

# Reload Cost Trees and Network Design

Ioannis Gamvros, *ILOG, Inc., 1080 Linda Vista Avenue, Mountain View, CA 94043, USA*

Luis Gouveia, *Faculdade de Ciencias da Universidade de Lisboa, Portugal*

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

**Keywords:** reload cost, trees, network design, formulations.

## 1. Introduction

In this paper we consider the notion of “reload costs” in network design. Reload costs occur in telecommunication networks using diverse technologies. For example, one of the products offered by commercial satellite service providers and their partners is a virtual private network (VPN) that can offer voice, video and data connectivity between all of the geographically dispersed locations of large corporate, government and military organizations. Typically, these VPNs are made up of satellite links *and* fiber optic cables. The use of diverse technologies at different junctions of the VPN results in an extra cost component (i.e., in addition to typical routing costs) that is associated with the equipment required to seamlessly bind them together. These interface costs are referred to as *reload* costs and depend on the technologies being connected. Moreover, these costs can sometimes dominate other costs such as the regular routing costs.

Reload costs can appear under many different contexts. In the telecommunications industry any network design that incorporates different technologies (i.e., fiber, copper, radio links etc.) will contain reload costs. Even in cases where the technology remains the same but there are many different telecommunications providers that participate in the complete network, switching between the networks of different providers might entail reload costs. In the transportation industry *intermodal* freight is a fast growing and very successful business. In these transportation networks the unloading of freight from one type of carrier to another results in significant reload costs. In the energy industry reload costs can capture the losses associated with the interfaces used to transfer energy from one type of carrier to another. For example during the transportation of natural gas we might have to convert it from a liquid to a gas state or vice versa. This conversion introduces losses which have to be taken into consideration since they represent a significant cost component.

Given that reload costs occur naturally in many settings, we are motivated by the desire to develop “good models” for reload cost network design problems. In this extended abstract we discuss reload cost tree network design problems. Formally, we are given a graph  $G_R = (V_R, E_R)$ , a color  $\mathcal{C}(i, j)$  for each edge  $\{i, j\} \in E_R$  (the colors represent different technologies in the telecommunications industry context), a per unit of flow reload cost  $R_{nm}$  for each pair of colors  $(n, m)$ , and a set of demands between all nodes in  $V_R$ . We wish to build a tree network that spans the nodes in  $V_R$  and has the minimum total reload cost. We call this problem the Minimum Reload Cost Spanning Tree (RCST) problem. For the moment, we ignore routing costs. Later, we explain how to incorporate them within our models.

Despite their apparent usefulness in modeling complex cost structures in both the telecommunications and transportation industry, reload costs have not been studied extensively in the literature. Specifically, the only paper in which reload costs appear is by Wirth and Steffan [1] who introduce a minimum diameter spanning tree with reload costs. In their problem we are given a graph in which edges have different labels (colors) and the reload costs between all of the different labels (colors). We wish to build a tree network that spans all the nodes in the graph but has the smallest possible diameter with respect to the reload costs (i.e., we wish to minimize the maximum reload cost between any two nodes in the network). The authors show that the minimum diameter reload cost spanning tree problem is NP-hard for graphs with an arbitrary node degree. They also present an approximation algorithm for graphs with maximum node degree 5 and an exact algorithm for graphs with maximum node degree 3.

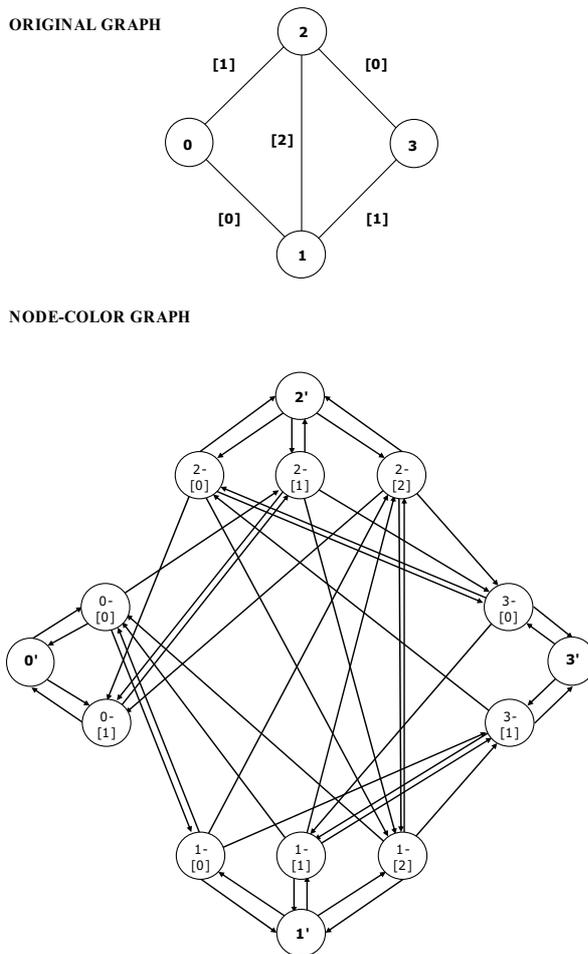


Figure 1: A small graph and the associated node-color graph.

## 2. A Node-Color Graph Formulation

Since reload costs occur when there is a change in color on the edges on a route, a natural formulation is a quadratic optimization model (i.e., one with a quadratic objective function and linear constraints).<sup>1</sup> A natural way to linearize such a model is to expand the graph building a so called directed line graph. In a directed line graph two nodes  $(i, j)$  and  $(j, i)$  are created to represent each edge  $\{i, j\}$  in the original graph. Node  $(i, j)$  in the directed line graph is connected to nodes of the type  $(j, l)$  with an arc directed from node  $(i, j)$  to node  $(j, l)$ . The reload cost associated with going from edge  $\{i, j\}$  to edge  $\{j, l\}$  in the original graph is associated with this arc. This approach is detailed in Gamvros [4]. However, this graph grows quite rapidly, and its size is not a function of the number of colors in the problem! Specifically, for an undirected colored graph  $G_R = (V_R, E_R)$  the associated directed line graph,  $G_L = (V_L, A_L)$ , contains  $|V_L| = |V_R| + 2|E_R|$  nodes and  $|A_L| = 4|E_R| + \sum_{i \in V_R} deg(i)(deg(i) - 1)$  arcs, where  $deg(i)$  is the degree of node  $i$ . Motivated by the desire to develop a linearized formulation on a more compact graph we discuss a formulation for the reload cost problem on an extended graph that we call the *node-color* graph and denote as  $G_C = (V_C, A_C)$ .

<sup>1</sup>In this regard we should point out that a closely problem related to the RCST is the Quadratic Spanning Tree (QST) problem [2]. In the QST we wish to build a minimum cost tree that spans the nodes of a graph. However, the costs provided are associated with pairs of edges as opposed to single edges (the special case in which only adjacent pairs of edges have non-zero costs is called the adjacent-QST). Notice that the distinction between the costs in the QST and reload costs is that the former are fixed costs associated with the selection of edges whereas the latter are variable (per-unit of flow) costs associated with flow on the edges. Exactly, the same distinctive difference exists between the classical Minimum Spanning Tree (MST) problem and the Optimal Communications Spanning Tree (OCST) problem [3].

In Figure 1 we present an example of a simple graph and its associated node-color graph. Notice that in the original graph the labels on the edges indicate colors (e.g., “[0]”, “[1]” etc.). The labels of the nodes in the node-color graph indicate the node in the original graph and the adjacent color represented by that node. For example the label “2-[1]” represents the version of node 2 that is associated with color 1. The other nodes in the node-color graph represent copies of the nodes in the original graph and are labeled accordingly. For example node  $0'$  represents node 0 in the original graph. Essentially, node “2-[1]” represents the fact that we reach node 2 from an edge of color 1.

Formally, a node-color graph  $G_C = (V_C, A_C)$  of a graph  $G = (V_R, E_R)$  can be constructed as follows. The node set of the node-color graph,  $V_C$  includes nodes  $i_n$  for each node  $i \in V_R$  and each color  $n \in \mathcal{C}(i)$  that is adjacent to node  $i$ . Notice that we refer to a color being adjacent to a node, if that node is adjacent to an edge of that color and we denote the set of colors adjacent to a node  $i$  as  $\mathcal{C}(i)$ . The node set of the color graph  $V_C$  also includes copies of the nodes of the original graph just like the line graph did. For each edge  $\{i, j\}$  of the original graph the arc set  $A_C$  contains multiple arcs  $(i_n, j_m)$  for all  $n \in \mathcal{C}(i)$  (i.e., the colors of  $i$ ), to node  $j_m$  where  $m$  is the color of the edge  $\{i, j\}$  on the original graph, (i.e.,  $m = \mathcal{C}(i, j)$ ). Observe, that an arc  $(i_n, j_m)$  is associated with the color pair  $(n, m)$ . In other words a commodity flowing on arc  $(i_n, j_m)$  is using edge  $\{i, j\} \in E_R$  of color  $m$  immediately after using an edge of color  $n$ . Therefore arc  $(i_n, j_m)$  is associated with a very specific reload and is therefore assigned the reload cost involved in using colors  $n$  and  $m$  consecutively. We note that for an undirected colored graph  $G_R = (V_R, E_R)$  the associated directed node-color graph  $G_C = (V_C, A_C)$  contains  $|V_C| = |V_R| + \sum_{i \in V_R} cdeg(i)$  nodes and  $|A_C| = \sum_{i \in V_R} 2cdeg(i) + \sum_{\{i, j\} \in E_R} (cdeg(i) + cdeg(j))$  arcs, where  $cdeg(i)$  is the color degree of node  $i$  (i.e., the number of colors adjacent to the node).

In our formulation, we use decision variables  $w_{\{i, j\}}$ , to indicate whether edge  $\{i, j\}$  is selected or not and we also define new decision variables  $z_{i_n, j_m}^{s, t}$  that indicate the proportion of flow from  $s$  to  $t$  that uses arc  $(i_n, j_m)$  of the node-color graph. Notice that the variables  $z_{i_n, j_m}^{s, t}$  are defined only for  $m = \mathcal{C}(i, j)$  since all arcs from  $i$  to  $j$  are headed to the version of node  $j$  that corresponds to the color of the edge  $\{i, j\}$ . We now present an arc-flow formulation on the node-color graph.

$$(CGFB) \quad \min \sum_{(s, t): s < t} \sum_{(i, j) \in A_R} \sum_{n \in \mathcal{C}(i)} c_{i_n, j_m} z_{i_n, j_m}^{s, t}$$

subject to

$$\sum_{i: (i, j) \in A_R} \sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{s, t} - \sum_{i: (j, i) \in A_R} z_{j_m, i_n}^{s, t} = o_{j_m}^{s, t}, \quad \forall (s, t) : s < t, j_m \in V_R, m \in \mathcal{C}(j), \quad (1)$$

$$\sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{s, t} + \sum_{n \in \mathcal{C}(j)} z_{j_n, i_m}^{s, t} \leq w_{\{i, j\}}, \quad \forall (s, t) : s < t, \{i, j\} \in E_R, \quad (2)$$

$$\sum_{\{i, j\} \in E_R} w_{\{i, j\}} = |V| - 1, \quad (3)$$

$$w_{\{i, j\}} \in \{0, 1\}, \quad \forall \{i, j\} \in E_R, \quad (4)$$

$$z_{i_n, j_m}^{s, t} \geq 0, \quad \forall (s, t), i_n, j_m \in V_R, n \in \mathcal{C}(i), m = \mathcal{C}(i, j), \quad (5)$$

where  $o_{j_m}^{s, t}$  is equal to  $-1$  when  $j = s$ , equal to  $1$  when  $j = t$  and zero otherwise. Constraint (1) ensures flow conservation and constraint (2) ensures that flow can only use edges that have been selected. Constraint (3) forces the number of edges that are selected to be exactly  $|V_R| - 1$  and together with the rest of the constraints ensures the design of a spanning tree. In the CGFB model the underlying shortest paths between nodes of the original graph correspond to shortest paths on the node-color graph.

In the context of Uncapacitated Network Design, Balakrishnan et al. [5] present a way to strengthen the forcing constraints (2), that are associated with the design variables and the flow over them. The idea they present is that when edge  $\{i, j\}$  is selected then all commodities flowing to node  $a$  will flow either from  $i$  to  $j$  or from  $j$  to  $i$ . We model this situation for a *limited* combination of commodities and edges, with constraints (6) and (7). Constraint (6) is defined for commodities that flow to node  $a$  and constraint (7) complements the earlier ones for commodities that originate at node  $a$ . Note that these constraints are used in the CGFB model

Problem Set	Primal	IP-LP Gap (%)	Time (s)
N5E10C3	0.00	0.00	0.025
N10E25C3	3.70	44.00	0.137
N10E25C5	9.65	10.79	0.137
N10E25C7	23.67	8.72	0.206
N15E50C3	3.48	128.70	4.125
N15E50C5	28.50	14.32	2.038
N15E50C7	49.96	17.04	3.419
N15E50C9	71.17	11.80	4.231
N20E100C3	0.00	0.00	132.519
N20E100C5	4.94	417.41	115.187
N20E100C7	26.48	61.58	48.125
N20E100C9	58.40	23.19	43.897
Aggregate	16.26	54.40	21.84

Table 1: LP relaxation results for the CGFB model with the forcing constraints of set 1.

in addition to the existing forcing constraints.

$$\sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{i,a} + \sum_{n \in \mathcal{C}(j)} z_{j_n, i_m}^{j,a} \leq w_{\{ij\}}, \quad \forall a \in V_R, \{i, j\} \in E_R, \quad (6)$$

$$\sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{a,j} + \sum_{n \in \mathcal{C}(j)} z_{j_n, i_m}^{a,i} \leq w_{\{ij\}}, \quad \forall a \in V_R, \{i, j\} \in E_R. \quad (7)$$

In order to account for all combinations of commodities and edges we introduce constraints (8) and (9). These new constraints define forcing restrictions in the spirit of the earlier constraints for all commodities originating and terminating at node  $a$ . We can therefore replace constraint (2) by constraints (8) and (9).

$$\sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{s,a} + \sum_{n \in \mathcal{C}(j)} z_{j_n, i_m}^{t,a} \leq w_{\{ij\}}, \quad \forall (a, s, t) \in V_R : \{s < a, t < a\}, \{i, j\} \in E_R, \quad (8)$$

$$\sum_{n \in \mathcal{C}(i)} z_{i_n, j_m}^{a,s} + \sum_{n \in \mathcal{C}(j)} z_{j_n, i_m}^{a,t} \leq w_{\{ij\}}, \quad \forall (a, s, t) \in V_R : \{a < s, a < t\}, \{i, j\} \in E_R. \quad (9)$$

**Routing Costs:** We note that it is easy to include routing costs, since routing costs can be incorporated as part of the reload costs as follows. The cost coefficient in the objective function can then be defined as  $c_{i_n, j_m} = R_{nm} + U_{ij}$  where  $R_{nm}$  is the reload cost associated with going from an edge of color  $n$  to an edge of color  $m$ , and  $U_{ij}$  is the per unit of flow routing cost on edge  $\{i, j\}$ . As a result the routing costs can be considered as part of the reload cost in the model and do not have to be treated separately.

### 3. Computational Results

In this section we will first explore the strengths of the reload cost tree problem formulations and evaluate the benefits of the additional forcing constraints we presented earlier. We then discuss avenues for future research.

Our computational work was conducted on a set of randomly generated problem instances with varying characteristics. All graphs were generated on a 100x100 grid. The endpoints for the edges were randomly picked among the nodes in the graphs and the color for each edge was drawn from a uniform distribution. In the tables that follow we identify each set as “NxEyCz” where  $x$  denotes the number of nodes,  $y$  the number

Problem Set	Primal	IP-LP Gap (%)	Time (s)	Improvement (%)
N5E10C3	0.00	0.00	0.031	0.00
N10E25C3	5.24	44.00	0.206	28.82
N10E25C5	10.93	10.79	0.209	9.19
N10E25C7	24.64	8.72	0.287	4.16
N15E50C3	6.69	128.70	11.209	66.70
N15E50C5	31.62	14.32	4.181	9.37
N15E50C7	53.92	17.04	7.353	9.33
N15E50C9	74.62	11.80	9.516	4.81
N20E100C3	0.00	0.00	474.784	0.00
N20E100C5	13.42	417.41	404.428	239.01
N20E100C7	36.51	61.58	208.913	47.94
N20E100C9	65.43	23.19	149.390	11.63
Aggregate	18.42	8.54	76.02	30.95

Table 2: LP relaxation results for the CGFB model with the forcing constraints of set 2. The “improvement” column represents average percentage improvement over the set 1 constraints.

of edges and  $z$  the number of colors. Each set consists of 5 random problem instances. Unless otherwise noted all the reload costs between all combinations of colors were set equal to 1. For our computational work we generated problems where the number of nodes and edges in the graph varied between 5 and 20 and between 10 and 100, respectively. Also, we increase the number of different colors in a graph from 3 to 9. The formulations were solved with CPLEX 9.0 on a Windows PC with two Pentium Xeon processors at 2.66 GHz with 2 GBytes of RAM. We first compare the percentage gaps between the LP relaxation of the various versions of the CGFB model with optimal solutions. We will refer to the set of forcing constraints (2) with which the CGFB model was presented initially as “set 1”. Also, we refer to constraints (6) and (7) in addition to constraint (2) as “set 2”. Additionally, “set 3” denotes the use of constraints (8) and (9).

Table 1 presents the average primal bound of the LP relaxation of the CGFB model with the forcing constraints of set 1 for various problem sets. The table also presents the average percentage IP-LP gap calculated as the difference between the values of the LP bound and the optimal integer solution, over the value of the LP bound. The last column in the table shows the average running time of the LP relaxation in seconds. In a similar fashion, Table 2 presents the solutions of the LP relaxation of the CGFB model with the use of the forcing constraints of set 2. The table presents the average primal solution, the average percentage IP-LP gap and the average running time for the relaxation. The last column in this table provides the average percentage improvement of the LP bound from the CGFB model with constraint set 1. This improvement is calculated as the difference between the LP bound with set 1 and set 2 over the value of the LP bound with set 1. Table 3 has exactly the same structure as Table 2 and provides solution information for the CGFB model with constraint set 3. The only difference is that the average percentage improvement presented is over the set 2 solutions.

From the information presented in these tables it is clear that the forcing constraints associated with sets 2 and 3 can provide improvements over the original model. However, notice that these improvements come with a penalty in terms of the computation time required. These computational penalties are a direct consequence of the increased number of constraints in the various formulations. For example consider a problem with 15 nodes and 50 edges. For this problem, the number of forcing constraints in Set 1 is equal to 5, 250. For set 2 we add an extra 3, 000 constraints and reach a total of 8, 250 constraints. For set 3 we consider triplets of nodes, and as a result the number of constraints is 124, 000. From this example it should be clear that the significant increase in the size of the formulation is causing the very large computation times observed. Based on the improvement percentages and computation times presented in Table 3 for problems with 15 nodes one can argue that the extra running time associated with set 3 outweighs the benefits introduced. Consequently, we have not tested constraint set 3 on problems with more than 15 nodes. Also, notice that for graphs with a specific number of nodes and edges, increasing the number of different colors seems to make the problems easier. This is particularly clear in Table 2. Even though at first this might seem counterintuitive observe

Problem Set	Primal	IP-LP Gap (%)	Time (s)	Improvement (%)
N5E10C3	0.00	0.00	0.028	0.00
N10E25C3	5.42	5.20	2.800	1.77
N10E25C5	11.20	0.00	1.009	1.39
N10E25C7	25.25	1.91	2.097	2.20
N15E50C3	8.01	15.16	3782.663	11.77
N15E50C5	32.33	2.85	2812.165	1.47
N15E50C7	56.06	2.80	4655.703	4.14
N15E50C9	76.87	3.47	2437.356	3.00
Aggregate	17.93	3.23	1,141.15	2.77

Table 3: LP relaxation results for the CGFB model with the forcing constraints of set 3. The “improvement” column represents average percentage improvement over the set 2 constraints.

that in the extreme case in which the number of different colors equals the number of edges in a graph all trees are associated with the same total reload cost and therefore all trees are optimal. Of course the other extreme, in which all edges have the same color also presents a trivial case, where any tree would again be an optimal solution. We note that since the linear relaxations of our directed line graph formulation and color graph formulations are equivalent the linear relaxations provide identical objective function values. However, as expected the running times of the node-color graph model are much faster than the line-graph for all three sets of forcing constraints.

**Concluding Remarks:** In this paper we motivated the notion of reload costs, that occurs in telecommunications, transportation, and energy networks. Additionally, reload costs are theoretically significant since they can be thought of as an extension to the Optimum Communications Spanning Tree (OCST) problem in the same way that the Quadratic Spanning Tree (QST) problem is an extension to the MST. Despite their wide applicability reload costs have only recently been treated in the literature [1]. In this paper we looked at the reload cost tree problem and developed a node-color graph model that can solve it exactly. Gamvros [4] discusses several other classic network design problems where traditional costs are replaced by reload costs and showcase the applicability of our approaches in all these cases.

Our computational experiment focuses on the strength of the LP relaxation of the reload cost tree problem that motivated this work. Our strongest model results in an LP relaxation that is on average only 3.23% from optimality. The extended graphs that we developed allow us to assign reload costs to specific edges rather than pairs of edges. Moreover, each path in the original graph is associated with a specific path on the node-color graph. As a result it is natural to think that a path-based model (and the use of branch-and-price) for reload cost problems in which the pricing problem is solved on the node-color graph might possibly result in a faster solution procedure. This is part of our present work.

## References

- [1] H. C. Wirth and J. Steffan, “Reload cost problems: Minimum diameter spanning tree,” *Discrete Applied Mathematics*, Vol. 113, pp. 73-85.
- [2] A. Assad and W. Xu, “The quadratic minimum spanning tree problem,” *Naval Research Logistics*, Vol. 39, no. 3, pp. 339-417.
- [3] T. C. Hu, “Optimum Communication Spanning Trees,” *SIAM Journal on Computing*, Vol. 3, no. 3, pp. 188-195.
- [4] Ioannis Gamvros, *Satellite Network Design, Optimization, and Management*, Ph.D. Thesis, University of Maryland, College Park, Aug. 2006.
- [5] A. Balakrishnan, T. L. Magnanti, and R. T. Wong, “A dual-ascent procedure for large scale uncapacitated network design,” *Operations Research*, Vol. 37, no. 5, pp. 716-740.