When moving one edge $(2i-1, 2(i+1)-1)$ of distance c_i out of the shared edge list, we have to add an edge $(2i, 2(i+1))$ with cost c_i. To derive the relationship between the solutions of PARTITION and 2-PMOP, we define

$$
y_{uv} = \begin{cases} 1, & \text{if } i \in S \text{ and } u = 2i, v = 2(i+1), \\ 0, & \text{otherwise,} \end{cases}
$$

which means if i is selected to be an element of S, then edge $(2i, 2i+2)$ is added to the graph. Consequently, the objective value satisfies

$$
Z_1 = \sum_{1}^{n} c_i - \sum_{i \in S} c_i.
$$

It is obvious that a ‘yes’ instance of PARTITION corresponds to a solution of 2-PMOP. Similarly, a solution of an instance of 2-PMOP implies a solution to the PARTITION instance. Thus 2-PMOP is NP-hard.■

III. Problem Structure and Algorithms

In this section, we first present some properties of the 2-PMOP formulation and then later use these properties in designing heuristics. The key idea is based on the solution of the following relaxation problem of 2-PMOP: Given a selected set $T \subseteq F$ of edges, define

$$
\begin{aligned}
Z_2(T) \quad = \quad & \min \sum_{l=1}^{k} \sum_{(i,j) \in E \cup T} \frac{1}{2} d(i,j)(x_{ijs_l d_l} - 1) x_{ijs_l d_l} \\
s.t. \quad & \sum_{i:(j,i) \in E \cup T} x_{jis_l d_l} = 2, \; j = s_l, \; l = 1, 2, \cdots, k \\
& \sum_{i:(i,j) \in E \cup T} x_{ijs_l d_l} = 2, \; j = d_l, \; l = 1, 2, \cdots, k
\end{aligned}
$$

$$\sum_{i:(i,j)\in E\cup T} x_{ijs_ld_l} = \sum_{i:(j,i)\in E\cup T} x_{jis_ld_l},$$
$$j \neq s_l, d_l, l = 1, 2, \cdots, k$$
$$x_{ijs_ld_l} \geq 0 \text{ and integer.}$$

In this relaxation model, the budget constraint of 2-PMOP is relaxed by a given set T; this set will be defined in various heuristics that we present later in this section. The following theorems outlines the solution of the relaxation problem. First, we observe a relationship between the shared edges and the existence of two edge-disjoint paths.

Proposition 2: Given a connected graph $G = (V, E)$ and two distinct vertices $s, t \in V$, all paths connecting s and t have common shared edges if and only if there are no two edge-disjoint paths connecting the two vertices.

Proof. By the min-cut theorem[19], an edge is shared by all the paths connecting s and t if and only if one min-cut set has this edge as the only element, that is, no two edge-disjoint paths connect s and t. ■

Proposition 3: Given a connected graph $G = (V, E)$ and two distinct vertices s and t, if edge $(i, j) \notin E$ is added to the graph then (i, j) is not a shared edge for all the paths connecting s and t.

Proof: If an added edge (i, j) is a shared edge for all the paths connecting s and t, then it is the min-cut of partition sets W and $\bar{W}$ by Proposition 2 such that s and t belong to the two sets respectively. Removing this edge causes disconnection of the graph, contradicting the assumed connectedness of the original topology. ■

Proposition 4: Given a connected undirected graph $G = (V, E \cup T)$ and two distinct vertices $s, t \in V$, the shared edges of the paths connecting s and t can be found in time $O(n^3)$. For the entire connection request set $R = \{(s_1, d_1), \cdots, (s_k, d_k) : s_i, d_i \in V\}$, the relaxation problem can be solved in time $O(kn^3)$.

Proof: For any request pair s, t, we first find one path connecting s and t. For example we can use Dijkstra's algorithm to find a shortest path in $O(n^2)$. (Note that faster variants exist that use newer data structures [5]). There are at most $n - 1$ edges on the path. Then by Proposition 2, we have to check which edges in the path are shared. For this purpose, successively delete one edge from the path, give the edge an infinite distance and use Dijkstra's algorithm again to check if there is at least one finite length path connecting s and t. When there is no finite length path connecting s and t after deleting an edge (i, j), then (i, j) is a common shared edge. The total computation time is $O(n^3)$. Thus for all k requests we solve the relaxation problem of 2-PMOP in $O(kn^3)$.■

Finally, for clarity, we state our original problem in terms of the relaxation. This formulation outlines the search space for each heuristic.

Proposition 5:

$$Z_1 = \min\{Z_2(T) : \sum_{(i,j)\in T} c(i, j) \leq B \; and \; T \subseteq F\}.$$

The above results are the basis for designing of heuristics: the set T is constructed by each heuristic and varies across the heuristics.

A. Greedy Algorithms and a Lower Bound

Recall that the input to our problem is a graph $G = (V, E)$, edge weights $D = \{d(i, j) > 0 : (i, j) \in E\}$, a 'potential' edge set $F \subseteq \{(i, j) \notin E\}$ and cost $C = \{c(i, j) \geq 0 : (i, j) \in F\}$, connection required pairs $R = \{(s_1, d_1), \cdots, (s_k, d_k) : s_i, d_i \in V\}$, and a budget B.

1) Simple Greedy Algorithm:

Step 0 Let $b = B$ and $T = \emptyset$.

Step 1 If $Z_2(T) = 0$, stop the algorithm: there exist two edge-disjoint paths for all request pairs. Otherwise successively consider (i, j) from $F - T$ one by one until no further edges can be added. If $c(i, j) \leq b$, then calculate $Z_2(T \cup \{(i, j)\})$ of the relaxation problem, otherwise the objective value remains $Z_2(T)$.

Step 2 Choose the edge with the largest $\frac{Z_2(T) - Z_2(T \cup \{i,j\})}{c(i,j)}$ and denote it as (u^*, v^*). If $Z_2(T \cup \{(u^*, v^*)\}) = Z_2(T)$, stop, with the local optimal solution T. Otherwise let $T = T \cup \{(u^*, v^*)\}$ and $b = b - c(u^*, v^*)$, return to Step 1.

At each iteration, the above greedy algorithm adds one edge at a time. The order of edges is determined, knapsack-like, by the per-unit-cost effectiveness in reducing the objective function. At each iteration, we compute the value of the relaxation objective for each edge in F, which takes $O(|F|kn^3)$ time according to Proposition 4. The overall complexity of this algorithm is $O(|F|^2kn^3)$.

2) A lower bound:

Proposition 6: For any subset T of F and two distinct edges $(i, j), (k, l) \in F - T$, we have

$$Z_2(T) \geq Z_2(T \cup \{(i, j)\}),$$

and

$$Z_2(T) - Z_2(T \cup \{(i, j), (k, l)\})$$
$$\leq Z_2(T) - Z_2(T \cup \{(i, j)\}) + Z_2(T) - Z_2(T \cup \{(k, l)\}).$$

Therefore a lower bound for 2-PMOP is

$$Z_L = \max\{0, Z_2(\emptyset) - B \max\{\frac{Z_2(\emptyset) - Z_2(\{(i, j)\})}{c(i, j)} : c(i, j) \leq B, \; (i, j) \in F\}\}.$$

Proof: The first inequality is easy to derive. For any common shared edge (u, v), whose contribution to the objective function is removed, when adding edges (i, j) and (k, l) to the graph, there exist two vertex cut sets W and $\bar{W}$, such that $W \cap \bar{W} = \emptyset$ and $u \in W$, $v \in \bar{W}$ from the proof of Proposition 2. Thus, objective function is decreased by the shared distance of this edge if and only if at least one of the two edges (i, j) and (k, l) connects two vertices in W and $\bar{W}$ respectively, giving us the second inequality.

Let $T^* = \{(u_1, v_1), (u_2, v_2), \cdots, (u_l, v_l)\}$ be the optimal solution of 2-PMOP, we get

$$Z_2(\emptyset) - Z_2(T^*)$$
$$\leq \sum_{p=1}^{l} [Z_2(\emptyset) - Z_2(\{(u_p, v_p)\})]$$
$$\leq \max\{\frac{Z_2(\emptyset) - Z_2(\{(i,j)\})}{c(i,j)} : c(i, j) \leq B \text{ and } (i, j) \in F\}\{\sum_{p=1}^{l} c(u_p, v_p)\}.$$

This last inequality implies the bound. ■

B. Path-based Greedy Algorithm

The previous greedy algorithm was based on defining a "profitability" value for each edge and then choosing the edges in decreasing order of profitability. However, experimental results show that this approach is not always effective. The reason is that, in many cases, the overlap on a certain path is best reduced by a secondary *path* (rather than a single edge) that may need to be strung together from the set of potential edges. Furthermore, in considering the addition of new paths, each such potential path might impact different source-destination pairs in various ways, reducing overlap for some but not for others. Thus, the path complexity includes not only structural properties of the topology but also the particular set of source-destination pairs. To accommodate these different concerns, we attempt to first identify some "important" sub-paths and weight the edges in the sub-paths according to their contribution to reducing the objective function.

Step 0 Let $T = \emptyset$. Halt if there exist 2-edge disjoint paths for all the request pairs.

Step 1 Construct a bi-connected components tree from G, and call it $\bar{G}$.

Step 2 Re-map F and R, for $\bar{G}$, reducing the size of the problem. Let the new problem be ($\bar{G}$,$\bar{F}$,$\bar{R}$).

Step 3 Let $S = \{(i,j) : (i,j) \in \bar{F}$ and $(i,j) \in \{p_1 \cup p_2 \cup p_k\}$ where p_i is a path for ith pair in $\bar{R}\}$

Step 4 Let $P = \{p_i : p_i$ is a secondary path for the ith pair, $c(p_i) \leq B\}$. Here the secondary path is the shortest path after removing all edges e_j, where $e_j \in S$, in the path for the ith pair from $\bar{G}$.

Step 5 Assign a profit function f over the paths:

$$f(p_i) \quad = \quad d(p_i) + \sum_{e_l \in p_i, e_l \in p_j st p_j \in P, p_j \neq p_i} d(p_j) * \frac{c(e_l)}{c(p_j)}$$

Here $d(p_i) = d(i,j)$ if, p_i is the secondary path for edge (i,j), and $c(p_i) = \sum_{e_j \in p_i} c(e_j)$.

Step 6 Rank the paths using f, and then successively include paths until the budget is exceeded or until 2 edge-disjoint paths for all pairs have been found.

The "profit" function f for a path p_i, is directly proportional to the length of the critical edge for that, but also depends upon the partial length of the critical edges, whose secondary paths share edges with p_i. This gives greater importance to paths that consist of frequently occurring edges.

C. Largest-Overlap-Reduction Greedy Algorithm

We now examine a potential weakness of the previous heuristic. Recall that the above heuristic finds alternate paths for articulation edges and attempts to add these to the topology within the budget constraint. However, an alternate path for a particular articulation edge might have many edges. Moreover, if there are several such articulation edges in a single primary path, it may not be possible to select every edge in the alternate paths for these critical edges. Instead, we now propose to stitch together some of these alternate paths into more compact paths by weighting commonly occurring edges more heavily. Note that the path-based heuristic was unable to select individual edges that along with edges from other paths could have further reduced the total overlap, but was not listed in any path since it did not occur on the shortest secondary path for a critical edge. Thus, once again we identify a profitability function for edges rather than paths. However, we retain the usefulness of working with paths by assigning profitability based on overlap reduction for individual paths, rather than the total overlap reduction.

Step 0 Let $T = \emptyset$. Stop the algorithm if there exists a 2-edge disjoint path for every request pairs.

Step 1 Let $L = \{l_i : l_i$ be the list of critical nodes for $s_i d_i, \forall s_i d_i \in R\}$. The two vertices of a critical edge are critical nodes.

Step 2 Choose edges (i,j) from F, one by one until no edge can be chosen, calculate the longest overlap reduction for each of the source-destination pair i, by considering all node pairs in l_i, with secondary paths containing edge (i,j). Assign profit $g(i,j)$ as the largest overlap reduction possible over a path for any s-d pair i by including edge (i,j).

Step 3 Choose edges from F according to the function g, until the budget exceeds or 2 edge-disjoint paths are found for all request pairs.

The profit function g will assign the same high "profit" to all the edges in the secondary path for the largest overlap reduction, thus retaining the path-based approach, but with an edge based greedy solution.

D. Local search

Based on the formulation of 2-PMOP and Proposition 4, a natural way to represent a solution of 2-PMOP is to provide a configuration of potential edges, such as

$$Y = \{y_{ij} \in \{0,1\} : (i,j) \in F\} \in \{0,1\}^{|F|},$$

where $y_{ij} = 1$ means that (i,j) is added into the graph. Thus, for a given Y, we can obtain a set T of added edges by $T = \{(i,j) : y_{ij} = 1\}$, denote it as $T(Y)$, and easily compute the objective value $Z_2(T(Y))$ by solving the relaxation problem in $O(kn^3)$.

For local search , we define the neighborhood of Y as

$$N(Y) = \{W \in \{0,1\}^{|F|} : |Y - W| = \sum_{(i,j) \in F} |y_{ij} - w_{ij}| \leq 2\}.$$

Note that the number of the neighbors of Y is $1 + |F| + \frac{|F|(|F|-1)}{2}$. The local search algorithm is described below:

Step 0 Let K be the maximum number of randomly selected solutions. Let $Y = 0, k = 0$. If $Z_2(\emptyset) = 0$, there exist 2 edge-disjoint paths for each request pair. Otherwise denote $z = Z_2(\emptyset)$, the best solution $Y^* = Y$ and the best value $z^* = z$.

Step 1 While $k < K$, find the best neighbor W^* from $N(Y)$, such that

$$\begin{aligned} Z_2(T(W^*)) \quad &= \quad \min\{Z_2(T(W)) : W \in N(Y) \\ &\qquad \text{and} \sum_{(i,j) \in F} c(i,j) w_{ij} \leq B\}. \end{aligned}$$

Step 1.1 If $Z_2(T(W^*)) \geq z$, then randomly change at most three ones in Y to zeros and also denote the solution as Y, $k = k+1$, $z = Z_2(T(Y))$, go to Step 1.

Step 1.2 Otherwise $Z_2(T(W^*)) < z$, then $Y = W^*$ and $z = Z_2(T(Z^*))$. If $z \leq z^*$, then $z^* = z$ and $Y^* = Y$. Go to Step 1.

Step 2 Output the best solution Y^* and value z^*.

Local search is a simple heuristic that is sometimes effective in combinatorial optimization problems [15]. However, as is well known, local search can also get stuck in a highly sub-optimal local minimum. We next consider simulated annealing as the natural extension to using local search.

E. Simulated annealing

Simulated annealing has been found to provide reasonably good solutions for many NP-hard problems [14] and therefore to provide a baseline against which to evaluate faster heuristics. Simulated annealing has the following essential components: the initial solution, a method for selecting a random neighbor in solution space, a cooling schedule and a stopping criterion.

For the given potential edge set $F = \{(i,j) : (i,j) \notin E, i,j \in V\}$, we denote a solution of 2-PMOP as a sequential order of the elements in F, i.e.,

$$S = \{(i_1, j_1), (i_2, j_2), \cdots, (i_{|F|}, j_{|F|})\}, \tag{1}$$

where $(i_l, j_l) \in F$ and is the lth edge added into the graph $G = (V, E)$. By considering the budget constraint, we can find $k(S)$ such that

$$\sum_{l=1}^{k(S)-1} c(i_l, j_l) \leq B, \sum_{l=1}^{k(S)} c(i_l, j_l) > B.$$

Consequently we divide S into two parts, the first $k(S)$ edges and last $|F| - k(S)$ edges. Similar to the 2-opt neighborhood function of Lin [15], we interchange elements between the two sets to create a neighboring candidate solution. Note that two solutions S and S' are different if and only if the set of the first $k(S)$ edges in S is not the same to that of the first $k(S')$ edges in S'. Finally, it is easily verified that any two solutions are reachable by repeatedly using the interchanging procedures. Therefore the neighborhood of S is defined as

$$N(S) = \{S' : \text{two elements of the distinct parts of } S \text{ interchange their positions or two elements from one part are exchanged for one element from the other part.}\}$$

One can check that there are $k(S)(|F| - k(S))$ neighbors of S. From the variety of cooling schedules possible, we use the simple linear cooling schedule for which the initial temperature was calculated to allow the freedom to search the entire neighborhood initially. The algorithm is halted after a (large) finite number of temperature changes.

V	**E**	**F**	**R**	**LB**	**A1**	**A2**	**A3**	**A4**	**A5**
6	6	3	3	0.94	0.94	0.94	0.94	0.94	0.94
6	6	6	3	0.96	0.65	0.54	0.56	0.65	0.65
8	8	3	3	0.92	0.76	0.76	0.68	0.76	0.76
8	8	10	3	1	1	0.76	0.88	1	1
10	10	5	3	0.67	0.64	0.58	0.54	0.64	0.64
10	10	10	3	0.98	0.85	0.52	0.66	0.85	0.85
10	10	30	3	1	0.97	0.70	0.90	0.97	0.97
15	15	5	3	0.89	0.56	0.37	0.50	0.56	0.56
15	15	15	3	1	0.80	0.35	0.80	0.85	0.85
20	20	5	3	0.88	0.50	0.39	0.56	0.56	0.56
20	20	20	3	1	0.80	0.56	0.54	0.80	0.80
25	25	10	3	0.88	0.53	0.43	0.42	0.60	0.60
25	25	25	3	0.98	0.73	0.51	0.60	0.73	0.76
30	30	10	3	1	0.73	0.59	0.52	0.73	0.73

TABLE I

COMPUTATIONAL RESULTS: SUMMARY

```
 V = number of vertices
 E = number of edges
 F = number potential edges
 B = budget
A1 = Greedy Algorithm
A2 = Path Based Greedy Algorithm
A3 = Largest Overlap reduction Based Greedy
     Algorithm
A4 = Local Search Algorithm
A5 = Simulated Annealing Algorithm
```

IV. COMPUTATIONAL RESULTS FOR 2-PMOP

We implemented the heuristics proposed in the previous section. We also sought to use the CPLEX optimization package to run Integer Programming on small data sets for comparison. However, CPLEX, in its current version does not yet support the type of objective function in our formulation. At the same time, we note that annealing is run long enough to obtain good solutions to be able to compare the heuristics.

Test instances

The test instances used to evaluate the proposed heuristics consist of randomly generated instances of 2-PMOP. They are divided into different classes according to the budget levels and the number of vertices, edges and potential edges. These edges are randomly selected but leave the graph connected. The potential edges as well as source-destination node pairs are also randomly selected and their cost is directly proportional to their lengths. The stopping criteria for the simulated annealing algorithm was set at 10000 temperature adjustments.

Results

Table I provides an example of the kind of typical results obtained for some test graphs of different sizes. The results are presented as the fractional overlap reduced after adding potential edges from F. As can be seen, almost in all cases, one of the three greedy heuristics has been able to provide a solution as good as the one obtained from simulated annealing. Notice that, although algorithm A2 always seems to provide a solution as good as one of the other heuristics, it never outperforms them. Also, among the two state-space search algorithms, local search and simulated annealing, local search executes faster but does not always produce the best results. As is evident from Table I, A1 performs well for small size graphs, but A3 works well with larger graphs, though the average values do not show it, but individual test instances show that A3 outperforms A1 regularly with larger graphs. Notice that none of the algorithms perform as well as the lower bound for the problem, indicating that the constraints are often active.

The execution-time graph in Figure 2 below is strong evidence that the greedy algorithms can be used in combination instead of simulated annealing. However for smaller graphs, it might be useful to actually use the more time-consuming simulated annealing. The execution time of annealing greatly increases as the number of edges in the graph increases whereas the greedy heuristics execute within reasonable time. Instances of 2-PMOP with unit-cost edges were also tested, and it was found that in those cases the solution obtained from the heuristics were similar, from which we may infer that the problem does depend strongly on the costs of potential edges.

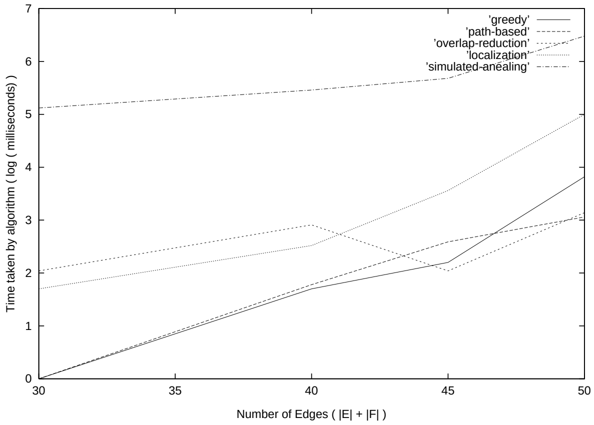

Fig. 2. Execution time performance

V. Augmentation Formulation for Optical Networks

We believe an augmentation formulation for optical networks is important for several reasons. First, the backbone of many conventional networks is often an optical network. Second, because of its high capacity, an optical link failure can result in significantly higher loss of data than a conventional link; indeed, most routing problem formulations in optical networks already account for so-called "protection" requirements in which a backup path is assigned along with a primary path. Finally, while augmentation has been studied for conventional networks, there is very little work along those lines for optical networks.

Our optical network augmentation formulation differs from the previous formulation for conventional networks in several ways. First, each link is modeled as a collection of fibers each of which supports a given number of wavelengths. For simplicity, we assume that the number of wavelengths per fiber is fixed and common to all fibers and that the number of fibers per link is also fixed and common to all links. This assumption reflects a single technology being used in the network, but can be relaxed without significantly changing the heuristic approach. Second, we note that adding a link can incur a significant switch upgrade cost if the switch has already used up all its ports. – our formulation includes switch upgrade costs as well. Third, in keeping with current practice in optical network routing, we provide a backup protection path for each connection. Since industry practice provides for a completely disjoint backup path, our formulation prohibits any overlap between primary and backup path. Therefore, rather than minimizing overlap we focus on minimizing costs to sufficiently augment an optical network in order to support backup paths for a given set of connections. Finally, note that a single wavelength is used end-to-end to support a connection; this wavelength contiguity constraint is typical in most optical routing problems.

We refer to the above problem as the Optical Augmentation Problem (OAP). A formal problem formulation is presented next.

A. *Formulation of OAP*

We are given a connected network graph G=(V,E) representing the topology, where each link $e \in E$ is a collection of fibers, each of which can carry multiple wavelengths. Each graph node represents an optical switch. The input consists of connection requests $r \in R$ each of which has a source node and a destination node. Let $P = \{(i,j) : i,j \in V\ , (i,j) \notin E\}$ be the set of potential links that can be added to the network. As before, for each $(i,j) \in P$ we associate a positive cost of constructing the link $c(i,j)$, usually proportional to the distance between the nodes. Our formulation allows a setup cost to be added as well.

The objective of the problem is to add additional links to the network such that each connection has a primary and backup path, and to incur the least overall cost in adding the additional resources. As mentioned earlier, while the problem formulation is general, we only consider the special case where the number of fibers per link is the same across all links and where the number of wavelengths per fiber is the same across all fibers. This assumption reflects the use of a common optical technology; however, our algorithmic approach can be extended to the general case in a straightforward manner.

OAP can be viewed as consisting of three parts. The first part adds resources (links) at minimal cost to support primary and backup paths. The second assigns path-pairs (a primary and a backup) to each source-destination pair. And the third assigns a wavelength along each assigned path. Each part is individually NP-hard, and since they are inter-dependent, they increase the complexity of the problem. In this paper, we focus on the first part – the augmentation problem. However, the second and third parts require a solution. We make two simplifying assumptions to reduce the complexity of the latter two parts. Consider the second part, finding primary and backup paths. Although in principle one could consider all possible paths from source to destination, we use the shortest path as the primary path, and select the backup path from the remainder of the network (after removing the primary path from consideration). This assumption is commonly applied in practice and greatly reduces the path complexity of the problem. Next, for the third part, we use a result from Erlebach et al [2] in which an upper bound is given for the number of wavelengths required in a directed fiber tree. We use this result by assuming that if the maximum load in the network can be bounded, a wavelength routing for the assigned paths is always possible. This simplification reduces the complexity of the problem; however the problem remains NP-hard, as we show next.

B. Problem Complexity

We first examine the complexity of OAP and show that it is NP-hard.

Proposition 7: OAP is NP-hard.

Proof: The well-known NP-complete Partition problem can be reduced to an instance of the OAP problem.

The partition problem is defined as: given integers c_1, c_2, ... , c_n is there a subset $S \in \{1, 2, ..., n\}$ st $\sum_{i \in S} c_i = \sum_{j \notin S} c_j$?

Given any instance of the partition problem, we create a network graph N of the following type, N = (V,E), where V = {s,t,1,2,....,2n-1,2n }, E = {(s,t)} and P = {(1,2),(3,4),(5,6),...,(2n-1,2n),(s,1),(s,3),(s,5),...,(s,2n-1),(2,t),(4,t),(6,t),...,(2n,t) }. The edge (s,t) has $\sum_{1 \leq i \leq n} c_i/2$ fibers, while all edges (2i-1,2i), (s,2i-1) and (t,2i) have c_i fibers $\forall i \in \{1, 2, ..., n\}$. All fibers carry single wavelength. And the maximum degree of a node is infinite. The cost function of edges in P is defined as follows:

$$
\begin{aligned}
c(2i-1, 2i) &= c_i \\
c(s, 2i-1) &= c_i \\
c(2i, t) &= c_i
\end{aligned}
$$

where $i \in \{1, 2, ..., n\}$

The connection request set has $\sum_{1 \leq i \leq n} c_i/2$ pairs of (s,t), with the 1+1 path protection requirement, which requires two node-disjoint paths. Thus we have transformed the partition problem into the OAP problem in polynomial time. A solution for the above OAP problem would answer the partition problem, if the minimal cost of the set of edges to solve the problem is $\sum_{1 \leq i \leq n} c_i/2$. As the partition problem has been shown to be a special case of the OAP problem, OAP is NP-hard.

VI. Heuristic Algorithms for OAP

In the same manner that we devised heuristics for the graph augmentation problem, we now consider similar heuristics for the optical augmentation problem. In the algorithms proposed below, the load of a link at any given intermediate point of execution is the number of paths assigned up to the intermediate point that contain the link.

A. Simple Greedy Algorithm (SG)

We first consider the simplest of greedy approaches. The network is augmented with the minimal cost edge until either all requests are met or the set of potential edges has been exhausted.

Step 1 Compute primary and protection paths.

Step 2 Calculate the load on each link, l_e .

Step 3 Recompute the primary and protection paths to reduce the load on the overloaded fibers (links) and utilize the less-used fibers (links).

Step 4 If the total network $Overload$ is zero or there are no links left to add then exit.

Step 5 Add the minimum cost link $e \in P$ to the network. $E = E \cup e$, $P = P - e$, go to Step 2.

In Step 5, if there is more than one link that has the minimum cost value, the link that requires the least upgrade cost is chosen, and in case the upgrade costs are also the same, the first link that was considered is chosen.

B. Best Resource Addition Algorithm (BR)

In this algorithm, the resource (link) addition is based on the greedy criterion of the most profitable resource, similar to what was defined in Section II for the graph augmentation problem. Profitability is defined as the network requirement fulfilled per unit cost of the resource added to the network. Thus, let $U(e, i)$ and $U(e, j)$ be costs of upgrading the end nodes of a link $e = (i, j)$. Since switches are usually available in sizes that are powers of 2, as a result switching capability can only be increased in discrete quantities. Hence, the cost of upgrading a node depends on its current state in the algorithm, since at any intermediate step it might have been upgraded and hence may not require any extra upgrade cost for subsequent links that are added at that node.

Step 1 Compute primary and protection paths.

Step 2 Calculate the load on each link, l_e .

Step 3 Recompute the primary and protection paths to reduce the load on the overloaded fibers (links) and utilize the less-used fibers (links).

Step 4 If the total network $Overload$ is zero or there are no links left to add then exit.

Step 5 For each edge $e \in P$, compute the total network overload upon adding it to the network, and call it $overload_e$.

Step 6 If for any edge $e(i, j)$, $overload_e$ is 0, then let $e = \{e : minc(e) + U(e, i) + U(e, j) \forall e$ st $overload_e = 0\}$, $E = E \cup e$, $P = P - e$, exit. Else, add edge e to E for which $(Overload - overload_e)/(c(e) + U(e, i) + U(e, j))$ is maximum, $P = P - e$, and go to Step 2.

The maximum load for a link is a function of the number of wavelengths that the link carries. An *overloaded link* above is a link whose load exceeds its maximum allowed load, where *Overload* is defined as the amount by which it exceeds that maximum load. Recomputation of paths is done by excluding overloaded links in computing paths between nodes. Only if by recomputing backup paths, there is still an overload on network links, we then recompute primary paths too.

C. Largest Request Completion Algorithm (LRC)

As we have seen in the graph augmentation heurisitcs, the cost based approach does not always perform well particular when the path structure of the problem comes into play. We try to take advantage of the path-first approach discussed in Section III, and shift the profitability criteria from the individual edges to entire paths, which in this case turn out to be a pair of paths (primary and protection). We define the largest pending request based on the overload seen on the primary and protection paths for the request. After identification of such a request, resources (links) are added to the network in order to fully establish that connection request.

Step 1 Compute primary and protection paths.

Step 2 Compute the load on each link, l_e .

Step 3 Recompute the primary and protection paths to reduce the load on the overburdened fibers (links) and utilize the less-used fibers (links).

Step 4 If the total overload is zero or there are no links left to add then exit.

Step 5 Let $R\prime$ be the largest incomplete request.

Step 6 Calculate the set $S \subseteq P$ of links that allows the connection requirements of $R\prime$ to be met upon being appended to the network. If no such set of links is found, then discard the request, and go to Step 5 else append S to the network and go to Step 2.

Finding the cheapest subset of links to add to the network to satisfy the largest request is again a combinatorial problem in itself; therefore, we use a greedy approach to repeatedly add the cheapest links until the largest request is met. The cost of each link not only includes its cost but the cost of upgrading the associated end nodes as well.

D. Simulated Annealing Approach (SA)

As with the graph augmentation problems, simulated annealing is again used to provide a good baseline solution by searching the solution space. The basic annealing technique is the same as defined earlier, but for OAP we need to specify the initial solution, a method for selecting a random neighbor in solution space, a cooling schedule and a stopping criterion.

For the given potential edge set $F = \{(i,j) : (i,j) \notin E, i,j \in V\}$, we denote a solution of OAP as a set of elements in F, i.e.,

$$S = \{(i_1, j_1), (i_2, j_2), \cdots, (i_{|F|}, j_{|F|})\}, \tag{2}$$

where $(i_l, j_l) \in F$ and is the l-th edge added into the graph $G = (V, E)$. Consequently, we divide S into two parts, the first $k(S)$ edges and last $|F| - k(S)$ edges. Similar to the 2-opt neighborhood function of Lin [15], we interchange elements between the two sets to create a neighboring candidate solution. Note that two solutions S and S' are different if and only if the set of the first $k(S)$ edges in S is not the same as that of the first $k(S')$ edges in S'. A solution is better than some other solution only if the overload on the network from that solution is less than the overload from the other solution. Finally, it is easily verified that any two solutions are reachable by repeatedly using the interchanging procedures. Therefore the neighborhood of S is defined as

$$\begin{aligned} N(S) &= \{S' : \text{two elements of the distinct parts of } S \\ &\quad \text{interchange their positions or one element} \\ &\quad \text{from one part is thrown into the other part.} \end{aligned}$$

One can check that there are $k(S)(|F| - k(S))$ neighbors of S. From the variety of cooling schedules possible, we use the linear cooling schedule for which the initial temperature was calculated to allow the freedom to search the entire neighborhood initially. The stopping criteria is a constant number of such iterations.

VII. Computational Results for OAP Heuristics

To test and compare the heuristics, we devised a variety of test cases to cover a broad range of parameter values and to expose differences between the heuristics. The following assumptions are used in our test cases. Each link has a single fiber and all fibers support five wavelengths. The switching capability of a node can only be doubled (stepped up to the next power of 2). The cost of upgrading a switch has a fixed cost incurred each time the number of ports crosses a power of 2.

The network parameters that we have varied across test cases include: the network size, the number of requests, the size of the potential link set and the protection requirements. The initial network, potential links and the requests are randomly generated. In particular, we present the performance for the following test cases:

1) *Model 1*
   A sparse network with low connection request density. The inital network has 10% of the total links possible for that network, while the potential link set has further 10% links available, and the connection requirements are 10% of the total possible(all-all pairs). Full protection requirements for all requests. All links are unit cost.
2) *Model 2*
   High connectivity with a large request set. The initial network is assumed to have 50% of the total possible links and approximately 50% of the rest are available as potential links. Connection requests are 50% of all possible. Rest of the parameters are same as Model 1.
3) *Model 3*
   A non-unit cost links network. Link cost is a randomly distributed between 1 and 50.
4) *Model 4*
   A network with 15 nodes, and 25% connectivity. We will study the performance under different request density.

Performance Metrics

We use the following performance metrics to evaluate the heuristics:

1) *Solution cost*
   This is the value of the objective function as formally defined in the problem formulation.
2) *Solution size*
   The number of links that are added to the network is defined as the size of the solution. In some applications, the number of links added to the network may be more important than the actual cost of establishing those links, reflecting the installation labor costs.
3) *Degree of solution completion*
   A given connection set may not be supportable even after adding every possible link from the given set of potential

links. In this case, the degree to which a feasible solution can be provided for some of the requests is important. For example, a network service provider may only have limited resources to upgrade its existing infrastructure, and the requests it receives exceed available resources. The provider can then use a partial solution to provide service to a subset of those requests.

4) *Time efficiency*
The execution time of a heuristic.

Results

Several network topologies were randomly generated for each of the network models described above. As seen in the table above, the LRC algorithm produces better results than the other two greedy algorithms if there are enough links to provide for a complete solution (satisfying all requests). At the same time, since the LRC algorithm chooses links that are good for individual requests but perform poorly on the whole, it performs poorly in producing partial solutions for sparse networks. In execution time, it is much faster than the other greedy algorithms.

Nodes	SG	BR	LRC
10	8	8	2
15	18	18	12
20	34	34	22
25	54	54	39

TABLE II
MODEL 1: NUMBER OF LINKS ADDED TO THE NETWORK

Nodes	SG	BR	LRC
10	22	22	11
15	57	57	48
20	79	79	53
25	90	90	82

TABLE III
MODEL 1:PERCENTAGE OF REQUESTS COMPLETED

Nodes	SG	BR	LRC
10	29	29	5
15	52	52	42
20	128	128	72
25	190	190	119

TABLE IV
MODEL 1: AUGMENTATION COST

Our test cases reveal the same overall characteristics for this set of heuristics as with the heuristics for the graph augmentation problem. Some heuristics are often better than others, but no single heuristic dominates. At the same time, the combination of heuristics produces near-optimal solutions.

Table IX and Table X show that as the requests increase, the network connectivity becomes smaller compared to the demands, resulting in better solutions from BR and SG. Although LRC almost always provides a smaller cost solution, that require less number of links to be augmented to the network. A large number of requests can still be completed using LRC for a smaller investment.

The two greedy algorithms SG and BR return similar results for unit cost links, but in case of non-unit cost links, their performance varies and depending on the topology and the cost structure either can provide a better solution. It is evident from the tables, that the various performance metrics are not interrelated, and an algorithm returning good results for one metric is not certain to provide better solutions for other metrics too. Simulated annealing is not practical because it requires

Nodes	SG	BR	LRC
10	240	260	10
15	1012	17194	80
20	14351	269267	460
25	53767	1515319	2284

TABLE V
MODEL 1: EXECUTION TIME

Nodes	SG	BR	LRC
5	5	3	2
10	3	22	4
15	51	52	5
20	95	95	4

TABLE VI
MODEL 2: NUMBER OF LINKS ADDED TO THE NETWORK

Nodes	SG	BR	LRC
5	100	100	100
10	100	100	100
15	100	100	100
20	100	100	100

TABLE VII
MODEL 2:PERCENTAGE OF REQUESTS COMPLETED

Nodes	SG	BR	LRC
5	17	11	6
10	3	78	20
15	179	196	21
20	127	127	20

TABLE VIII
MODEL 2: AUGMENTATION COST

Conn Req %	SG	BR
10	2	4
10	3	16
50	22	18
50	22	10

TABLE IX

MODEL 3: NUMBER OF LINKS ADDED TO THE NETWORK

Conn Req %	SG	BR
10	10	147
10	19	369
50	511	372
50	627	183

TABLE X

MODEL 3:AUGMENTATION COST

extremely long execution times and does not provide the flexibility of choosing solutions based on individual merits of cost or number of links.

Conn Req %	SG	BR	LRC
10	9	6	9
50	79	79	63
75	79	79	77
100	79	79	73

TABLE XI

MODEL 4: NUMBER OF LINKS ADDED TO THE NETWORK

Conn Req %	SG	BR	LRC
10	100	100	100
50	100	100	100
75	100	100	91
100	100	100	83

TABLE XII

MODEL 4:PERCENTAGE OF REQUESTS COMPLETED

Although the network models that have been simulated are single fiber models. It is not difficult to see that the algorithms are easily extendible to multi-fiber networks as well. In case of multi-fiber links, the link load function would become the maximum of all loads over each individual fiber. Among other assumptions for the network model, we have also assumed that there is no wavelength conversion capability present in the network. Our algorithms can be applied regardless of the presence or the absence of that capability. In the presence of wavelength conversion, the only change needed would be the upper bound for the maximum load on the links. We have left that upper bound undefined in this paper, since the algorithms use that as an input parameter. Erlebach et al [2] have shown that the number of wavelengths required for routing in a directed fiber network is bounded by $5L/3$, where L is the maximum load on the network. Hence accordingly, the maximum network load can be bounded by the number of wavelengths. Since there is a possibility of better bounds being available for optical networks, we have chosen to ignore the exact value of the bound, and instead work with the idea that there exists *some* bound.

Conn Req %	SG	BR	LRC
10	9	14	25
50	349	349	253
75	371	371	353
100	331	331	261

TABLE XIII

MODEL 4: AUGMENTATION COST

VIII. CONCLUSION

In this paper, a variant of the general augmentation problem was presented that forms an important sub-problem in the area of network survivable design. This augmentation problem was defined for both for conventional wired networks as well as optical networks. The problem was shown to be NP-hard in both cases. Several heuristics were presented and evaluated. It was shown that none of the greedy heuristics alone were sufficient because of the unusual nature of the problem that combines both the cost structure as well as topological stucture. Instead, a combination of the heuristics performed well over different test cases. Since the combined execution time of the different heuristics for each problem is orders-of-magnitude faster than well-known state-space algorithms such as simulated annealing, it makes a compelling argument to run all heuristics and then to pick the best solution produced.

REFERENCES

[1] Gaoxi Xiao, Yiu-Wing Leung and Kwok-Wah Hung, Two-Stage Cut Saturation Algorithm for Designing All-Optical Networks, IEEE Transactions on Communications, Vol 49, No 6 (2001).

[2] Thomas Erlebach, Klaus Jansen, Christos Kaklamanis Milena Mihail, and Pino Persiano, Optimal Wavelength Routing on Directed Fiber Trees, Theoretical Computer Science (221) 1-2 (1999), 119-137.

[3] W.B. Ameur(2000), Constrained length connectivity and survivable networks, Networks, 36(1), 17-33.

[4] M.O. Ball, J.N. Hagstrom and J.S. Provan(1998), Two-path subsets: efficient counting and applications to performability analysis, Discrete Applied Mathematics, 85(1), 25-45.

[5] T.Cormen, C.E.Leiserson, R.Rivest and C.Stein. Introduction to algorithms (2nd edition). MIT Press, 2001.

[6] S. Cosares, N.D. Deutch, I. Saniee, O.J. Wasem(1995), SONET toolkit: A decision support system for designing robust and cost-effective fiber-optic networks, Interfaces 25 20-40.

[7] R.J.Ellison, D.A.Fisher, R.C.Linger, H.F.Lipson, T.Longstaff and N.R.Mead. Survivable network systems: an emerging discipline. Technical report CMU/SEI-97-TR-013, Carnegie Mellon University, 1997.

[8] K.P.Eswaran and R.E.Tarjan. Augmentation problems. SIAM J. Computing, Vol. 5, 1976, pp.653-665.

[9] B. Fortz, Design of survivable networks with bounded rings. PhD thesis, Universit'e Libre de Bruxelles, Service de Math'ematiques de la Gestion, 1998.

[10] M.R. Garey and D.S. Johnson(1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

[11] O.Gerstel and R.Ramaswami. Optical layer survivability: a service perspective. IEEE Comm. Magazine, Vol. 38, No. 3, 2000.
[12] T.Ishii, H.Nagamochi and T.Ibaraki. Optimal augmentation to make a graph k-edge-connected and triconnected. ACM-SIAM Symp. on Discrete Algorithms, 1999, Baltimore, pp.280-289.
[13] J. de Kort(1992), Edge-disjointness in combinatorial optimization: problems and algorithms, Thesis Publishers, Amsterdam 1992.
[14] P. J. M. Laarhoven and A. Aarts(1987), Simulated annealing: theory and applications, Dordrecht, Kluwer Academic Publishers.
[15] S. Lin(1965), Computer solutions of the traveling salesman problem, BSTJ, 44(10), 2245-2269.
[16] C. Lucet, J-F. Manouvrier and J Carlier(2000), Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs, Algorithmica, 27(3-4), 316-36.
[17] C.L. Monma and D.F. Shallcross(1989), Methods for designing communication networks with certain two-connected survivability constraints, Operation Research, 37(4), 531-541.
[18] Y. Myung, H. Kim and D. Tcha(1999), Design of communication networks with Survivability constraints, Management Science, 45(2), 238-252.
[19] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications, INC, Mineola, New York, 1998.
[20] S. Soni, R. Gupta and H. Pirkut(1999), Survivable network design: the state of the art, Information Systems Frontiers, 1(3), 303-315.
[21] K. Steiglitz, P. Weiner, D.J. Kleitman(1969), The design of minimum-cost survivable networks, IEEE Transactions on Circuit Theory, ct-16(4), 455-460.
[22] T.S.Hsu. Graph augmentation and related problems: theory and practice. PhD thesis, University of Texas, Austin, 1993.