The Regenerator Location Problem

Si Chen, *Smith School of Business, University of Maryland, College Park, MD 20742, USA*
S. Raghavan, *Smith School of Business, University of Maryland, College Park, MD 20742, USA*

Keywords: optical network design, heuristics, complexity.

1. Introduction

An optical network provides higher capacity and reduced costs for new applications such as the Internet, video and multimedia interaction, and advanced digital services. However, despite the many advantages an optical network has, it also has limitations. In this paper we address the regenerator location problem (RLP), which deals with the constraint on the geographical extent of lightpaths in optical networks. In RLP, all the nodes are terminal nodes (i.e., generate and receive traffic). A signal can only travel a maximum distance of $d_{\max}$ before its quality deteriorates and needs to be regenerated. To accomplish this regenerators may be installed only at nodes of the network. As the cost of regenerators is very high we wish to deploy as few regenerators as possible, while ensuring all terminal nodes can communicate with each other (i.e., send a signal to each other).

Mathematically, the RLP problem can be described as follows. Given a network $G = \{N, F, D\}$, where N is the set of nodes, F is the set of edges, and D is the associated distance matrix of edges, and a maximum distance of $d_{\max}$ that determines how far a signal can traverse before its quality deteriorates and needs to be regenerated. Determine a minimum cardinality subset of nodes L such that for every pair of nodes in N there exists a path in G with the property that there is no subpath (i.e., a subsequence of edges on the path) with length $> d_{\max}$ without internal regenerators (i.e, we do not include the end points of the subpath).

Although the geographical or physical extent that a signal can travel is an important issue, it seems to have been largely ignored by the academic literature on telecommunications network design. In our literature search, we have only come across two papers [1] and [2] that discuss the issue of regenerator placement within the context of a larger network design paper. In this paper we consider the regenerator location problem as a stand alone problem. We prove that it is NP-complete and discuss high-quality heuristics for it. Since network design in practice is typically done in a hierarchical fashion, we believe that the RLP problem should be addressed at the outset of the network design. Application of our procedures to solve RLP ensures placement of regenerators so that all nodes of the network may communicate without worry of physical impairments of the signal.

The rest of this extended abstract will be organized as follows: Section 2 proves that RLP is NP-Complete; Section 3 discusses heuristics for solving RLP; and Section 4 presents the computational results from our heuristics and compares them with the optimal solutions obtained from an MIP formulation.

2. NP-Completeness of RLP

In this section we prove that RLP is NP-Complete.

Theorem 1. *The regenerator location problem is NP-complete.*

Proof. We consider a special case of Hitting Set Problem (HSP), which is stated as follows [3].
Instance: Collection C of subsets of a finite set S, where $|c| = 2$ for all $c \in C$, positive integer $K \leq |S|$.
Question: Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that S' contains at least one element from each subset in C?
We now construct the corresponding instance of the RLP. Create a node for every $s \in S$. Connect all nodes

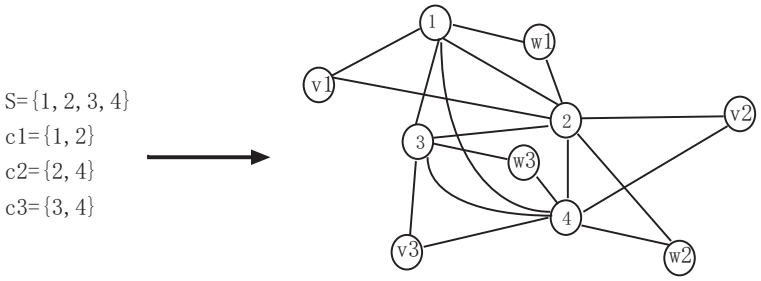

Figure 1: *Transform a HSP to RLP*

in S. For every $c_i = \{c_i^1, c_i^2\}$ in C, create a pair of nodes v_i and w_i. Connect v_i to nodes c_i^1 and c_i^2 (the elements of c_i), and w_i to nodes c_i^1 and c_i^2. Set the length of all edges in the resulting graph equal to $d_{\max}$.

The question is whether there is a feasible solution (a set of nodes L where we place regenerators) to the RLP problem with cardinality less than or equal to K. Observe that in the RLP problem obtained from transforming a HSP, a feasible solution need not place a regenerator at the v_i and w_i nodes. This is due to the fact that the nodes representing the elements in S are fully connected. Thus a feasible solution with a regenerator at a v_i or w_i node remains feasible when the regenerator is removed from that node. With this it is easy to observe that an instance of HSP has a "yes" answer if and only if the corresponding RLP has a "yes" answer. As this is a polynomial transformation, the decision version of the RLP is NP-complete. □

3. Proposed Heuristics

In this section we discuss three heuristics. Before we do, it is useful to consider the following graph transformation of RLP.

Given a graph $G = \{N, F, D\}$, and a maximum distance of $d_{\max}$, apply the all pairs shortest path algorithm. Replace edge lengths by the shortest path distance. If the edge length is less than or equal to $d_{\max}$ then keep the edge, and if the edge is greater than $d_{\max}$ delete the edge. Denote this new graph as M (with node set N and edge set E). Observe then, if M is a complete graph, no regenerators are required. On the other hand every node pair that is not connected by an edge in M requires regenerators to communicate. We call such node pairs "not directly connected" or NDC node pairs. It suffices to consider the RLP problem on the transformed graph M and determine the minimum cardinality subset of nodes L, such that for the NDC node pairs in M there exists a path with regenerators at all internal nodes on the path.

Observe the following property in the graph M. Suppose we place a regenerator at a node t. Then every node pair that is not directly connected in M, but that is connected to t can communicate. Consequently, after the placement of a regenerator at node t, such node pairs can be viewed as being directly connected (i.e., can communicate with each other) and the graph M can be updated with edges between such nodes. In this setting, our objective then is to minimize the number of nodes where regenerators are placed so that M is fully connected (i.e., complete).

We now discuss a preprocessor procedure (to reduce the problem size and fix regenerators in the solution) and a post-optimizer (to improve upon the heuristic solutions) that we apply to all of our heuristics. Observe that if node i is only connected to one other node j, i.e., i has degree one, every feasible solution must include a regenerator deployed at node j. Once a regenerator is deployed at node j, node i can be eliminated from M. After applying the heuristics we apply the following post-optimizer which consists of two subroutines: k**-swap** and **Remove**. k**-swap** tries to swap k (k=1 or 2) regenerator locations in the current solution with locations that are not regenerator locations. If k**-swap** results in a feasible solution, we continue to apply **Remove**. **Remove** simply tries to remove a regenerator location from the current solution.

```
TREE(i)
{
    visit(i) =true;
    For each neighbor of i that is not in S or L, add it to the list C_i;
    If S = ∅, let S = S ∪ {i}; Otherwise let S = S ∪ {i} ∪ C_i;
    While not all the nodes in C_i are visited
    {
        Among all the unvisited nodes in C_i find the node c that has the largest degree,
        D_c, in the graph {N, E\E{S}};
        If D_c > 0
        {
            L = L ∪ {c};
            TREE(c);
        }
        Else, return L;
    }
    Return L;
}
```

Figure 2: *Steps of Tree*(i).

Greedy Heuristic: The greedy heuristic tries to find the node which can eliminate the most NDC node pairs in M if a regenerator is deployed at its location. It keeps looking for such nodes until a feasible solution is reached. Greedy calculates for each node i the number of NDC node pairs ($nr(i)$) it can reduce if a regenerator is added at its location. It takes $O(|N|^2)$ to find $nr(i)$ for one node and $O(|N|^3)$ for all the nodes in the graph. We also need $O(|N|)$ comparisons to find the node with the largest value of $nr(i)$. We add a regenerator to the node, update the graph M, and check the feasibility (i.e., whether it is fully connected). Since we can at most add N regenerators, the complexity of Greedy is $O(|N|^4)$.

Heuristic H1: We make three observations that will be the basis of H1 (and our second heuristic H2). First, a RLP instance is feasible only if the underlying graph is connected. Second, every connected graph has at least one spanning tree. And third, we can obtain a feasible solution from a spanning tree by deploying one regenerator to every non-leaf node. Obviously the fewer non-leaf nodes a spanning tree has the fewer regenerators are needed in the corresponding feasible solution. In H1, we use a subroutine called TREE(i) that tries to construct a spanning tree with as few non-leaf nodes as possible. Heuristic H1 has the following steps.

1. Initialization. $S = \emptyset$, $C_i = \emptyset$ and visit(i) =false $\forall i \in N$.
2. Find the node i that has the lowest degree.
3. Call $V =$ TREE(i).

The pseudo-code for TREE(i) is provided in Figure 2. Let L be the set of chosen regenerator locations. Let $S \subseteq N$ be a subset of N, $E(S) \in E$ be the set of edges that have both end points in S. Also, let C_i be the set of nodes that are connected to node i, and are not in S or L. The intuition behind TREE(i) is that if we add a regenerator to a node with a larger degree in $\{N, E\backslash E\{S\}\}$ (i.e., choose it as a non-leaf node), there is a better chance that we can connect more NDC node pairs. The running time of H1 is $O(|N||E|)$ and its memory requirement is $O(|N|^2)$ (details are left for the full length paper).

We illustrate H1 using a small example shown in Figure 3. In the resulting graph, H1 finds node 1 which has the lowest degree. Thus H1 starts by calling TREE(1). We have $C_1 = \{8, 2\}$, $S = \{1\}$ and $E\{S\} = \emptyset$. The degrees of node 8 and node 2 in $E\backslash E\{S\}$ are both three (we call the degree of nodes in $E\backslash E\{S\}$ the revised degree). Breaking the tie arbitrarily, let H1 choose node 2 and call TREE(2). We have $C_2 = \{7, 8\}$ and $S = \{1, 2, 8, 7\}$. Since the revised degree of node 7 is four, which is greater than that of node 8, we call TREE(7). We have $C_7 = \{4, 5, 6, 3\}$ and $S = \{1, 2, 8, 7, 4, 5, 6, 3\}$. The revised degree of every element of C_7 is zero. Thus H1 stops and returns $V = \{2, 7\}$.

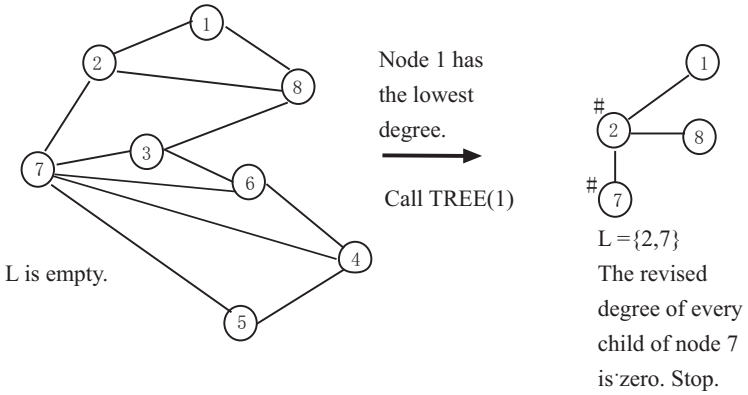

Figure 3: *Apply H1 to a Small Example*

Node(i)
{
Among all the nodes adjacent to node i that are not in L
find the node c that has the largest degree D_c;
If $D_c > 0$
{
Return c;
}
Return $\emptyset$;
}

Figure 4: *Steps of Node*(i).

Heuristic H2: Heuristic H2 has the following steps.

1. Find the node i that has the lowest degree.
2. Call $U =$ Node(i). If $U \neq \emptyset$, set $L = L \cup U$, update the graph and go to 1. Otherwise go to 3.
3. Return L.

The difference between H1 and H2 is that the former constructs the entire spanning tree all at once while the latter starts all over again every time a regenerator is added. In other words, H2 searches the graph for the node with the lowest degree and every time a regenerator is added it updates the graph M by adding edges between nodes that can now communicate with each other (due to the addition of the regenerator). We provide the pseudo-code for Node(i) in Figure 4. The complexity of H2 is $O(|N||E|) + O(|N|^3)$. (details are in the full length paper).

4. Computational Results

We now briefly discuss our computational experience with the three heuristics. We compared our heuristics with the exact solution obtained by applying CPLEX 9.0 (a commercial MIP solver) to an MIP formulation of the problem. For the small-sized instances we are able to obtain the optimal solutions. For the remaining instances, we allow the CPLEX MIP solver to run for 7,000 seconds and record the best lower bounds. All the computational experiments are conducted on a PC with Pentium IV processor at 3.4GHZ and 1G of RAM.

In an attempt to capture the attributes of various networks, we generated networks in three different ways. First, we *randomly generated networks* to directly generate the transformed graph M (i.e., we do not generate

	MIP		H2			GD			H1			
n	LB	RT	BF	NF	RT	BF	NF	RT	BF	NF	RT	Diff
40	4.25	–	5.00	4.50	0.50	4.75	4.50	1.00	5.00	4.75	0.50	0.00
50	6.25	–	8.25	8.00	3.25	7.75	7.75	5.50	9.00	7.50	2.00	1.00
60	6.50	–	9.25	8.25	11.00	9.25	8.25	12.25	11.00	8.75	7.25	1.50
70	11.00	–	13.75	13.00	46.50	13.00	12.75	48.25	18.00	13.50	34.00	1.50
80	10.75	–	14.25	13.25	121.50	13.00	12.75	111.25	16.25	13.25	63.00	1.75
90	7.25	–	9.75	9.25	70.75	9.25	8.50	56.00	12.00	8.75	98.25	1.25
100	8.50	–	10.75	10.50	175.00	10.25	10.00	189.00	14.00	10.50	151.50	1.50

Table 1: Computational Results for Randomly Generated Networks

				MIP		H2			GD			H1			
n	p	a	b	LB	RT	BP	NF	RT	BP	NF	RT	BP	NF	RT	Diff
40	80	25	75	2.00*	973.45	2.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	0.00	0.00
	90	1	100	5.50*	732.25	5.50	5.50	1.00	5.50	5.50	0.00	6.50	5.50	1.00	0.00
		25	75	4.25*	133.33	4.50	4.25	0.50	4.50	4.25	0.00	5.25	4.25	0.50	0.00
50	80	25	75	2.00*	998.55	2.00	2.00	0.00	2.00	2.00	0.25	2.50	2.00	0.00	0.00
	90	1	100	3.50	–	3.75	3.50	0.50	4.00	3.50	1.00	5.00	3.50	0.75	0.00
		25	75	5.00	–	5.50	5.50	0.75	5.50	5.25	2.25	7.00	5.50	1.75	0.25
60	80	25	75	1.50	–	1.50	1.50	0.00	1.50	1.50	0.00	1.50	1.50	0.25	0.00
	90	1	100	3.00*	1385.70	3.33	3.00	0.67	3.33	3.00	1.33	3.00	3.00	0.33	0.00
		25	75	4.00	–	4.00	4.00	1.33	4.33	4.00	2.67	5.00	4.00	1.67	0.00
70	80	25	75	1.50	–	1.50	1.50	0.25	2.00	1.50	0.25	1.50	1.50	0.50	0.00
	90	1	100	2.00	–	2.00	2.00	0.67	2.33	2.00	0.67	2.00	2.00	0.33	0.00
		25	75	3.50	–	4.00	4.00	1.50	4.50	4.00	7.25	4.75	4.25	4.00	0.50
80	80	25	75	1.00	–	1.00	1.00	0.00	2.00	1.00	1.00	1.00	1.00	0.00	0.00
	90	1	100	1.00	–	1.00	1.00	0.00	1.00	1.00	0.33	1.00	1.00	0.00	0.00
		25	75	3.00	–	4.67	4.33	6.00	4.33	4.00	6.67	5.00	4.33	4.67	1.00
90	80	25	75	1.00	–	1.00	1.00	0.00	2.00	1.00	1.00	1.00	1.00	0.00	0.00
	90	1	100	1.00*	32.03	1.00	1.00	0.00	1.00	1.00	0.33	1.00	1.00	0.00	0.00
		25	75	3.25	–	3.75	3.75	7.00	4.00	3.50	13.00	4.75	4.00	11.25	0.25
100	80	25	75	1.25	–	1.25	1.25	0.25	1.50	1.25	1.25	1.25	1.25	0.25	0.00
	90	1	100	1.00*	38.80	1.00	1.00	0.00	1.00	1.00	0.33	1.00	1.00	0.00	0.00
		25	75	2.25	–	3.75	3.50	6.75	4.25	3.00	14.00	4.00	3.50	7.00	0.75

Table 2: Computational Results for Networks with Random Edge Lengths. Parameter p controls the percentage of edges in the complete graph with distances greater than d_{max}. The edge distances for the remaining $(1 - p\%)$ percent edges are uniformly distributed in the range $[a\% \times d_{max}, b\% \times d_{max}]$.

any distances but the graph M). In the other two cases we generated a network and their edge lengths; and from these computed the transformed graph M. In the first of these two cases, we generated edge lengths at random, and in the latter case (*Euclidean Networks*) we used the Euclidean distance between the nodes. Details of the problem generation procedures are described in greater detail in the full length paper.

We now discuss the performance of the three heuristics H1, H2, and the Greedy heuristic (which we refer to as GD from hereon). First, we explain the notation in the Tables. Column "n" is the number of the nodes. "LB" is the lower bound[1] on the number of regenerators produced by CPLEX. An asterisk beside the number indicates an optimal solution. "RT" records the running time in seconds. If CPLEX terminates due to the 7,000 seconds time limit we leave"RT" empty. "BP" is the number of regenerators provided by the heuristics before the post-optimization procedure. "NF" is the number of regenerators after the post-optimization procedure. The last column "Diff" is calculated by comparing the best solution from H1, H2 and GD with the lower bound. Each row in the table corresponds to *averages* from one problem set containing four instances.

[1]We actually round up the bound produced by CPLEX.

n	d_{max}	MIP		H2			GD			H1			Diff
		LB	RT	BP	NF	RT	BP	NF	RT	BP	NF	RT	
40	30	7.75*	4048.08	7.75	7.75	0.50	8.00	8.00	1.00	8.00	8.00	0.75	0.00
	40	4.50	–	4.50	4.50	0.50	5.25	4.50	0.50	5.25	4.50	0.00	0.00
	50	3.00	–	3.00	3.00	0.00	3.25	3.00	0.00	3.25	3.00	0.50	0.00
50	30	6.25	–	7.50	7.00	0.75	8.00	6.75	2.75	7.75	7.25	1.50	0.50
	40	4.00	–	4.50	4.50	0.25	4.75	4.50	1.00	5.25	4.75	0.50	0.25
	50	0.75	–	2.75	2.75	0.00	3.25	2.75	0.50	2.75	2.75	0.25	1.25
60	30	3.25	–	8.75	8.50	6.25	8.75	8.50	8.75	9.75	8.25	7.25	4.25
	40	2.00	–	4.75	4.50	1.25	5.25	4.50	3.50	5.75	4.75	1.75	1.75
	50	3.00	–	3.00	3.00	1.00	3.50	3.00	0.75	3.50	3.00	1.00	0.00
70	30	1.00	–	7.75	7.75	4.75	7.75	7.75	11.00	8.25	8.00	7.75	6.75
	40	1.00	–	5.00	5.00	2.25	5.25	5.00	4.25	6.25	5.00	1.75	4.00
	50	1.00	–	2.75	2.75	1.25	4.00	2.75	2.50	3.25	2.75	1.25	1.75
80	30	1.00	–	8.00	7.50	12.75	7.50	7.25	21.50	8.50	7.25	10.00	6.25
	40	1.00	–	4.50	5.50	3.00	5.50	4.25	7.25	5.50	4.25	4.25	3.25
	50	1.00	–	3.25	3.25	2.00	3.25	3.00	3.50	3.50	3.00	1.00	2.00
90	30	1.00	–	7.50	7.00	30.25	8.00	7.50	52.75	9.00	7.25	22.75	6.00
	40	1.00	–	4.00	4.00	3.75	5.50	4.75	13.50	5.50	4.50	9.00	3.00
	50	1.00	–	3.50	3.25	2.75	3.75	2.75	5.75	3.50	2.75	1.75	1.75
100	30	1.00	–	7.50	7.25	43.50	8.50	8.50	71.00	9.25	9.00	43.50	6.25
	40	1.00	–	4.50	4.50	7.75	5.25	4.75	17.00	5.25	4.25	11.25	3.25
	50	1.00	–	3.25	3.00	6.75	3.75	3.25	11.00	3.75	3.00	4.00	2.00

Table 3: Computational Results for Euclidean Networks

Table 1-3 summarize the computational results for the three types of networks, respectively. There are a total 49 problem sets in the tables. Observe that CPLEX succeeds in finding the optimal solutions for 8 problem sets, for which our heuristics also find the optimal solutions but at a much faster pace. For the remaining problem sets, we compare the heuristic solutions against the lower bounds. In Table 1 and 2, Diff ranges from 0 to 1.75. In Table 3, Diff ranges from 0 to 6.75. The big gap occur when CPLEX fails to even solve the LP relaxation of the problem within the time limit and therefore a value of 1 is used as the lower bound. We now compare the performances of the three heuristics against each other. Observe that at least one of H2 and GD outperforms H1 in all but two of the 49 problem sets. However, there is not a clear dominance between H2 and GD. We suggest practitioners apply H2 and GD in parallel and keep the better solution.

Overall, the computational results demonstrate that our heuristics can find the optimal solutions for small-sized problems much faster than a commercial MIP solver. In addition, they can rapidly find solutions to large-sized problems which a commercial MIP solver failed to solve to optimality within a 7000 second time limit. Based on our comparison of the performance of the three heuristics it appears that H2 and GD outperform H1 for the RLP problem. As part of the future work we wish to find tighter lower bounds that can better evaluate the performances of our heuristics.

References

[1] Luis Gouveia, Pedro Patricio, Amaro F. de Sousa and Rui Valadas, "MPLS over WDM Network Design with Packet Level QoS Constraints based on ILP Modes," *IEEE INFOCOM* 2003.

[2] E. Yetginer and E. Karasan, "Regenerator Placement and Traffic Engineering with Restoration in GMPLS Networks," *Photonic Network Communications*, Vol. 6, no. 2, pp. 139-149, Sept. 2003.

[3] Garey, M. R., D. S. Johnson., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, NY,1979.