Constrained Rerouting in Networks: An Integer Programming Formulation

W. Weyerman, B. Durtschi, and R. Bhandari

Laboratory for Telecommunications Sciences
8080 Greenmead Drive
College Park, Maryland 20740, USA *

October 10, 2011

Abstract

In this paper, we address the problem of changing link weights within an OSPF network to reroute a single demand over a given network link (or node) while maintaining the routes for all other demands. It is often not feasible to require that no other routes change, so we resort to infeasibility analysis to find the smallest set of routes that must change in order to achieve our main objective of rerouting the single demand. We formulate an integer program (IP) that finds new link weights for the network that achieves this goal. We then propose optimal iterative methods of reducing the number of constraints and decision variables in this program. To arrive at a computationally feasible method, we then show a linear relaxation in conjunction with some variants of the iterative method to reduce computation time. We validate our algorithm by comparing the variants to each other and to the sliding shortest path algorithm, which was developed for a similar problem.

1 Introduction

In telecommunication networks, it is often desirable to change the link weights to engender path changes in a certain desired fashion. For example, a high capacity link might suddenly become available over which the traffic between the given source-destination pair of an important customer might need to be rerouted to alleviate congestion on the current route, and maintain the desired quality of service as warranted, perhaps, by a service-level agreement.

*Questions or comments may be directed to Sam Weyerman at `wsweyerman@gmail.com` or Ramesh Bhandari at `drbhandari617@gmail.com`.

The problem of establishing (or changing) routes within a network by assigning (or altering) link weights has received immense attention in the recent past, see, e.g., Ref. [9, 10, 16]. Fortz and Thorup [11] address the NP-hard problem of weight-setting for a given set of demands in an Open Shortest Path First (OSPF) environment, and subsequently consider the problem of changing weights [12] to avoid congested links; the emphasis in [12] is on making as few link weight changes as possible to minimize the messaging chaos and the network congestion time. Ref. [1, 4, 5, 16] also attempt to find the administrative weights so that a given set of paths (one path per demand) is realized by the standard OSPF routing protocol, but they treat the problem as an uncapacitated problem. In particular, Bley [5] goes further in evaluating the inherent difficulty of the problem posed by Ben-Ameur and Gourdin [1].

More recently, Bhandari [3] posed the problem of rerouting a single demand, with the requirement that the number of links on which the weights are changed be as small as possible in order to reduce the implementation time of the link weight changes within the OSPF environment; it was further required that the weight changes (increments) be as small as possible in order to reduce the possibility of shortest paths for the other demands within the network changing. Bhandari [3] treated the problem as an uncapacitated problem and provided a simple, polynomial-time heuristic called the Sliding Shortest Path Algorithm to solve it. While the algorithm addressed the above two requirements of the problem, there was no direct control over the number of paths of other demands in the network changing on account of the link weight changes.

In this paper, we address a slightly different, but more challenging, problem of not only modifying link weights to reroute a single demand such that the route contains a desired link or vertex not already on the shortest path, but also to minimize the number of routes of other demands that may change on account of the link weight modifications. We assume that a simple path, including the desired link or vertex, exists. Excluding the diagonal elements that are always zero, a demand matrix in general can have several entries that are zeros on account of no traffic flow for the corresponding pairs of nodes. Instead of performing an analysis for every possible demand matrix (having a different set of zero entries), we adopt the general case of no entries being a zero (other than the diagonal elements) within the demand matrix; i.e., there is traffic flowing between every pair of nodes within the network. The constraint that no other routes change is almost always infeasible, so we resort to minimizing the number of routes that change using infeasibility analysis. Because this problem, as those related to it, appears to be NP-hard, we seek heuristics. The requirement of integrality of link weights leads to an integer formulation of the problem which is then solved via its linear relaxation. The problem is treated as an uncapacitated problem.

The paper provides a detailed theoretical as well as numerical analysis of this problem. In Section 2, we develop and describe the problem formally, and in Section 3, we discuss some methods of reducing the problem size without losing optimality. In Section 4, we present some polynomial time heuristics, and discuss their performance in Section 5.

2 Problem Formulation

We wish to reroute the shortest path between two nodes over a particular link (or node) while minimizing the number of other shortest paths that change. We will consider weighted, undirected graphs of the form $G = \{V, E, w\}$. The set of vertices of the graph is denoted V with $|V| = n$. An edge is denoted $(i,j) \in E$ with $i,j \in V$ and the number of edges is $|E| = m$. We consider graphs where the weights of the edges, w, are positive integers, thus $w = \{w_e \in \mathbb{N}_+ | e \in E\}$ where $\mathbb{N}_+ = \{1, 2, \ldots\}$. The notation w_e is used to denote the weight of edge $e \in E$. A path, P, from i to j, $i,j \in V$ is denoted $P(i,j)$, and the length or cost of the path is $|P|_\ell = \sum_{e \in P} w_e$. As we are dealing with routing networks, the only paths we deal with are simple paths. To denote the shortest path from i to j, we will write $P^*_G(i,j)$. When the context is clear we will leave out the source and destination, (i,j), the symbol G for graph, or both.

Suppose we are given a pair of nodes $s,t \in V$, and an edge $(p,q) \in E$ such that there is at least one simple path in G from s to t that contains the edge (p,q). While we explicitly treat the case of rerouting a single demand over a given edge (p,q) in this paper, we note without proof that the fomalism of the paper also holds for the problem of rerouting over a single node, p, by considering a simple path from s to t that contains node p.

We wish to find a modified graph $G' = \{V, E, w'\}$, where $w'_e = w_e + x_e$, with $(p,q) \in P^*_{G'}(s,t)$; that is, the edge (p,q) is contained in the shortest path from s to t in the modified graph. We continue to impose positive edge weights, $w'_e \in \mathbb{N}^m_+$, which implies that $x_e \in \mathbb{Z}, x_e > -w_e$, where $\mathbb{Z}$ is the set of integers. Furthermore, we wish to minimize the amount of chaos in the network, so we want to maximize the number of i,j pairs such that the unique shortest path $P^*_G(i,j) = P^*_{G'}(i,j)$. The set of all shortest paths excluding the desired path, the path from s to t, in a graph is $\mathcal{P}^*$. For a given shortest path, $P^* \in \mathcal{P}^*$, we will denote the set of all of the simple paths that are not the shortest path, or alternative paths, $\mathcal{A}^{P^*}$. For a given pair i,j, we write the shortest path restriction as a set of inequalities

$$\sum_{e \in P^*_G} w'_e < \sum_{d \in A} w'_d, \qquad \forall A \in \mathcal{A}^{P^*_G}. \tag{2.1}$$

which we can expand to

$$\sum_{e \in P^*} x_e - \sum_{d \in A} x_d < |A|_\ell - |P^*|_\ell \qquad \forall A \in \mathcal{A}^{P^*}. \tag{2.2}$$

Suppose for some $P^* \in \mathcal{P}^*$, $\left|\mathcal{A}^{P^*}\right| = s$, we can enumerate $\mathcal{A}^{P^*}$ as $\{A_1, \ldots, A_s\}$. The inequalities can then be written in matrix form as $Cx < b$, with $C \in \mathbb{R}^{s \times m}$, $x \in \mathbb{R}^m$ and $b \in \mathbb{R}^s$. A row of C specifies which edges belong to the shortest path and which belong to the alternative path in a given comparison. The vector x represents the changes in edge weights, and b is the difference in the weight of the shortest path and the alternative path. We set

$$c_{kl} = \begin{cases} 1 & e_l \in P^* \text{ and } e_l \notin A_k \\ -1 & e_l \notin P^* \text{ and } e_l \in A_k \\ 0 & \text{otherwise.} \end{cases}$$

The values of b are $b_k = |A_k|_\ell - |P^*|_\ell - 1$.

Since our work is motivated by the OSPF routing protocol where the minimum edge weight is 1, all edge weights must be positive. We will show that this allows us to, without loss of generality, consider only positive changes to the edge weights.

Lemma 1. *Suppose $C \in \mathbb{R}^{n\times m}$, $b, x, w \in \mathbb{R}^n$, with $Cw = -b$, $Cx \leq b$, and $-w < x$. Then there is some $x' \in \mathbb{R}^n$, $0 \leq x'$ such that $Cx' \leq b$.*

Proof. Let $C \in \mathbb{R}^{n\times m}$, $b, x, w \in \mathbb{R}^n$, with $Cw = -b$, $Cx \leq b$, and $-w < x$. Because $-w < x$, for each x_i there is some $\alpha_i \in \mathbb{R}$ such that $|x_i| = \alpha_i(w_i + x_i)$. We will pick $\alpha = \max_i(\alpha_i)$ and let $x' = \alpha w + (1+\alpha)x$. Then we have

$$\begin{aligned} C(x') &= \alpha Cw + (1+\alpha)Cx \\ &\leq -\alpha b + (1+\alpha)b \\ &= b. \end{aligned}$$

We conclude by noting that for any $i \leq n$,

$$\begin{aligned} \alpha w_i + (1+\alpha)x_i &= x_i + \alpha(w_i + x_i) \\ &\geq x_i + \alpha_i(w_i + x_i) \\ &= x_i + \|x_i\| \\ &\geq 0. \end{aligned}$$

Hence, x' satisfies the conditions of the lemma. □

We previously showed how we can represent a set of path constraints as a linear relationship $Cx < b$. This representation will be used to apply Lemma 1 to show that any set of shortest paths that can be achieved by allowing weights to change negatively can also be achieved while only allowing positive weight changes.

Theorem 1. *Consider two graphs $G = \{V, E, w\}$ and $G' = \{V, E, w'\}$ with $w, w' \in \mathbb{N}_+^m$. There exists some other graph $G^+ = \{V, E, w^+\}$ with each $w_e^+ \geq w_e$ and $P^*_{G'}(i,j) = P^*_{G^+}(i,j)$ for every pair of vertices $i, j \in V$.*

Proof. Let $G = \{V, E, w\}$ and $G' = \{V, E, w'\}$. We will define $x = w' - w$. Then $-w = x - w' < x$. For each pair of vertices, $i, j \in V$, we construct a set of (satisfied) inequalities of the form

$$|P^*_{G'}|_{\ell(w'))} < |A|_{\ell(w')} \qquad A \in \mathcal{A}^{P^*_{G'}}, \tag{2.3}$$

where the lengths are with respect to the graph G'. We have previously shown that these can be written as a set of linear inequalities of the form $Cx \leq b$ with x as defined above and $b_k = |A|_{\ell(w)} - |P^*_{G'}|_{\ell(w)}$. With this definition, we have

$$\begin{aligned} [Cw]_k &= |P^*_{G'}|_{\ell(w)} - |A|_{\ell(w)} \qquad (2.4) \\ &= -b_k. \qquad (2.5) \end{aligned}$$

We have shown that the conditions for Lemma 1 are satisfied, so there is some $x^+ \geq 0$ such that $Cx^+ \leq b$. Hence we can define $w^+ = w + x^+$ to arrive at the desired G^+. □

In Theorem 1 the graph G is used simply as a point of reference. The theorem states that for any graph and any set of minimum edge weights, a graph with equivalent shortest paths can be constructed with edge weights greater than or equal to the minimum weights. The graph G is used to define these minimum edge weights. In our problem, we use our original graph as the point of reference, G, and leverage this theorem to reduce our search space to only positive changes. This is done because if there is a feasible graph with negative changes, we know by Theorem 1 that we can construct a graph with only positive changes that is also feasible.

We pose this topology reconfiguration problem as an integer program. Our development is similar to that of [1, 16]. We will minimize the sum of edge changes, although other objective functions may be used. Our constraints will be that the path from s to t contain edge (p,q) and that no other paths change. However, when changing the path from s to t, there will be paths that will be required to change. We can easily characterize some of these paths.

We will denote the set of all paths from s to t that include edge (p,q), $\mathcal{M} = \{\mu(s,t) | (p,q) \in \mu(s,t)\}$. For a path $\mu \in \mathcal{M}$, the subpath consisting of the path from s to p or q will be denoted μ^1, whichever arrives in the path first, and the subpath consisting of the path from p or q to t as μ^2, we start this path at the vertex p or q that comes later in $\mu(s,t)$. Hence μ^1 will contain s and either p or q, and μ^2 will contain t and either p or q (whichever is not in μ^1), and the concatentaion of these paths, $\{\mu^1, \mu^2\}$ is $\mu(s,t)$. For example, consider again the graph in Figure 1. For this graph $\mu = \{s,a,p,q,t\}$, so $\mu^1 = \{s,a,p\}$ and $\mu^2 = \{q,t\}$. If the shortest path between two other nodes, $i,j \in V$ passes through both μ^1 and μ^2, such as $\{a,t\}$, it is required to change in order for $\mu(s,t)$ to be the shortest path. We can generalize this as a theorem.

Theorem 2. *Consider a graph $G = \{V,E,w\}$, such that $\mu(s,t) = \{\mu^1,\mu^2\}$ with $u,v \in V$ and $u \in \mu^1$, $v \in \mu^2$, and $u,v \in P^*(i,j)$. If the shortest path $P^*(i,j) \not\subseteq \mu(s,t)$ and $\mu(s,t) \neq P^*(s,t)$, then for any graph $G' = \{V,E,w'\}$ with $\mu(s,t) = P^*_{G'}(s,t)$, $P^*_G(i,j) \neq P^*_{G'}(i,j)$.*

Proof. Let $G = \{V,E,w\}$ with $i,j \in V$, such that there is some $u \in \mu^1$ and $v \in \mu^2$ and $u,v \in P^*(i,j)$ and the shortest path $P^*_G(u,v) \not\subseteq \mu(s,t)$. Furthermore, suppose that $\mu(s,t) \neq P^*(s,t)$. Also, let $G' = \{V,E,w'\}$ be a graph with positive weights such that $\mu(s,t) = P^*_{G'}(s,t)$. Consider the path $P(i,j)$ with the same path from i to u and from v to j as $P^*(i,j)$, but with the path from u to v being that of $\mu(s,t)$. It follows that $P^*_{G'}(u,v)$ is the path contained in $\mu(s,t)$, and since $P(i,j)$ and $P^*(i,j)$ differ only over this path, $|P(i,j)|_{\ell(w')}) < |P^*(i,j)|_{\ell(w')}$, so the shortest path from i to j must change. □

The paths that satisfy the assumptions of Theorem 2 must be excluded from our constraints. These paths are generally few in number. We call the paths that satisfy Theorem 2, $\mathcal{P}^*_e$; this set may be different for different μ's. We then define

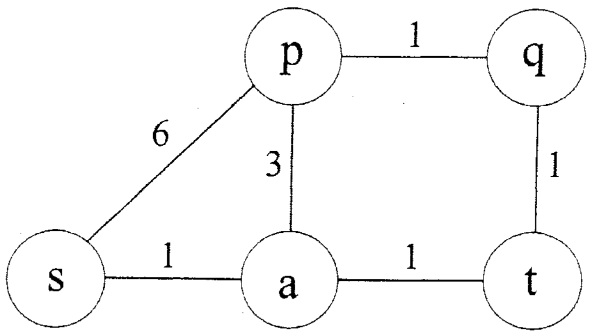


Figure 1: This graph gives an example of when the shortest path from s to t over (p, q) requires more paths to change than a longer path from s to t.

the set of constrained paths $\mathcal{P}_0^* = \mathcal{P}^* \setminus \mathcal{P}_e^*$. The other paths are not as easily determined as discussed in [1, 16].

Picking μ is not a simple task, as the shortest path that contains the desired edge, (p, q), may not be the best choice because rerouting the network traffic over the shortest path may require more paths to change than a longer path. It may be necessary to route the traffic over a higher cost path in order to minimize the number of other paths changed. Consider the graph in Figure 1. The shortest path from s to t over edge (p, q) is $\{s, a, p, q, t\}$. Forcing the shortest path to be this one requires adding 5 to the weight of edge (a, t). This will change the following 3 paths: s to q, a to t, and a to q. Another solution, increasing the weight of edge (s, a) by 7 only changes 2 shortest paths: s to p and s to q. Hence, to find the minimum number of shortest paths changed, alternate paths from s to t must be considered and not only the shortest path traversing edge (p, q). We formulate a two-stage optimization problem to first find the μ which requires the smallest set of paths to change, and then determine the edge weight changes to reroute the demand over that path.

We first determine the μ that satisfies

$$\min_{\mu(s,t) \in \mathcal{M}} |\mathcal{P}_e^*(\mu)| .$$

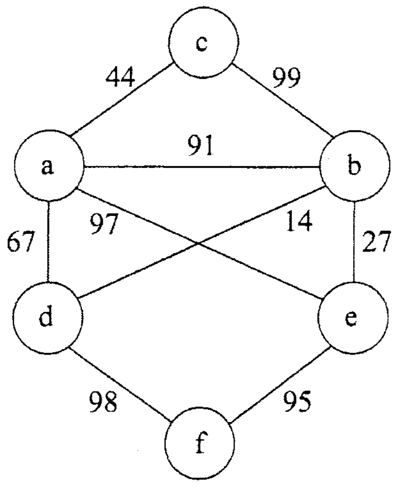


Figure 2: This graph results in infeasibility in problem (2.6).

Using this choice for μ, we solve

$$\text{minimize} \quad \sum_{e\in E} x_e \tag{2.6a}$$

$$\text{subject to} \quad \sum_{e\in\mu} x_e - \sum_{d\in P(s,t)} x_d < |P(s,t)|_\ell - |\mu|_\ell$$

$$P(s,t) \neq \mu \tag{2.6b}$$

$$\sum_{e\in P^*} x_e - \sum_{d\in A} x_d < |A|_\ell - |P^*|_\ell$$

$$P^* \in \mathcal{P}_0^*,\ A \in \mathcal{A}^{P^*} \tag{2.6c}$$

$$x_e \in \{0,1,2,\ldots\}$$

$$e \in E.$$

2.1 Infeasibility

This problem is often infeasible. An example of a graph that results in infeasibility is given in Figure 2. In this network, let $s = a$ and $t = f$ with the desired edge (e, f). In this graph, $\mu(a, f) = \{a, e, f\}$, with $\mu^1 = \{a, e\}$, and $\mu^2 = \{f\}$, and

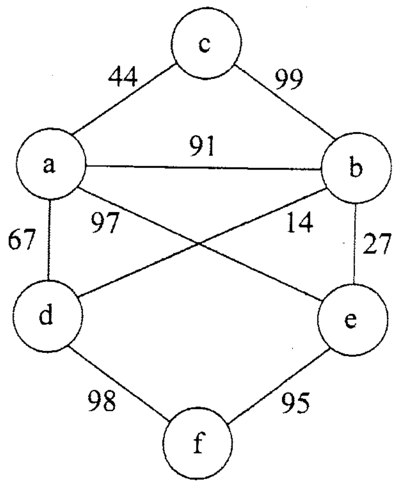


Figure 2: This graph results in infeasibility in problem (2.6).

Using this choice for μ, we solve

$$\text{minimize} \quad \sum_{e \in E} x_e \tag{2.6a}$$

$$\text{subject to} \quad \sum_{e \in \mu} x_e - \sum_{d \in P(s,t)} x_d < |P(s,t)|_\ell - |\mu|_\ell$$

$$P(s,t) \neq \mu \tag{2.6b}$$

$$\sum_{e \in P^*} x_e - \sum_{d \in A} x_d < |A|_\ell - |P^*|_\ell$$

$$P^* \in \mathcal{P}_0^*, \; A \in \mathcal{A}^{P^*} \tag{2.6c}$$

$$x_e \in \{0, 1, 2, \ldots\}$$

$$e \in E.$$

2.1 Infeasibility

This problem is often infeasible. An example of a graph that results in infeasibility is given in Figure 2. In this network, let $s = a$ and $t = f$ with the desired edge (e, f). In this graph, $\mu(a, f) = \{a, e, f\}$, with $\mu^1 = \{a, e\}$, and $\mu^2 = \{f\}$, and

Size	10	25	50	75	100	150
No. Infeasible	5	34	74	85	93	97

Table 1: Number of trials out of 100 that problem (2.6) was infeasible for varying network sizes with $\delta = 3$.

$P^*(s,t) = \{a, d, f\}$. An infeasible subset of the path constraints is

$$\begin{aligned}
x_{a,e} + x_{e,f} - x_{a,d} - x_{d,f} &< -w_{a,e} - w_{e,f} + w_{a,d} + w_{d,f},\\
x_{a,c} + x_{a,d} - x_{b,c} - x_{b,d} &< -w_{a,c} - w_{a,d} + w_{b,c} + w_{b,d},\\
x_{b,d} + x_{d,f} - x_{b,e} - x_{e,f} &< -w_{b,d} - w_{d,f} + w_{b,e} + w_{e,f},\\
x_{b,c} + x_{b,e} - x_{a,c} - x_{a,e} &< -w_{b,c} - w_{b,e} + w_{a,c} + w_{a,e}.
\end{aligned}$$

The sum of all four constraints gives

$$0 < 0.$$

Which is clearly a contradiction. Relaxation of any one of these constraints would result in a feasible problem.

We have also seen that as the size of the network grows, so does the probability of infeasibility. As shown in Table 1, the probability of infeasibility using this method quickly approaches 1 as the size of the network grows over 100 nodes. These results are obtained using random graphs of varying size up to 150 nodes, with average degree, δ, near 3. For each network size, we attempted to solve problem (2.6) for 100 randomly generated graphs and report the number of times the problem was infeasible.

2.2 Constraint Relaxation

To solve (2.6), we want to find the smallest set of constraints to relax to make the problem feasible. We will solve a problem closely related to infeasibility analysis discussed in [17, 6, 7, 15] where the goal is to determine the minimal set of infeasible constraints. We follow an approach using binary elasticity variables to allow constraints to be violated and impose a penalty for doing so and minimize the total penalty incurred.

$$\begin{aligned}
\text{minimize}_{v,\mu} \quad & \sum_{P^* \in \mathcal{P}_0^*} v_{P^*} + |\mathcal{P}_e^*(\mu)| && \text{(2.7a)} \\
\text{subject to} \quad & \sum_{e \in \mu} x_e - \sum_{d \in P(s,t)} x_d < |P(s,t)|_\ell - |\mu|_\ell \\
& \qquad P(s,t) \neq \mu && \text{(2.7b)} \\
& \sum_{e \in P^*} x_e - \sum_{d \in A} x_d < |A|_\ell - |P^*|_\ell + \eta v_{P^*} \\
& \qquad P^* \in \mathcal{P}_0^*, A \in \mathcal{A}^{P^*} && \text{(2.7c)} \\
& x_e \in \{0, 1, 2, \ldots\} \\
& \qquad e \in E \\
& v_{P^*} \in \{0, 1\} \\
& \qquad P^* \in \mathcal{P}_0^*.
\end{aligned}$$

Here $\eta \in \mathbb{N}_+$ is some number large enough to render the constraints where $v_{P^*} = 1$ non-binding. We use the solution of this problem to define the set of constraints for (2.6). We define the set $\mathcal{P}_r^* = \{P^* | P^* \in \mathcal{P}^*, v_{P^*} = 1\}$, and solve (2.6) using the set of paths $\mathcal{P}_0^* \setminus \mathcal{P}_r^*$.

3 Optimal Constraint Reduction

The number of constraints in problem (2.7) makes it computationally difficult not only to solve the problem, but to construct it as well. From combinatorics, it is known that $|\mathcal{P}^*| = \frac{(n)(n-1)}{2} - 1$. Furthermore, for each path $P^* \in \mathcal{P}^*$, there are several constraints in the form of equation (2.7c). In the worst case (a fully connected graph),

$$\left|\mathcal{A}^{P^*}\right| = \sum_{i=1}^{n-2} \frac{(n-2)!}{((n-2)-i)!}.$$

If every path is enumerated as a constraint, then the number of constraints in the IP is exponential in the size of the graph. We will explore methods of reducing the number of constraints and call the included alternative paths $\mathcal{A}_0^{P^*} \subset \mathcal{A}^{P^*}$. The collection of all included alternative paths is $\mathcal{A}_0 = \bigcup_{P^* \in \mathcal{P}^*} \mathcal{A}_0^{P^*}$.

3.1 Determination of $\mathcal{A}_0$

For a path $P^* \in \mathcal{P}^*$, we propose two methods to determine $\mathcal{A}_0^{P^*}$. Although there are many alternative paths, those that are significantly longer than the shortest path are not likely to represent a binding constraint. It is not known a priori which alternative paths will be binding so we propose two iterative methods for adding constraints. These methods differ in the initial setup of constraints, but the iterative method is the same. For initialization we only add a subset of each $\mathcal{A}^{P^*}$ to our initial constraint paths, $\mathcal{A}_0^{P^*}$. This may allow the shortest path to change without penalty, so we perform an iterative method similar to the one described in [16]. After solving the problem at iteration k, each $P^* \in \mathcal{P}^*$ with $v_{P^*} = 0$ must be checked to insure that the shortest path remains P^*. If any shortest path P^* changed, the new shortest path is added to $\mathcal{A}_{k-1}^{P^*}$ to arrive at $\mathcal{A}_k^{P^*}$. We also add P^* to a new set, $\mathcal{C}_k$, which is the set of paths that changed in iteration k. The problem is then solved again with the new constraints. If no paths changed, then the algorithm terminates. By adding the iterative method, the solution remains optimal because no paths are allowed to change without penalty. We desire a balance between the number of constraints in the mathematical program with the number of iterations necessary to solve the problem.

3.1.1 Fixed Number of Constraints Per Path

For the first technique, $\mathcal{A}_0^{P^*}$ is generated by adding a fixed number of constraints for each path. This technique adds constraints that the shortest path remain shorter than only the K shortest paths in $\mathcal{P}^{P^*}$. The problem of finding the K shortest simple paths has been studied widely and polynomial time algorithms are given in [8, 14, 18] among others. Thus, for each P^*, $\mathcal{A}_0^{P^*}$ is initialized to be the set containing only the K shortest paths, excluding the shortest path. We then employ the iterative process.

3.1.2 Constraints Based on Path Weight

The second technique is a more targeted approach which results in varying cardinality of $\mathcal{A}_0^{P^*}$, even allowing $\mathcal{A}_0^{P^*} = \emptyset$ for some paths. A path is seen as likely to change if the weight of an alternative path is close to the weight of the shortest path. For this technique, an alternate path $A \in \mathcal{A}^{P^*}$ is added to $\mathcal{A}_0^{P^*}$ only if

$$\frac{|P^*|_\ell}{|A|_\ell} \leq r, \tag{3.1}$$

where $r \in \mathbb{R}$ is some predetermined ratio. We have found that values between 1.05 and 1.3 seem to work well. These alternative paths are as easy to generate as the k shortest paths. The same algorithms can be used by changing the stopping condition to be that of equation (3.1) instead of number of paths found. As with the previous method, this may allow some paths to change without penalty, so we employ the same iterative process.

3.2 Desired Path Constraint Reduction

The desired path is not already the shortest path, so there may be many paths that are shorter than the desired path. Thus, a slightly different method is used to initialize these path constraints. To reduce the number of constraints on the desired path, only a subset of the alternate paths that are shorter than the desired path are added. First, the shortest path from s to t is added to $\mathcal{A}_0^\mu$ while maintaining the set of edges $\mathcal{E}$ that are in at least one path in $\mathcal{A}_0^\mu$. Each subsequent path is only added to $\mathcal{A}_0^\mu$ if there are enough unique edges in the path, i.e. the ratio of edges in the path that are in $\mathcal{E}$ to the total number of edges in

Algorithm 3.1 Iterative Optimal Method

```
Require: $G, s, t, p, q$
 1: for all $\mu \in \mathcal{M}$ do
 2:   $(\mathcal{C}_0, \mu, \mathcal{P}_0^*, \mathcal{A}_0) \leftarrow$ initialConstraints$(G, s, t, p, q)$
 3:   $k \leftarrow 0$
 4:   while $\mathcal{C}_k \neq \emptyset$ do
 5:     (status,$G'$) $\leftarrow$ solve$(\mu, \mathcal{P}_0^*, \mathcal{A}_k)$ {using (2.6)}
 6:     if status is infeasible then
 7:       $\mathcal{R}_k \leftarrow$ relax$(\mu, \mathcal{P}_0^*, \mathcal{A}_k)$ {using (2.7)}
 8:       $\mathcal{P}_{k+1}^* \leftarrow \mathcal{P}_0^* \setminus \mathcal{R}_k$
 9:       $G' \leftarrow$ solve$(\mu, \mathcal{P}_{k+1}^*, \mathcal{A}_k)$ {using (2.6)}
10:     else
11:       $\mathcal{P}_{k+1}^* \leftarrow \mathcal{P}_0^*$
12:     end if
13:     $(\mathcal{C}_{k+1}, \mathcal{A}_{k+1}) \leftarrow$ addConstrs$(G', \mathcal{P}_{k+1}^*, \mathcal{A}_k, \mathcal{C}_k)$
14:     $k \leftarrow k + 1$
15:   end while
16:   if $|\mathcal{P}_{k+1}^*| < |BestP|$ then
17:     $BestP \leftarrow \mathcal{P}_{k+1}^*$
18:     $BestG' \leftarrow G'$
19:   end if
20: end for
21: return $BestG'$
```

the path is small enough. For example, let P be a path from s to t such that $|P|_\ell < |\mu|_\ell$. Suppose that the P has b hops, and κ edges are in both P and $\mathcal{E}$. We will include the path if

$$\frac{\kappa}{b} < \rho,$$

where $0 < \rho < 1$. We then iteratively apply this method while the desired path is not the shortest path.

We have shown some methods of reducing the number of constraints in problem (2.7), while still arriving at an optimal solution. We show these methods in conglomerate as an algorithm in Algorithm 3.1. The last hindrances to the solution of this problem are the integer constraints on the decision variables and the need to solve these IPs for all $\mu \in \mathcal{M}$. The relaxation of the integer constraint, and any consequent integerization method, will result in a sub-optimal or perhaps incorrect solution. Furthermore, solving for only a subset of $\mathcal{M}$ may result in a sub-optimal solution. We will describe some heuristic methods of relaxing the problem to achieve sub-optimal solutions.

4 Linearization and Rounding

Because this problem formulation is an IP, it is intractable and, hence, unsolvable for moderate to large sized graphs. Because of this, we seek heuristics. One such method, the sliding shortest path (SSP) algorithm, is given in [3]. While this method is computationally fast, its aim is to modify the smallest number of edge weights in the graph rather than minimize the amount of turmoil. Therefore, we will seek heurisitics based on the methods given in Section 3 and compare these methods to the SSP.

To deal with the size of $\mathcal{M}$, we only solve the problem for the shortest path, $\mu \in \mathcal{M}$, such that for any $\mu' \in \mathcal{M}$, $|\mu|_\ell \leq |\mu'|_\ell$. A polynomial time algorithm to find this path is given in [2]; this requires finding the shortest vertex-disjoint paths connecting s to p (or q) and t to q (or p). A K-shortest vertex-disjoint paths algorithm would allow for the solution over the K shortest $\mu \in \mathcal{M}$.

To arrive at a polynomial time algorithm, we linearize problems 2.6 and 2.7 and round the linearized solution to arrive at an integer solution. In order to minimize rounding errors, we reduce the right hand side of each constraint by a fixed value $\alpha \in \{1, 2, \ldots\}$. This forces each shortest path to be at least α shorter than the next shortest path. We call the linearized solution method the Iterative Linear Elasticity (ILE) method.

This algorithm requires $m + \frac{n(n-1)}{2}$ variables each time the relaxation is solved. Although we are solving a linear program, this is a large number of variables and the solutions may take a considerable amount of time for large graphs. We propose two further relaxations to the problem to reduce the run time. First is the greedy ILE (G-ILE). In the greedy ILE when a path is relaxed, it is never put back into the problem. Hence we modify line 8 of Algorithm 3.1 to be $\mathcal{P}^*_{k+1} = \mathcal{P}^*_k \setminus \mathcal{R}_k$ and use $\mathcal{P}^*_k$ in place of $\mathcal{P}^*_0$ on lines 5 and 7.

We can further reduce the number of decision variables by considering only the most recently added constraints for relaxation. For this modification, we change constraint (2.7c) to be the following two constraints:

$$\begin{aligned}
&\sum_{e \in P^*} x_e - \sum_{d \in A} x_d < |A|_\ell - |P^*|_\ell - \alpha \\
&\qquad\qquad P^* \in \mathcal{P}^*_k \setminus \mathcal{C}_k, A \in \mathcal{A}^{P^*}_k, \\
&\sum_{e \in P^*} x_e - \sum_{d \in A} x_d < |A|_\ell - |P^*|_\ell - \alpha + v_{P^*} \\
&\qquad\qquad P^* \in \mathcal{C}_k, A \in \mathcal{A}^{P^*}_k.
\end{aligned} \tag{4.1}$$

We call this method the restricted ILE (R-ILE) method. Relaxation of only a subset of the relaxable constraints opens up the possibility of infeasibility, which we discuss next.

4.1 Infeasibility

The R-ILE method may result in infeasibility during the relaxation phase. Infeasibility can result if constraints are added to the desired path as well as to other

Algorithm 4.1 addConstrs for restricted ILE

```
Require: $G', \mathcal{P}^*_{k+1}, \mathcal{A}_k, \mathcal{C}_k$
  $(\mathcal{C}_{desired}, \mathcal{A}_{k+1}) \leftarrow$ addViolatedConstrs$(G', \mu, \mathcal{A}_k)$
  if $\mathcal{C}_{desired} \neq \emptyset$ then
    return $(\mathcal{C}_k, \mathcal{A}_{k+1})$
  end if
  $(\mathcal{C}_{k+1}, \mathcal{A}_{k+1}) \leftarrow$ addViolatedConstrs$(G', \mathcal{P}^*_{k+1}, \mathcal{A}_k)$
  return $(\mathcal{C}_{k+1}, \mathcal{A}_{k+1})$
```

paths in the same iteration. This is because these path constraints are the only constraints that are not allowed to be relaxed. If the previous iteration resulted in a feasible solution and constraints are added which cause the problem to become infeasible, relaxation of all newly added constraints will result in a feasible solution (namely the solution obtained at the previous iteration). Hence, if a feasible relaxation does not exist, there must be some previously added constraint that conflicts with a new constraint on the desired path, resulting in infeasibility. To prevent this, we propose an extension to the addConstrs algorithm such that if any constraints are added to the desired path, we do not add any other constraints. Furthermore, in the event that an infeasible solution is reached, the most recently added constraints on other paths are allowed to be relaxed. Our add constraints algorithm is given in Algorithm 4.1.

Conjecture 1. *Unless it is not possible to make the desired path the shortest path, using Algorithm 4.1 will always result in a feasible solution.*

Proof. Suppose we have a graph $G = \{V, E, w\}$, and $s, t, p, q \in V$, $(p, q) \in E$, such that there is a simple path from s to t over the edge (p, q). We will show that the ILE method using Algorithm 4.1 to add new constraints will not result in infeasibility. The only way to reach infeasibility using the ILE method is for the LRR method to be infeasible and then to not have a feasible relaxation. At iteration k of the algorithm, we will denote $\mathcal{A}_k$ as all constrained alternate paths, $\mathcal{C}_k$ the path constraints most recently added to paths other than the s to t path which are relaxable this iteration, and $\mathcal{A}_t$ as all constraints on the path from s to t added since the last iteration that constraints were added to a different path.

On the first iteration, $\mathcal{C}_0 = \{P^* | P^* \in \mathcal{P}^*_0, \mathcal{A}_0^{P^*} \neq \emptyset\}$. If the LRR problem is infeasible, then all constrained paths other than $\mu(s,t)$ may be relaxed in ILE problem. If all paths in $\mathcal{C}_0$ are relaxed, we are left with only the constraints on the path from s to t, $\mathcal{A}^{\mu}$. The edge weights can be changed to move the shortest path over μ, so the relaxation is feasible.

On iteration k, there are two cases. Either on iteration $k-1$ constraints were added to paths other than μ, or on iterations $k-z$ through $k-1$ constraints were added to μ. In either case, the LRR problem was feasible at the end of iteration $k-1$.

Suppose constraints were added to paths $\mathcal{C}_{k-1}$ on the previous iteration. Prior to these constraints being added, we know the LRR problem was feasible. There-

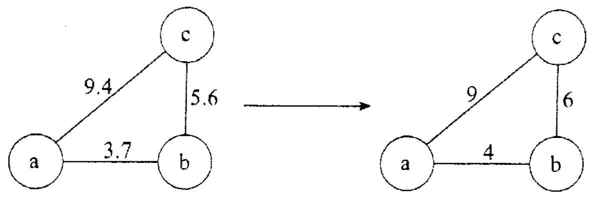


Figure 3: This graph gives an example of rounding changing the shortest path. Before rounding, the shortest path from a to c is $\{a, b, c\}$. After rounding, the path changes to $\{a, c\}$.

fore, since the relaxation of all constraints on the paths in $\mathcal{C}_{k-1}$ results in a less constrained problem than that at iteration $k-1$, the ILE problem is feasible.

Suppose constraints were added to μ during iterations $k-z$, through $k-1$. This is the first case we have considered where $\mathcal{A}_t \neq \emptyset$. If $z = k$, then we have $\mathcal{C}_k = \mathcal{C}_0$, and we have the first case where relaxation of all constraints on paths other than μ results in a feasible solution. If $z < k$, then some constraints were added to paths other than μ at iteration $k-z-1$, and $\mathcal{C}_k = \mathcal{C}_{k-z}$. At the end of iteration $k-z-1$, $\mu(s,t)$ was the shortest path from s to t, otherwise constraints would have been added to μ. Hence all constraints in $\mathcal{A}_t$ were satisfied. Therefore, if we relax all constraints in $\mathcal{C}_k$, then we arrive at a linear program similar to that at iteration $k-z-1$, excepting that there are more constraints on the μ. Because there was a feasible solution with $\mu(s,t)$ as the shortest path from s to t, none of these extra constraints were violated. Therefore the same solution is feasible if all constraints on paths in $\mathcal{C}_k$ are relaxed. $\square$

4.2 Sub-optimal and Incorrect Solutions

Although the solution obtained from the linear program is optimal for continuous variables, when rounded, it may lose optimality. This results in a sub-optimal, and perhaps incorrect solution because shortest paths may change when rounding weight changes. We say the problem is incorrect if the desired path from s to t does not contain edge (p,q) and suboptimal if after rounding other path constraints are violated. An example of this is given in Figure 3. The parameter α is intended to reduce the number of paths changed due to rounding. For example, in Figure 3, suppose we force path $\{a, b, c\}$ to have a length at least 2 greater than path $\{a, c\}$. This corresponds to a value of $\alpha = 2$. To do this, w_{ac} must change to be 11.3 which is longer than the shortest path length of 9.3 by just 2. Rounding off in this case preserves the shortest path. As the graph gets larger, α should also increase, since there are likely to be more edges in the shortest paths, and thus larger errors without the slack variable.

We also lose optimality when rounding problem (2.7). This method puts a lower cost on a small relaxation of several paths than a large relaxation on a single path. We also round the edge weight changes, so rounding errors may occur as described above. In the greedy ILE, once a set of path constraints is relaxed, they are not reinstated at a later iteration. In the restricted ILE only a subset of the possible paths for relaxation are considered each iteration, it may be the case that at an iteration of the algorithm, relaxing one path that was added several iterations ago may result in a feasible solution, but more than one of the newly added paths must be relaxed in order to achieve feasibility.

5 Experimental Results

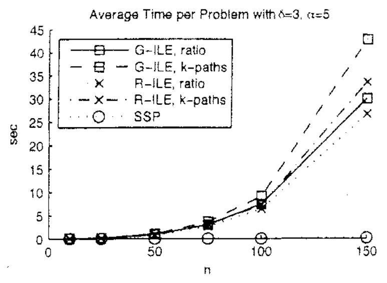
For the experimental results, we test two variations on the ILE algorithm, the greedy ILE (G-ILE) and the restricted ILE (R-ILE). For each we tested adding K constraints per path (with $K = 3$) and adding only the paths with a weight within a ratio of 1.1 of the shortest path. We will compare these to the results given by the SSP algorithm. We tested the algorithms using varying values for the size of the network, $n \in \{10, 25, 50, 75, 100, 150\}$, the average degree, $\delta \in \{2, 3, 5, 8\}$, and the linear programming value, $\alpha \in \{1, 2, 5, 10\}$. For our baseline values, we set $n = 100$, $\delta = 3$, and $\alpha = 5$. For each set of experiments, we fixed two of these values and allowed the other to range over the allowed set. For each combination of α, δ, n, we ran 100 trials. The metrics we compare are runtime, number of paths changed, number of edges required to change, and rate of incorrect solutions. The results are shown in Figures 4, 5, and 6.

We use randomly generated networks with a fixed number of nodes and a target connectivity. To generate an n-node graph with average degree δ, we first build a random n-node tree. We randomly assign an integral value to each edge weight, chosen from 1 to $1 + \delta * n/4$. Once the tree was built, further edges were added in accordance with the desired target average degree. It was ensured that no spurs (nodes with degree 1) were left within the constructed graph by adding an edge to connect each node with degree 1 to a randomly chosen node.

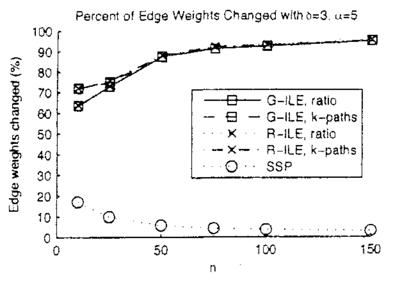
In Figure 4 are the results of varying the number of nodes in the network. From these figures, we see that the ILE algorithm in all forms has a much longer run time than the SSP algorithm. Furthermore, we can see the quadratic run time with respect to the number of nodes. The ratio initialization method resulted in better runtimes than the K-shortest path method while not greatly affecting the number of paths changed. Also, the restricted version of the ILE had a faster runtime than the greedy version while only taking a small penalty in the number of paths changed. We also see that the ILE algorithms performed much better than SSP in terms of number of paths changed, but required that most of the edges in the graph have their weights increased. This makes the topology reconfiguration a difficult task.

In Figure 5 are the results of varying the target average degree, δ. These graphs indicate a linear growth in the runtime of the ILE algorithm with respect to the number of edges in the graph. The percent of edge weights required to change using the ILE algorithm remains high while the number of paths changed is still much lower than SSP. The number of incorrect solutions seems to be independent of the number of edges in the graph.

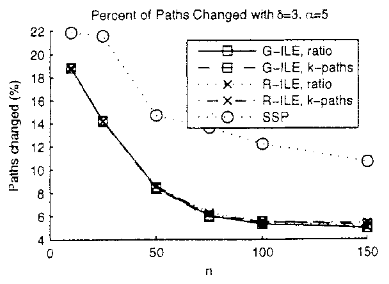
Figure 6 contains the results of varying α. These results indicate that there is a trade-off between the probability of getting an incorrect solution and the



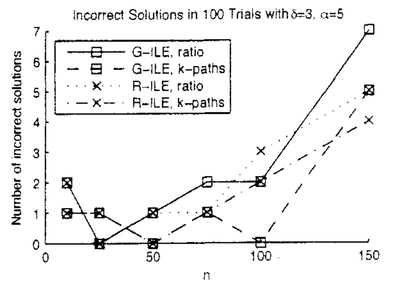
(a) Run time per problem. Each data point is averaged over 100 trials.



(b) Percent of edge weights changed. Each data point is averaged over 100 trials.



(c) Percent of paths changed. Each data point is averaged over 100 trials.



(d) Number of solutions out of 100 trials that were incorrect.

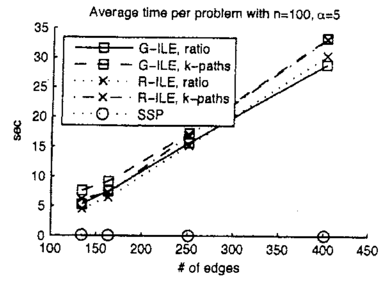
Figure 4: Plotted results allowing the number of nodes, n to vary. For these experiments we fix $\delta = 3$, $\alpha = 5$, and we vary $n \in \{10, 25, 50, 75, 100, 150\}$

number of paths that change. We suspect the apparent minimum of the number of paths changed around the value of $\alpha = 2$ is due to the increasing difficulty of meeting the path constraints as α increases. This 'difficulty' is also apparent in the number of edge weights that are required to change as α increases.

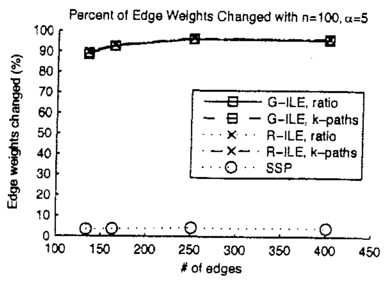
The totality of the results confirm that our proposed algorithm does indeed reduce the number of paths that are changed when re-routing traffic between two nodes. This comes at the expense of higher runtime and most of the edge weights needing to be changed when compared to the SSP algorithm. The parameter α did what it was intended and reduced the number of incorrect solutions. This improvement came at the price of more paths and edges changed.

6 Conclusion

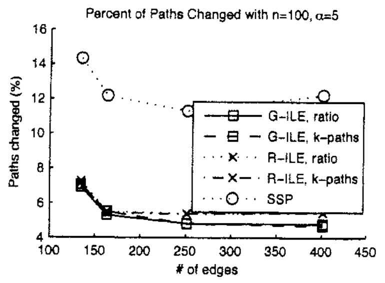
In this paper, we considered the problem of modifying the weights in an OSPF network to reroute a single demand over a given link (or node) while maintaining as many routes for all other demands as possible. We formulated an IP that gives



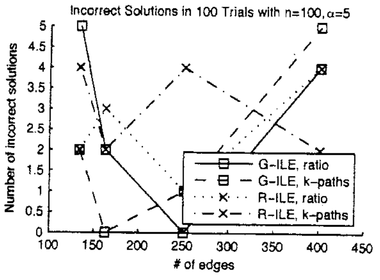
(a) Run time per problem. Each data point is averaged over 100 trials.



(b) Percent of edge weights changed. Each data point is averaged over 100 trials.



(c) Percent of paths changed. Each data point is averaged over 100 trials.



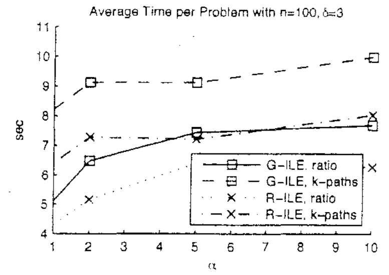
(d) Number of solutions out of 100 trials that were incorrect.

Figure 5: Plotted results allowing the target average degree, δ to vary. For these experiments we fix $n = 100$, $\alpha = 5$, and we vary $\delta \in \{2, 3, 5, 8\}$

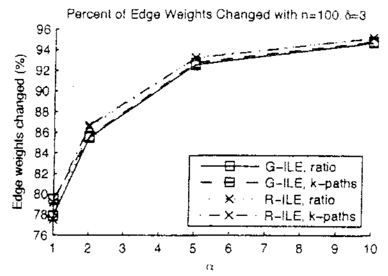
an optimal solution to this problem. Since the mathematical programming representation has an exponential number of constraints with respect to the number of vertices in the graph, we gave some iterative methods to reduce the number of constraints. We also proposed a linear program with variants that give sub-optimal solutions to this problem. We then discussed the reasons for sub-optimality and sometimes incorrect solutions. We compared these methods to a known method, the Sliding Shortest Path method, and experimentally showed that the new algorithms changed significantly fewer other shortest paths in the network, but require that many more edges be changed and take significantly longer to solve.

7 Future Work

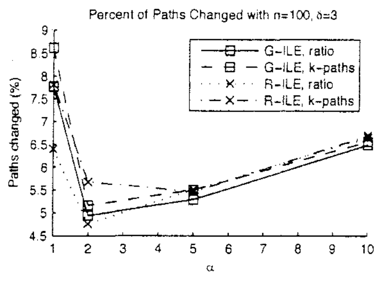
The solution to this problem is the weight changes that minimize the number of routes that change. It was shown in [13] how to change the topology in OSPF networks to avoid routing cycles. However, this does not indicate how to make



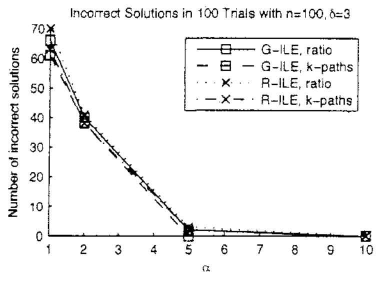
(a) Run time per problem. Each data point is averaged over 100 trials.



(b) Percent of edge weights changed. Each data point is averaged over 100 trials.



(c) Percent of paths changed. Each data point is averaged over 100 trials.



(d) Number of solutions out of 100 trials that were incorrect.

Figure 6: Plotted results allowing the parameter α to change. For these experiments we fix $n = 100$, $\delta = 3$, and we vary $\alpha \in \{1, 2, 5, 10\}$. We do not include the SSP algorithm in these results because it does not use the parameter α.

major changes to the topology while minimally affecting the routing tables. It is likely that many shortest paths that G and G' have in common will change to another path during the transition period and then change back. Effective implementation of the solution requires a method of arriving at these changes that also minimizes the number of paths that change.

The experimental results showed that these algorithms require a large percentage of the edge weights to change. This also makes the topology changes difficult. The SSP method had significantly worse results for the number of paths that change, but had a small number of edge weights to change. It would be of interest to determine the relationship between number of edge weights changed and number of paths affected. This could be used to generate a more refined algorithm that considers the trade off between topology changes and path changes. Further algorithm development that attempts to simultaneously minimize both of these metrics may shed further light upon this relationship.

We also have no guarantees on the performance of these heuristic methods. Other non-convex search procedures may also be applied to this problem such as genetic algorithms or hill climbing algorithms.

Due to the fact that we need to compute the solution for each path from s to t over (p,q), we need a method that allows the enumeration of these paths. It does not seem reasonable to attempt the solution on every such path, but a K-shortest paths method for node-disjoint pairs would allow for the enumeration of the first K of these paths.

References

[1] W. Ben-Ameur and E. Gourdin. Internet routing and related topology issues. *SIAM Journal on Discrete Mathematics*, 17(1):18–49, January 2004.

[2] R. Bhandari. *Survivable Networks: Algorithms for Diverse Routing.* Springer, 1999.

[3] R. Bhandari. The sliding shortest path algorithm. In *Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, pages 97–101, Paris, France, 2009.

[4] A. Bley. Finding small administrative lengths for shortest path routing. In *Proceedings of the 2nd International Network Optimization Conference*, pages 121–128, Lisbon, Portugal, 2005.

[5] A. Bley. Inapproximability results for the inverse shortest path problem with integer lengths and unique shortest paths. *Networks*, 50(1):29–36, August 2007.

[6] N. Chakravarti. Some results concerning post-infeasibility analysis. *European Journal of Operational Research*, 73(1):139–143, 1994.

[7] J. Chinneck and E. Dravnieks. Locating minimal infeasible constraint sets in linear programs. *ORSA Journal on Computing*, 3(2):157–168, Spring 1991.

[8] E. de Queirós V. Martins and M. Pascoal. A new implementation of yen's ranking loopless paths algorithm. *4OR: A Quarterly Journal of Operations Research*, 1(2):121–133, June 2003.

[9] M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization*, 6:299–333, 2002.

[10] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40:118–124, 2002.

[11] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. of the 19th IEEE INFOCOM*, pages 519-528, Tel-Aviv, Israel, 2000.

[12] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20:756-767, 2002.

[13] P. Francois, M. Shand, and O. Bonaventure. Disruption free topology reconfiguration in OSPF networks. In *Proceedings of 2007 IEEE INFOCOM*, pages 89–97, Anchorage, Alaska, USA, May 2007.

[14] J. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Trans. Algorithms*, 3(4):45, November 2007.

[15] K. Murty, S. Kabadi, and R. Chandrasekaran. Infeasibility analysis for linear systems, a survey. *Invited Paper for the Arabian Journal of Science and Technology, Special Issue on Optimization*, 2000.

[16] P. Nilsson. On the inverse shortest paths problem. In *Proceedings of 17^{th} Nordic Teletraffic Seminar*, Fornebu, Norway, August 2004.

[17] G. Roodman. Post-infeasibility analysis in linear programming. *Management Science*, 25(9):916–922, September 1979.

[18] J. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, July 1971.