Making IGP Routing Robust to Link Failures

Ashwin Sridharan[1] and Roch Guérin[2]

[1] Sprint ATL, 1 Adrian Court, Burlingame CA 94010
ashwin@sprintlabs.com
[2] University of Pennsylvania
guerin@ee.upenn.edu

Abstract. An important requirement of a robust traffic engineering solution is insensitivity to changes, be they in the form of traffic fluctuations or changes in the network topology because of link failures. In this paper we focus on developing a fast and effective technique to compute traffic engineering solutions for Interior Gateway Protocol (IGPs) environments that are robust to link failures in the logical topology. The routing and packet forwarding decisions for IGPs is primarily governed by link weights. Our focus is on computing a *single* set of link weights for a traffic engineering instance that performs well over all *single* logical link failures. Such types of failures, although usually not long lasting, of the order of tens of minutes, can occur with high enough frequency, of the order of several a day, to significantly affect network performance. The relatively short duration of such failures coupled with issues of computational complexity and convergence time due to the size of current day networks discourage adaptive reactions to such events. Consequently, it is desirable to *a priori* compute a routing solution that performs well in all such scenarios. Through computational evaluations we demonstrate that our technique yields link weights that perform well over all *single* link failures and also scales well, in terms of computational complexity, with the size of the network.

Keywords: Routing, Traffic Engineering, IGP, Optimization, link failures.

1 Introduction

With IP networks carrying increasing amounts of commercial traffic, reliability and steady performances have become critical factors. Large ISPs today offer Service Level Agreements (SLAs) covering delay, port availability and losses across their network, which must be maintained even if the network experiences congestion or failures. This provides strong incentives for engineering the network to maintain acceptable levels of performance across a broad range of operating conditions, i.e., fluctuating traffic patterns and intensities or topological changes caused by link and node failures. The focus of this paper is on one such aspect of traffic engineering, namely computation of robust routes to mitigate the impact of single link failures in logical topologies.

Most large service providers map a logical (IP) topology onto an underlying physical fiber network and then operate an Interior Gateway routing protocol like OSPF, IS-IS, or EIGRP on the logical topology. Hence, disruption can occur due to a failure in

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 634–646, 2005.

the physical network, for example a fiber cut, or due to a failure in the logical topology, for example a loss of adjacency between routers due to CPU overload, lost packets, interface failure, etc. Critical failures in the physical network, e.g., fiber cuts, are relatively infrequent but typically long lasting, from hours to days, [8]. As a result, they are dealt with either directly at the physical layer, e.g. automatic switch-over, or with *reactive routing* which re-computes a new routing pattern. The relatively long recovery time of reactive routing is warranted by the scope and severity of such failures. In contrast, logical link failures are much more common but normally affect just a single link (see again [8]), and are typically transient in nature, resolving over a time scale of minutes. Because of their relatively short durations, logical failures are a natural candidate for proactive solutions that rely on *robust routing* which *apriori* anticipate all possible logical link failure scenarios. In particular, relying on reactive solutions may not be appropriate or even feasible, as quickly computing and deploying a complete set of new routes can be challenging, especially in today's large networks that often include several hundred routers. Indeed, by the time the network has determined how to best adapt its routing and propagated the required changes, the failure may have been already fixed or be in the process of being fixed. In addition, re-computing a new optimal routing pattern has the potential for being disruptive to more flows, i.e., re-route them, than just the flows that were originally affected by the failed link.

In this paper, our focus is on developing a technique to compute such *a priori* robust routing solutions for OSPF [9], IS-IS [3] and EIGRP [1], three of the most common intra-domain routing protocols in the Internet. The IGP routing framework imposes two unique constraints. One, traffic in an IGP environment can flow only along shortest paths computed using a set of link weights[1] (say W). Two, load balancing of traffic between a node-pair is allowed only over equal cost multiple paths with the further requirement that traffic be split in *equal* proportions across paths[2]. Within the framework of robust routing, our goal is then to compute a *single* set of link weights for a *given* characterization of traffic entering the network, i.e., traffic matrix T, which performs well under both normal conditions and in the event of any single logical link failures This would ensure that only the routes traversing the failed link are affected and possibly re-routed[3], which is clearly the minimum re-routing required under a link failure. Another important goal is to ensure that the weight computation algorithm scales well with network size which, as mentioned previously, easily span hundreds of routers.

Most of the previous works, on optimizing IGPs for traffic engineering, e.g. [6], [4], [12] and [13] only address the problem in an environment without link failures. As one can expect and as we shall see, performance can degrade significantly if link weights are chosen without accounting for failures. [6] proposed a technique for coping with link failures in OSPF/IS-IS networks, but that involves changing a few link weights upon failure and hence is suitable only for *reactive* routing.

[1] As routing is fully specified by the set of link weights, in this paper we use interchangeably the terms "routing solution" and "weight setting."

[2] Although EIGRP allows unequal splitting, it is rarely used.

[3] Re-routing would not occur if there were multiple shortest paths.

The problem of computing a *single* set of link weights robust to all single link failures was first addressed in [2] and has been studied in [10], [7], and [14]. [10] presents a tabu-search heuristic that computes a set of weights robust to link failures by evaluating the impact of *all* possible link failures for each sampled weight setting. Although it yields good link weights, evaluation of all link failures for a weight setting has very high computation complexity making the search intractable for large networks. In order to avoid this bottleneck a typical method is to evaluate the impact of failure of only a few important links. The key issue then is the identification of such links. [14] relies on random sampling, but this obviously ignores the coupling that exists between links, traffic, and routing. [7] defines important links based on the increase in cost they contribute upon failure. However, the cost increase is computed for the current weight setting, which introduces a strong bias based on the routing that this weight setting induces. In other words, neither the technique of [14] nor that of [7] selects important links in a manner that fully accounts for the complex dependencies that exist between weight settings and traffic flows.

1.1 Our Contributions

We propose a solution to the above problem that is fast, scalable and performs well without the drawbacks of previous techniques. Our contributions are three-fold. One, we introduce a new definition to classify the importance of a link failure that takes into account the traffic matrix, routing *and* the potential benefit to the network from protection against the failure. Two, we develop a new technique to quickly identify critical links based on this definition by extracting relevant information from standard weight search heuristics, like [5] and [4], used to optimize OSPF/IS-IS in no-failure scenarios. *Our technique uses the weight perturbations performed by these heuristics to approximate link failures and capture statistical identifiers, based on which it selects critical links*. Three, our technique possesses a novel feature in that it yields quantifiers for identification of links and/or networks whose failure performance is insensitive to weight settings. We show through computations that compared to [10], our technique yields a computational gain of a factor of over 10 for small size networks and a factor of 100-200 for large real-life sized networks. Equally important, we demonstrate that it yields better results that the techniques of [7] or [14] and is consistently faster.

The rest of the paper is organized as follows. Section 2 provides a general overview of the problem and the challenges involved, including a brief review of related works. Section 3 presents the heuristic we use for computing weight settings. Section 4 evaluates its performance and we conclude in Section 5.

2 Problem Formulation and Related Work

We model the network as a directed graph $G = (V, E)$ with $M = \|V\|$ vertices and $N = \|E\|$ edges. We assume the existence of a traffic matrix T which characterizes the traffic

between each node-pair[4]. For a given set of link weights W, let $\Phi_0(W)$ denote the cost of routing the traffic matrix T over G, i.e., in the no-failure scenario, under IGP constraints. The choice of Φ_0 can be controlled by the network operator[5]. See Section 4 and [11] for details of the function used in this paper. Let $\Phi_l(W)$ denote the cost of routing T over $G-l$, i.e., when link l has failed [6]. Let F be some arbitrary function that weighs the importance of each no-failure and failure scenario. The IGP single link failure robust routing problem may then be stated as

$$\min_W F(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) \quad , W \in \mathbf{Z}^{+N} \tag{1}$$

For the purposes of this paper, we use

$$F(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) = \Phi_0 + \mu \sum_{l \in E} \Phi_l \tag{2}$$

where μ is a parameter that weighs the importance of failures.

The equal-split constraint in standard IP forwarding renders the problem NP-hard [5] even in the (simpler) no-failure scenario and hence motivates the use of heuristics.

In the remainder of the section we review previous related work. The technique presented in [10] proposes a tabu-search heuristic that samples different weight settings and evaluates F over each sample, choosing the minimum. A major drawback of this method is that the computation of F requires evaluating *all* possible link failures for each weight setting. This quickly becomes intractable for large networks.

In order to overcome such computational bottlenecks, a common technique is to hypothesize that the total cost is dominated by a few variables and hence evaluate only their cost. A natural extension of this reasoning is to assume that only a *few* link failures are important and compute weight settings to protect only against their failures. This is equivalent to defining a new cost function

$$F_{E_C}(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) = \Phi_0 + \mu \sum_{l \in E_C} \Phi_l \tag{3}$$

where the new set of links E_C satisfies $\|E_C\| \ll \|E\|$. If the hypothesis is indeed true, minimizing F_{E_C}, Eqn. (3) with a good choice of E_C should yield $F_{E_C}^{best} \approx F^{best}$ as well as significantly reduce computational complexity. However, this raises the question of how to carefully select the subset E_C so that performance is not compromised.

In [14], the links in E_C are determined via random selection. This ignores the relationship between the traffic T and the routing W. In [7] E_C is updated every few iterations by evaluating the cost Φ_l of *all* link failures and choosing the ones that incur the largest cost.

[4] The techniques developed in [6] for optimization over multiple traffic matrices are directly incorporable into our technique. For purposes of clarity and space, however, in this paper we focus on optimization for a *single* traffic matrix.

[5] It is typically a function of link load.

[6] We simply set $w_l = \infty$, re-compute any affected shortest paths and re-route the traffic matrix T.

This technique has two drawbacks. First, in some cases certain links may degrade network performance significantly on failure *regardless* of the weight setting. Including such links in E_C yields little reward in terms of computational effort since network performance would always be poor when these links fail. However, the definition used by [7] would likely include such links in the set E_C since their cost on failure is always large. The second drawback has to do with cases when the impact of a link failure *does* depend on weight settings. Since E_C is updated only every few weight changes, it is quite possible that E_C may not contain the critical links as defined by [7] with regards to the final chosen weight setting W^*. This can also lead to poor performance. We validate our observations in Section 4.

3 Heuristic for Failure Optimization

Based on our arguments in the previous section, we hypothesize that the criticality of a link, which decides its membership in E_C, should be a function of the range **and** variability of cost it incurs on failure over a large, if not the whole set of link weights. We now motivate and define a new metric for the *criticality* of a link that satisfies the above mentioned criteria and present a fast technique for its measurement.

Links which, regardless of weight settings *always* degrade network performance on failure, as well as links whose failure does not affect the network at all are considered *insensitive* to weight settings. Computational effort spent in finding weight settings to protect the network against failure of such links would yield very little reward. Hence such links should typically *not* be included in E_C. On the other hand, link failures that exhibit *large variations in cost across weight settings* would be ideal candidates for the critical link set (E_C). By carefully computing weights, one can protect the network against these link failures. Clearly their inclusion in the critical set yields increased returns in terms of performance improvement per unit of computational effort. These links are considered *sensitive* to weight settings. Hence, we define the *criticality* of a link as a function of some measure that is proportional to its *sensitivity* to weight settings.

In order to quantify the sensitivity of a link, we use a two-threshold based approach. We specify two thresholds to denote satisfactory and unsatisfactory performance. The sensitivity of a link is defined as the number of times, over all weight settings, that network cost on failure of the link falls below or above the satisfactory and unsatisfactory cost thresholds, respectively. The larger and close to each other the two numbers, the higher the likelihood that the link impacts network performance *and* responds to failure optimization. A simple example should illustrate this concept. Consider two links A and B, so that across 100 weight settings failure of link A results in a network cost that is below the satisfactory cost threshold (has little impact) 50 times and above the unsatisfactory cost threshold 50 times (large penalty). Conversely, for link B, the numbers are 90 and 10 respectively. Clearly, A is sensitive to weight settings, while the failure impact of link B is likely to be insensitive to weight settings. Hence inclusion of A in the critical set would be more beneficial.

It now remains to devise a technique that can determine the *sensitivity* of a link efficiently without having to compute the network cost after its failure for all possible

weight settings. Two observations are key to our technique. First, a link failure may be approximated by an *increase* in link weight. Second, weight-search heuristics like [5] devised to compute optimal weights in no-failure scenarios typically operate by making numerous single weight changes to explore the weight space. We can exploit this to achieve our goals. During the course of the exploration of the weight space, we can approximate the impact of failure for each link over a large set of weight settings, simply by capturing appropriate statistical identifiers whenever a link weight is increased. This yields two advantages. One, we can compute the *sensitivity* of a link over a large set of routings *without* incurring additional complexity[7]. Second, it also provides an indicator on network robustness in general (see Section 3.4).

Once the *sensitivity* of each link has been computed, the most sensitive links are included in E_C. The same weight search heuristic is then re-run to compute weights that now minimizes the cost function F_{E_C}, Eqn. (3)

The heuristic can be thought of as consisting of 3 phases:

1. Phase 1: Run a standard weight search heuristic, e.g. [5], to compute the optimal link weights for the no-failure scenario (i.e. minimize Φ_0). During this search, collect statistics whenever link weights are large enough to simulate failure.
2. Phase 2: Compute *sensitivity* of each link and choose the most *sensitive* to comprise the set E_C.
3. Phase 3: Run the standard weight search heuristic to minimize the cost function, F_{E_C}, Eqn. (3).

We note that any heuristic that explores the OSPF/IS-IS weight space by perturbing link weights could be used in Phases 1 and 3. We used the algorithm presented in [5] as our template because the algorithm has been found to be quite fast and effective. In the next sub-sections, we briefly describe each phase. A detailed pseudo-code is available in [11] and omitted here due to space constraints.

3.1 Phase 1: Collecting Link Statistics

The main functions of Phase 1 are to get an estimate of the best cost in the no-failure case, Φ_0^{best} and to gather statistics to compute a measure of criticality for each link.

In each iteration, the underlying no-failure weight search heuristic begins with a weight setting W and then searches the neighbourhood of that space by choosing a link, and perturbing its link weight. Let the weight vector after perturbation be W'. Denote the cost after perturbation as $\Phi_0(W')$. Let the current estimate of the best no-failure cost is Φ_0^{best}. We denote the 2 thresholds mentioned in Section 3 to measure the sensitivity of a link by θ_1(unsatisfactory) and θ_2 (satisfactory).

Each time the perturbation results in the increase of weight of a link l, we collect the following statistics :

1. The number of times link weight *increases*. We denote this by $w_{inc}(l)$.

[7] Note that a weight search to optimize performance in the no-failure scenario needs to be run anyway since our objective contains Φ_0.

2. The number of instances for which the cost of the network lies above the threshold θ_1. We denote this estimate by $\Delta(l)$ and increment it by 1 whenever the current cost $\Phi_0(W')$ satisfies
$$\frac{\Phi_0(W') - \Phi_0^{best}}{\Phi_0^{best}} \geq \theta_1. \tag{4}$$
3. The number of instances for which the cost of the network lies below the threshold θ_2. We denote this estimate by $\gamma(l)$ and increment it by 1 whenever the current cost $\Phi_0(W')$ satisfies
$$\frac{\Phi_0(W') - \Phi_0^{best}}{\Phi_0^{best}} \leq \theta_2. \tag{5}$$

Observe that since $w_{inc}(l)$ is *always* incremented for a weight increase,
$$w_{inc}(l) \geq \Delta(l) + \gamma(l). \tag{6}$$

3.2 Phase 2: Selection of Critical Links

At the end of the execution of Phase 1, we have the estimate of the best no-failure cost Φ_0^{best}, as well as the necessary statistical identifiers to compute the sensitivity of links.

Phase 2 involves the creation of E_C. Towards this end we compute for each link l, the *frequency of cost change* $\alpha(l)$, which is our measure of sensitivity. This is done using the equation
$$\alpha(l) = \min\{(w_{inc}(l) - \Delta(l)), (w_{inc}(l) - \gamma(l))\} \tag{7}$$

The above equation can be explained as follows. If over all sampled weight settings, weight increases of a link result in cost increases that are either always above θ_1 or below θ_2, then the impact of this link's failure is relatively insensitive to weight settings. For such scenarios, if the cost of failure is always high (low), then $\Delta(l)$ (or $\gamma(l)$) would be large. From Eq. (6) and Eq. (7), $\alpha(l)$, the criticality of the link assumes a low value for both cases. On the other hand, if an increase in cost of the link results in widely varying cost increases that are often above or below the corresponding thresholds, then again from Eq. (6) both $\Delta(l)$ and $\gamma(l)$ will be comparatively small, resulting in a large value for $\alpha(l)$. Continuing with the example of Section 3, link A would have a criticality $\alpha(A) = \min(100 - 50, 100 - 50) = 50$, while link B would have a criticality $\alpha(B) = \min(100 - 90, 100 - 10) = 10$.

Construction of the critical link set E_C is now carried out as follows. All candidate links are ordered in decreasing order of $\alpha(l)$. The top L links, which are deemed to be the most critical links, are then chosen to constitute the set E_C of candidate links for failure optimization. $L = \| E_C \|$ is a parameter that can be specified.

3.3 Phase 3: Optimization over Critical Links

The final phase is responsible for finding robust weight settings to protect network performance against failures of links in the critical link set E_C. This is achieved simply by running the same weight search heuristic, but now to optimize the cost function F_{E_C} from Eqn. (3). The search is guided by rejecting all solutions whose no-failure cost,

Φ_0 exceeds the best no-failure cost Φ_0^{best} found in Phase 1 by more than $\theta_2\%$. This coupled with the fact that we are optimizing for failure of a small set of links reduces computational complexity.

3.4 Network Sensitivity to Failure

We now present an additional benefit of our method, namely, the statistical identifiers computed in Phase 1 can be used to estimate the robustness of a network as a *whole*. We define two *network* indicators

$$P = \frac{1}{\| E \|} \sum_{l:\frac{\Delta(l)}{w_{inc}(l)} \geq 0.95} \frac{\Delta(l)}{w_{inc}(l)}, \tag{8}$$

$$G = \frac{1}{\| E \|} \sum_{l:\frac{\gamma(l)}{w_{inc}(l)} \geq 0.95} \frac{\gamma(l)}{w_{inc}(l)}. \tag{9}$$

Intuitively, P (Pathological) provides an estimate of how many link failures can significantly degrade performance regardless of any weight setting, while G (Good) gives an estimate of the fraction of links that have a negligible impact on failure. One would expect G to be close to 1 in instances where almost no link failures cause performance to degrade. Alternatively, one would expect P to be close to 1, if the network is so heavily loaded, or the traffic poorly matched to the network, that almost every link failure causes significant degradation of performance. We shall see the relevance of those indicators in Section 4.

4 Experimental Evaluation

We evaluated the performance of our heuristic through computational experiments on various topologies and traffic matrices. The set of topologies used for evaluation include a PoP level version of the Sprint backbone as well as 5 synthetic networks generated using the Georgia Tech [15] topology generator. For the Sprint backbone, we used an actual traffic matrix generated from traffic measurements, while the traffic matrices for other networks were generated by picking intensities for each node-pair from a uniform distribution. Due to space constraints, we present only a subset of the results. Complete details may be found in [11].

A piecewise linear cost function introduced in [5] is used to represent the cost of routing traffic on each link. It is similar to an $M/M/1$ cost function in that it increases exponentially with link utilization. The network wide cost, Φ, is assumed to be the sum of all link costs. We refer the reader to [11] for further details on the cost function. The thresholds θ_1 and θ_2 were set to 100% and 20% of the best no-failure cost Φ_0^{best}, respectively. The weight search heuristic was run for 1000 iterations in Phase 1 and 100 iterations in Phase 3. As is standard practice in optimization, these parameters may be tuned further if required, although we found them to work well for all topologies considered.

For the smaller networks considered, we compare our algorithm (we shall refer to our heuristic as "Freq") against techniques presented in [10] (referred to as "Exhaus-

tive"), [7] (referred to as "F&T"), random selection of E_C (referred to as "Random") as well as *optimal re-routing*. To obtain a fair comparison, the same weight search heuristic ([5]) used in our technique for exploring the weight space was also utilized in the other heuristics. For optimal re-routing, a new set of weights was computed for *all instances* of link failures on the reduced topology consisting of the original graph minus the failed link. We then compared the performance of our and other heuristics to the performance achievable when re-computing new routes after each failure.

For large networks, both the "Exhaustive" technique and computation of optimal routings for all of the $\| E \|$ possible failures are intractable. In order to circumvent this problem we first identify cases for which route re-computation has the potential to significantly improve performance, that is, those link failures that generate significant cost differences between the no-failure cost and that of our heuristic. For those and only those, we re-compute link weights (i.e recompute the routing) by running the no-failure weight search heuristic ([5]) on the truncated graph from which the failed link has been removed. This enables us to again compare the performance of the different heuristics (except the "Exhaustive") against a benchmark consisting of what is achievable when full re-routing is allowed.

Finally, we note that all the IGP weight search techniques ([5], [4] etc.) are random in nature and our identification of critical links is also based on statistical estimators. Hence it is possible that in a particular run we traverse an atypical sample path and make poor choices when selecting critical links, resulting in poor performance[8]. In order to quantify the frequency of such instances, we performed multiple runs (20) of all heuristics. Consequently, the results are presented in the form of the "fraction of times the heuristic cost lay within a certain fraction of the benchmark cost" over all the runs. The metric used for comparison was always the *worst case* deviation over all link failures between the heuristics and the benchmark costs. For example, a value of 60% under a column titled 30% would imply that in 60% of the runs, the *worst case* deviation of the heuristic from the benchmark over *all* link failures was less than 30%.

4.1 Small Networks

This section presents results for a PoP level version of the Sprint network. Re-computation of the optimal routing after each failure was tractable and hence used as the benchmark. Table 1 shows results for a maximum link utilization of 0.7 under the optimal routing no-failure scenario. From the first row, we observe that the "Exhaustive search" heuristic performs quite well, always within 30% of *optimal re-routing*.

The next two rows, namely $\text{Heur}_{L=40}$ and $\text{Heur}_{L=20}$, show the performance of our heuristic with 40 and 20 critical links, respectively. As can be seen, the results are very good and comparable to those of the "Exhaustive Search" technique. 100% of the sample runs for $L = 40$, and 90% of the samples for $L = 20$, yield a performance within 40% of that obtainable with the best possible re-routing. The "Exhaustive Search" heuristic outperforms our heuristic with $L = 40$ in only 6.6% of the sample runs, and that too by less than 10%.

[8] Given their random nature, *all* the weight search techniques suffer from this problem.

Table 1. Comp. Results for various heuristics on the Sprint n/w, Max Util = 0.7

Heuristic	Freq. of % Deviation from Optimal									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	$\geq 100\%$
Exhaustive	9.1	81.8	9.1	0	0	0	0	0	0	0
Freq. $L = 40$	13.33	73.33	6.67	6.67	0	0	0	0	0	0
Freq. $L = 20$	10	75	0	5	0	0	0	0	0	10
F&T $L = 40$	0	15	10	0	15	15	10	0	0	35
F&T $L = 20$	0	5	0	10	5	10	0	5	0	65
Random $L = 40$	0	25	20	0	20	10	0	0	0	25
Random $L = 20$	0	5	5	0	10	10	15	5	5	45
Plain	0	3.33	0	16.67	3.33	6.67	0	0	0	70

Table 2. Comp. Results for various heuristics on the Sprint n/w, Max Util = 0.5

Heuristic	Frequency of % Deviation from Optimal									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	$> 100\%$
Exhaustive	100	0	0	0	0	0	0	0	0	0
Freq. $L = 20$	100	0	0	0	0	0	0	0	0	0
F&T. $L = 20$	100	0	0	0	0	0	0	0	0	0
Plain	20	80	0	0	0	0	0	0	0	0

Our technique also outperforms the implemented "Random" as well as "F&T" techniques. From Table 1 we observe that the "F&T" technique has a worst-case deviation of more than 100% from optimal re-routing in 35% of the cases with $L = 40$ and in 65% of the cases with $L = 20$. "Random" selection of critical links results in a worst case deviation of more than 100% in 25% of the sample runs with $L = 40$ and in 45% of the sample runs with $L = 20$. This further re-enforces the importance of correctly selecting critical links. Finally, the last row ("Plain"), indicates performance when link weights are optimized only for the no-failure scenario. As expected, it performs the worst with 70% of the runs exceeding optimal performance by 100%.

Table 2 presents an interesting second sample point for the Sprint network, this time for a relatively low utilization of 0.5. As can be seen, all techniques including even the "Plain" heuristic perform exceedingly well. Why is this the case? Our network indicators P and G defined in Section 3.4 correctly explain this situation. The P and G indicators for the various networks at various levels of utilization have been computed in Table 3(a). Note from the 2nd entry, which represents the network in question, that it has a G entry of 0.98. This indicates that at the current utilization, the network should be relatively insensitive to failures, which is indeed the case. Note also that the 1st row which represents the Sprint network with the previous utilization of 0.7, has a P value of 0.046. This indicates the presence of some (4) links that always cause degradation on failure. We confirmed this observation for all 4 links by comparison with the optimal performance on failure of these links.

Our technique is the only one that allows the explicit identification of such instances without resorting to expensive computation. The P and G network indicators readily

Table 3. Computational Time and Insensitivity Indicators for Sprint n/w

Row	Network	Max. Util.	Insensitivity P	Insensitivity G
1.	Sprint	0.7	0.046	0
2.	Sprint	0.5	0	0.98
3.	150 Node	0.7	0	0.97

(a) Insensitivity of various network-traffic matrix instances to link failure

Heuristic	Avg. Time(secs)	Max. Time(secs)
Exhaustive	3320	4207
Freq. $L = 40$	510	626
Freq. $L = 20$	189	224
F&T $L = 40$	608	743
F&T $L = 20$	270	546
Plain	36	49

(b) Comp. Times for various heuristics

identify instances where failure optimization is not beneficial, simply by using information from the computationally inexpensive Phase 1. Based on the values of P and G, one can decide to skip the more expensive Phase 2, and/or initiate a redesign of the network.

Table 3(b), displays statistics of the computation time (on a Dell Intel 1.5 GHz) for each heuristic on the Sprint network. The table clearly demonstrates the significant gain in computation time of our heuristic over the "Exhaustive Search" technique. Our heuristic takes an order of magnitude less time to compute solutions, and in most cases yields equally good weight settings. We also observe that the "Freq." heuristic is faster than the "F&T" technique.

4.2 Large Networks

Scalability to cope with the large size of present day networks has been an important goal of our heuristic. By virtue of the size of large networks, one expects a link failure to have less impact than on small networks. Hence we use smaller critical link sets to improve the running time. Table 4(a) shows results for a 150 node 432 edge graph for "Freq." and "F&T" heuristics with $L = 10$, and for the "Plain" heuristic. Our heuristic as well as the "F&T" heuristic perform well, within 10% of the benchmark over all runs, while the "Plain" heuristic performs well in 80% of its runs. The results are attributable to the high value of $G = 0.97$ for the 150 node network. All but 4 of the links cause little impact on failure. However the 4 links are sensitive to weight settings which offsets the "Plain" heuristics performance in 20% of the runs.

Running times for the network are shown in Table 4(b). Our heuristic took about 4.6 hours and was much faster than the "F&T" heuristic. For the network in question, 10 iterations of the "Exhaustive Search" took about 7 hours. The underlying weight search scheme ([5]) typically takes about 1000 iterations to produce consistent results ([5]). A straightforward extrapolation indicates that the "Exhaustive" technique would require 700 hours to produce results equivalent to our method, i.e., about 2 orders of magnitude more.

Results for several other small and large networks are presented in [11]. They further demonstrate the speed and effectiveness of our technique over other methods as well as the benefits of the network indicators P and G.

Table 4. Performance and complexity statistics for 150 node n/w

Heuristic	Freq. of % Dev. from No-Failure Cost		
	10%	20%	$> 100\%$
Freq. $L = 10$	100	0	0
F&T $L = 10$	100	0	0
Plain	0	80	20

(a) Comp. Results for various heuristics, Max Util = 0.7

Heuristic	Avg. Time(secs)	Max. Time(secs)
Freq. $L = 10$	16489	17847
F&T. $L = 10$	23800	138646
Plain	5559	6387

(b) Comp. Times for various heuristics

5 Conclusions

We have addressed the problem of computing *a single set* of link weights for IGP routing to protect the network against single link failures and perform well under normal conditions. This is an important problem because single link failures are the most common type of failures, and can generate significant disruption if not properly guarded against. The use of "proactive" solutions limits the amount of overhead and re-routing required and also minimizes disruption which are desirable given the transient nature of such failures.

In this work we have presented an efficient and novel heuristic that exploits the information gathered by no-failure search heuristics. We showed how to use this information to construct a set of critical links that are most likely to degrade performance on failure. Exhaustive search to optimize for failure is then performed only for this smaller set of links. Through computational evaluations we have demonstrated that the heuristic scales well with network size, and outperforms previous approaches. Finally, a novel feature of our heuristic is in the form of the P and G indicators that provide early indication of the general sensitivity of the network to failures.

References

1. B. Albrightson, J.J. Garcia-Luna-Aceves, and J. Boyle. EIGRP - a fast routing protocol based on distance vectors. In *Proceedings of Networld/Interop*, May 1994.
2. W. Ben-Ameur. Multi-hour design of survivable classical IP networks. *International Journal of Communication Systens*, 15:553–572, 2002.
3. R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. Request For Comments (Standard) RFC 1195, Internet Engineering Task Force, December 1990.
4. M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. In *Journal of Combinatorial Optimization*, volume 6, 2002.
5. B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of INFOCOM'2000*, Tel Aviv, Israel, March 2000.
6. B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communication*, May 2002.
7. B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *W. Ben-Ameur and A. Petrowski, Editors, Proceedings of INOC*, pages 225–230, October 2003.

8. G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharya, and C. Diot. Analysis of link failures in an IP backbone. In *Proceedings of IMW 2002*, Marseilles, France, November 2002.
9. J. Moy. OSPF Version 2. Request For Comments (Standard) RFC 2328, Internet Engineering Task Force, April 1998.
10. A. Nucci, B. Schroeder, S. Bhattacharya, C. Diot, and N. Taft. IGP link weight assignment for transient link failures. *Proceedings of the 18th ITC*, August 2003.
11. A. Sridharan and R. Guérin. Making OSPF/IS-IS Routing Robust to Link Failures. Technical report, Available at http://einstein.seas.upenn.edu/mnlab/publications.html, University of Pennsylvania, July 2004.
12. A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. *Proceedings of INFOCOM'2003*, April 2003.
13. Z. Wang, Y. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proceedings of INFOCOM'2001*, Anchorage, Alaska, April 2001.
14. Di Yuan. A Bi-Criteria Optimization Approach for Robust OSPF Routing. Technical report, Linkoping University, 2003.
15. E. W. Zegura. GT-ITM: Georgia Tech Internetwork Topology Models (software). http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm, Georgia Tech, 1996.