Graceful Network State Migrations

Saqib Raza, *Member, IEEE,* Yuanbo Zhu, and Chen-Nee Chuah, *Senior Member, IEEE*

***Abstract*—A significant fraction of network events (such as topology or route changes) and the resulting performance degradation stem from premeditated network management and operational tasks. This paper introduces a general class of *Graceful Network State Migration* (GNSM) problems, where the goal is to discover the optimal sequence of operations that progressively transition the network from its *initial* to a desired *final* state while minimizing the overall performance disruption. We investigate two specific GNSM problems: (a) Link Weight Reassignment Scheduling (LWRS) studies the optimal ordering of link weight updates to migrate from an existing to a new link weight assignment, and (b) Link Maintenance Scheduling (LMS) looks at how to schedule link deactivations and subsequent reactivations for maintenance purposes. LWRS and LMS are both combinatorial optimization problems. We use dynamic programming to find the optimal solutions when the problem size is small, and leverage Ants Colony Optimization to get near-optimal solutions for large problem sizes. Our simulation study reveals that judiciously ordering network operations can achieve significant performance gains. Our GNSM solution framework is generic and applies to similar problems with different operational contexts, underlying network protocols or mechanisms, and performance metrics.**

I. Introduction

The Internet has been an enabling technology for mission-critical applications and services such as Voice over IP, VPNs, e-commerce applications, and multimedia streaming. Such applications rely upon consistent Quality of Service (QoS) provisioning by Internet Service Providers (ISPs), with five-nines availability (99.999% uptime) becoming the norm rather than the exception. The end-to-end perceived QoS can potentially be affected due to the dynamic nature of networks. For instance, network topology may change due to transient router/link outages or long-term network engineering. Furthermore, protocol configuration parameters may be altered to migrate from one setting to another. Ideally, QoS guarantees should persist across such dynamic conditions.

Some of these dynamic changes are *inadvertent* e.g., ones due to faulty interfaces, router crashes, and accidental fiber cuts. However, other changes ensue from deliberate and *premeditated* actions of network operators (e.g., routine maintenance). A failure characterization study of an IP backbone [18] observed that planned maintenance activities account for more than 20% of transient failures. Other studies [8] have also observed the prevalence of such planned maintenance activities. Premeditated network tasks also include network upgrade activities such as adding new routers or overhauling link capacity. Another example of a premeditated network task is migrating an existing OSPF [20] or IS-IS [25][1] link weight assignment to a new assignment that has been optimized based on the most up-to-date traffic matrix estimates.

In the case of premeditated tasks, network operators have the prerogative to decide the sequence of *atomic* operations that comprise such a task. This paper[2] introduces a general class of problems referred to as *Graceful Network State Migration* (GNSM) problems, which typically involve migrating a network from its *initial* state to a *final* state by executing a series of *atomic* operations. Each of these operations may cause some *performance disruption* that is a function of the network's changed state. The GNSM problem is to discover the sequence of operations that progressively transition the network to the final state while minimizing the overall disruption. This paper looks at two specific *GNSM* problems, as described below.

A. Link Weight Reassignment Scheduling

Setting link weights is the primary tool used by network operators to control network load distribution and traffic engineering [9–11, 22]. Link weights are optimized based on an estimate of the traffic matrix. They are usually not modified in response to short-term fluctuations in the traffic matrix. However, the estimated traffic matrix may change significantly over a longer period of time, prompting network operators to re-optimize and reset link weights. In such a case, network operators need to migrate from one weight setting to another. The sequence in which the link weights are changed determines the disruption to network traffic during this migration process.

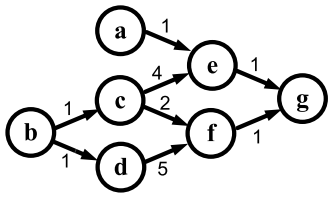


Fig. 1. Example Network

We illustrate this with the help of a toy example. Fig. 1 gives a network with the arc labels representing IGP link weights. Suppose all links have capacity c, and traffic demands between node pairs (a, g) and (b, g) are both $\frac{1}{4}c$. The traffic demand between all other node pairs is 0. The link weights depicted in Fig. 1 are optimal for such a traffic matrix given the objective of minimizing the maximum link utilization (MLU). Now suppose that the traffic demand between node pair (b, g) increases to $\frac{3}{4}c$. Shortest path routing using Equal Cost Multi-Path (ECMP) yields a new optimal weight setting that corresponds to all weights being 1 [10]. This means that three links weights ($w(c, e)$, $w(c, f)$, and $w(d, f)$) have to be

[1]Most common intra-domain (IGP) protocols.

[2]This paper is an extended version of our previous work [28].

	Schedule I		Schedule II	
Step	**Operation**	**MLU**	**Operation**	**MLU**
1	Switch *w(c,e)*	100%	Switch *w(c,f)*	75%
2	Switch *w(d,f)*	62.5%	Switch *w(c,e)*	62.5%
3	Switch *w(c,f)*	50%	Switch *w(d,f)*	50%
	Cost (Max. MLU)	100%	**Cost** (Max. MLU)	75%

TABLE I
LINK WEIGHT REASSIGNMENT SCHEDULING

	Naive Scheduling		Optimal Scheduling	
Step	**Operation**	**MLU**	**Operation**	**MLU**
1	De-activate *(b,c)*	50%	De-activate *(b,c)*	50%
2	Re-activate *(b,c)*	50%	De-activate *(c,f)*	50%
3	De-activate *(c,f)*	100%	Re-activate *(c,f)*	50%
4	Re-activate *(c,f)*	50%	Re-activate *(b,c)*	50%
	Cost (Max. MLU)	100%	**Cost** (Max. MLU)	50%

TABLE II
LINK MAINTENANCE SCHEDULING

changed to 1. There are $3! = 6$ possible orders of changing these link weights (one link at a time). Table I shows two such migration schedules. We see that Schedule I results in a maximum transient MLU of 100% during the weight migration. In comparison, the MLU never exceeds 75% during any stage of the migration process represented by Schedule II. This shows that the sequence in which link weights are changed can be crucial to the extent of disruption to network traffic during the migration process.

A *Link Weight Reassignment Scheduling* (LWRS) problem is characterized by a set of links that we refer to as our *job-set*. All links in the job-set have an *old* weight and a *new* weight. An *atomic* operation involves switching the weight of a link in the job-set from the *old* weight to the *new* weight. The LWRS problem is to find the *minimum-cost* sequence of atomic operations to switch the weights of all links in the job-set. The notion of the cost of a sequence of operations can vary. Our example in Table I used the maximum MLU across all stages of the migration process as a measure of disruption cost.

For a job-set consisting of n links, the solution space for LWRS consists of $n!$ schedules. LWRS is a routine network management task faced by network operators. One might ask why do we stipulate that only a single link weight be changed at a time (as opposed to multiple link weights being simultaneously changed). We understand this to be a common method of operation for LWRS as a result of conversations with network operators. A key reason for this is that certain failure resilience and loop-avoidance mechanisms during IGP convergence work under the constraint that at most one link changes at a time [13, 21]. [14, 17] show how multiple simultaneous changes in the network can result in significant performance degradation.

B. Link Maintenance Scheduling

Network links need to be temporarily taken down for maintenance purposes [18]. Since these link deactivations act as normal failures, they have the same impact on network traffic as normal failures. The difference is that, when more than one link needs to be maintained, network operators can determine the order in which links are deactivated and reactivated. During each step either a link can be deactivated or a deactivated link can be reactivated. It is possible to have multiple links deactivated during an intermediate stage. However, current practice is to fail and restore elements one by one. The intuition behind such a scheme is that the greater the amount of network resources available, the lesser is the disruption to network traffic. This may not always hold as can be seen with the help of a counter example.

Again we refer to the network in Fig. 1. All links have capacity c. We have a traffic demand of $c/2$ each between node pairs (a, g) and (b, g), and zero elsewhere. Suppose we need to take down links (b, c) and (c, f) for maintenance. Table II gives two possible schedules for the required maintenance. We find, somewhat counter-intuitively, that the solution that has both links simultaneously deactivated is better than the solution that fails and restores the links one by one.

A *Link Maintenance Scheduling* (LMS) problem is characterized by a set of links we refer to as our *job-set*. All links are initially active. All links in the job must be deactivated at least once for maintenance, and then must be reactivated. An *atomic* operation involves either deactivating an active link or reactivating a deactivated link. The LMS problem is to find the *minimum-cost* sequence of operations to complete the job. As in the case of LWRS (Table I), we use the maximum MLU across all stages of the migration process as a measure of disruption cost in Table II. Similar to LWRS, the size of the solution space of LMS depends on the number of links in the job-set. Since we have two atomic operations (deactivate and reactivate) for each link in the job-set we can have $(2n)!$ schedules for a job-set with n links. However, we have the added constraint that for each link, deactivation must precede the reactivation. So the actual number of feasible schedules is $(2n)!/2^n$.

The preceding examples for LWRS and LMS are contrived scenarios presented to elucidate the LWRS and LMS problems. The performance gain in a realistic scenario may or may not be as significant. In order to see that, Fig. 2(a) plots the distribution of the disruption cost across all the possible $7! = 5040$ schedules for an LWRS problem, where we migrate the weights of 7 links from one setting to another, in the Abilene [1] network for a given traffic matrix. For the same network and traffic matrix, Fig. 2(b) plots the distribution of the disruption cost across the all possible 2520 schedules for an LMS job consisting of 4 links. In either case we consider the disruption cost to be the maximum MLU experienced during any stage of the migration. We can see from Fig. 2(a) that the difference between the optimal schedule and a randomly chosen schedule can be 20% of link utilization for our LWRS problem. Similarly, Fig. 2(b) shows that the difference between the optimal schedule and a randomly chosen schedule can be more than 15% of link utilization for our LMS problem.

Two things ought to be noted. Firstly, we speculate that the difference between judicious scheduling and rule-of-thumb heuristics stands to be greater for larger job sizes where the solution-space is large. This is because a larger solution space increases the probability that an arbitrary migration schedule

is better than a rule-of-thumb migration schedule. Judicious scheduling allows us to search for and select such better schedules. Fig. 2 shows the distribution of the solution space for job sizes that are small enough for it to be feasible to do so. Our purpose in doing so is to present a preliminary picture of the diversity in solution costs. Results for larger networks and realistic job sizes, and confirmation of our hypothesis, is deferred until Section V. Secondly, our solution has merit even if the average difference between judicious scheduling and rule-of-thumb heuristics is not large. This is because even if rule-of-thumb heuristics work well on average their worst-case performance can be unacceptable. A tractable, well-founded mechanism for planning these operations has merit since it can avoid the pitfalls of using a schedule a human engineer assumes to be 'reasonable' but which turns out to be highly disruptive and inferior in practice.

We summarize our contributions as follows:

- We propose the Graceful Network State Migration (GNSM) problem that seeks to minimize the disruption cost of network operations. Although LMS and LWRS are presented as examples in this paper, the general category of GNSM problems is not restricted to them. Another example of a GNSM problem is to determine the optimal sequence of upgrading a network by adding new routers and links.
- We provide a generic solution framework for GNSM problems, on how to find the optimal scheduling for a series of network operation. We propose a dynamic programming algorithm to find the optimal solution for problem sizes where it is feasible to do so. We propose an Ant Colony Optimization (ACO) based heuristic to find near-optimal solutions for larger problem sizes. Our solution framework is also general and can be applied to any GNSM problem. It can also incorporate different underlying routing protocols as well different metrics of network disruption.
- We show by both example and a detailed simulation study that judicious scheduling of network operations yields significant performance gains over naive scheduling.

The rest of this paper is organized as follows: We discuss related work in Section II. Section III presents our generic GNSM problem formulation and shows how LWRS and LMS can be framed within the GNSM context. Section IV presents two generic solutions for GNSM: a) a Dynamic Programming solution, and b) a solution based on the Ants Colony Optimization meta-heuristic. Section V gives the results of our simulation study for LWRS and LMS, and we conclude and discuss further applications in Section VI.

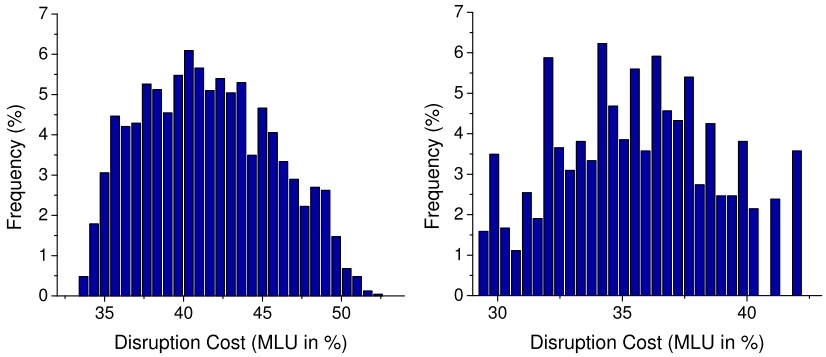
II. Related Work

There exists a rich body of work geared towards avoiding disruption to network traffic in the presence of failures and reconfigurations. In networks employing link-state routing protocols such as OSPF and IS-IS, traffic balancing is achieved by judiciously configuring link weights. [11, 22] present solutions that attempt to optimize weights such that the configured weights remain optimal even if network nodes or links fail. In such networks network failure triggers protocol re-convergence which may result in transient routing loops. Extensions to routing protocols have been proposed to avoid such transient loops [12]. A common theme of such solutions has been ordering forwarding table updates during protocol convergence so that transient loops are circumvented. Proactive approaches to mitigate the effects of failure have also been proposed, especially for Multi-Protocol Label Switching (MPLS) networks [2, 27]. Bandwidth guaranteed backup paths can be pre-configured so that the traffic can immediately be transplanted onto backup routes in event of failure. [17, 32] suppresses failure notification and leverages interface-specific forwarding tables to reroute packets across alternative loop-free paths.

All the above solutions attempt to achieve disruption-free network operation by making routing mechanisms and parameter settings resilient to unexpected failures that we have no control over. However, a significant fraction of failures are actually part of planned maintenance activities. For such premeditated maintenance activities we can control the order and timings of such failures and reconfigurations. Not much attention has been directed towards examining *graceful* network state migrations for premedicated tasks. Certain best-practices exist such as scheduling maintenance activities in the evening when traffic load is low [18]. This increases the operating costs of the network and falls short of optimizing network operations for the existing traffic conditions. Other solutions involve heuristics such as unit increments to a link's weight till no traffic traverses it [5], setting a link weight to the maximum value to gracefully reroute traffic before failure [31], and failing links at most one at a time to minimize disruption.

More recently, Francois et al. proposed an attractive solution that involves progressively iterating through a sequence of link weight changes in order to fail a link or reconfigure its weight [13]. Their solution is specific to avoiding transient routing loops during convergence. [13] is closest in spirit to our work since both leverage the network administrators control over maintenance activity to mitigate network disruption. However, our problem is different. [13] looks at the progressive weight changes for a single link in order to migrate the link from an initial to a final state. We focus on how atomic jobs on a set of links (or nodes) should be ordered so as to minimize overall network disruption. Our work is complimentary to [13], in that once an optimal sequence has been determined for the LWRS problem, their solution can be employed to progressively change the weight for each link. For example, once we determine the sequence in which links are to be deactivated and reactivated, each individual deactivation and reactivation in the sequence can be realized according to [13].

Finally, our state migration problem shares many features with similar problems in artificial intelligence and control theory such as robotic motion planning [16]. The general solution methods that we use, i.e. dynamic programming and swarm intelligence algorithms have also been shown to be applicable to solve such decision-theoretic problems. [15] provides an excellent overview of such problems and proposed solutions.



(a) LWRS problem with 7 links (b) LMS problem with 4 links

Fig. 2. Example distribution of disruption costs for LWRS and LMS problem instances in the Abilene network. Links in the job-set are randomly selected and the disruption cost for these results is defined as the maximum MLU seen during any stage of the migration process.

III. Problem Formulation

A. Graceful Network State Migration (GNSM)

Our definition of LMS and LWRS problem applies to networks employing IGP protocols such as OSPF and IS-IS. We, therefore, restrict our problem formulation to the context of OSPF/IS-IS for ease of exposition. However, the class of problems falling within the rubric of GNSM is much broader. For instance, one can consider a counterpart of the LMS problem for MPLS networks. Our problem formulation can easily be extended to cover such problems.

We consider a network $G(V, E)$, where V is the set of nodes and E is the set of links. We have a traffic matrix $\{\Omega\}_{(i,j)\in V\times V}$ that gives the traffic demand between node-pairs. We define w to be a weight function, $w : E \rightarrow Z^+ \cup \infty$, that assigns a positive integer or ∞ to each link in E. $w(e) = \infty$ implies that link e is down. We also define a *progress* vector $\mathcal{I}$ used to indicate the overall job progress. We use the 4-tuple $(G, \Omega, w, \mathcal{I})$ to represent a network *state*. Let S denote the set of all network states. The network can be in a single unique state at any given point in time. We transition from one network state $s_a = (G, \Omega_a, w_a, \mathcal{I}_a)$ to another $s_b = (G, \Omega_b, w_b, \mathcal{I}_b)$ by altering the weight function from w_a to w_b, where $w_a \neq w_b$. In reality a change of the weight function can result in a change in the traffic matrix as is the case if the BGP egress point for a prefix changes at a router [6]. Our formulation and solution framework can easily account for this. However, for simplicity we assume that a change in the weight function does not affect the traffic matrix, i.e., we assume $\Omega_a = \Omega_b$. We also have a disruption function $d : S \times S \rightarrow \Re$ that gives the disruption cost to transition from one network state to another. We discuss $d(.)$ in detail in Section III-B.

As mentioned in Section I, GNSM basically involves migrating the network from an initial state $s_{\text{initial}} \in S$ to a final state $s_{\text{final}} \in S$. This migration can be realized by executing a series of permissible *atomic network operations*. For every given state $s \in S$, we let the $\mathcal{N}(s)$ denote the set of network states to which we can transition by a single atomic network operation. In other words, the set $\mathcal{A} = \{(s_a, s_b)|s_a \in S \text{ and } s_b \in \mathcal{N}(s_a)\}$ corresponds to the set of all permissible atomic network operations. $\mathcal{N}$ and $\mathcal{A}$ depend upon the specific problem context.

A solution to the GNSM problem is a sequence of state transitions $x = (s_{\text{initial}} = s_0, s_1, s_2, ..., s_{n-1}, s_n = s_{\text{final}})$. Eq. 1-5 give the formal specification of the GNSM problem.

$$\min_{(s_0,s_1,s_2,...s_{n-1},s_n)} \Gamma((s_0, s_1, s_2, ...s_{n-1}, s_n)) \tag{1}$$

subject to:

$$s_0 = s_{\text{initial}} \tag{2}$$

$$s_n = s_{\text{final}} \tag{3}$$

$$(s_i, s_{i+1}) \in \mathcal{A} \qquad \forall 0 \leq i < n \tag{4}$$

$$n \leq B \tag{5}$$

Eq. 2, 3 stipulate that the solution represents a migration from the initial state to the final state. Eq. 4 constrains the transitions to correspond to atomic network operations. Eq. 5 represents the budget constraint that stipulates the maximum number of transitions (B) allowed to complete the migration. Our objective (Eq. 1) is to minimize the overall cost function $\Gamma(x)$, where x is the sequence of state transitions $(s_0, s_1, s_2, ...s_{n_1}, s_n)$. $\Gamma(.)$ is explained in Section III-B.

We now formulate the LMS and LWRS problems within this framework.

1) Link Weight Reassignment Scheduling: In the Link Weight Reassignment Scheduling (LWRS), let $w_{initial}$ represent the old weight setting and let w_{final} represent the new weight setting to which we need to migrate. Therefore, $s_{\text{initial}} = (G, \Omega, w_{initial}, \mathcal{I}_{initial})$ and $s_{\text{final}} = (G, \Omega, w_{final}, \mathcal{I}_{final})$. Let $J \subseteq E$ represent our *job-set* that contains those links j for which $w_{initial}(j) \neq w_{final}(j)$. The progress vector $\mathcal{I}$ is a $|J|$ element vector, with an indicator variable associated with each link in the job-set:

$$\mathcal{I}(e) = \begin{cases} 0 & \text{if } e \text{ has weight set to } w_{initial}(e) \\ 1 & \text{if } e \text{ has weight set to } w_{final}(e) \end{cases} \tag{6}$$

Hence, $\mathcal{I}_{initial}(e) = 0\ \forall e \in J$, and $\mathcal{I}_{final}(e) = 1\ \forall e \in J$. We define an atomic operation as switching the weight of a link in J from $w_{initial}(j)$ to $w_{final}(j)$. Therefore, $(s_a = (G, \Omega_a, w_a, \mathcal{I}_a), s_b = (G, \Omega_b, w_b, \mathcal{I}_b)) \in \mathcal{A}$ implies that:

- $w_a(e) = w_b(e)$ for all links except for a single link $j \in J$, for which $w_b(j) = w_{final}(j)$.
- $\mathcal{I}_a(e) = \mathcal{I}_b(e)$ for all links except for a single link $j \in J$, for which $\mathcal{I}_b(j) = 1$ and $w_b(j) = w_{final}(j)$.

We set $B = |J|$, which is the minimum number of transitions required to migrate from the initial state to the final state.

2) Link Maintenance Scheduling: In the Link Maintenance Scheduling (LMS), let $J \subseteq E$ represent our *job-set*. Let w_0 represent the starting weight function. All links are initially active i.e. $w_0(e) \neq \infty \ \forall e \in E$. The progress vector $\mathcal{I}$ is a $|J|$ element vector, with a variable associated with each link in the job-set.

$$\mathcal{I}(e) = \begin{cases} 0 & \text{if } e \text{ has never been deactivated} \\ 1 & \text{if } e \text{ is deactivated} \\ 2 & \text{if } e \text{ has been reactivated after deactivation} \end{cases} \tag{7}$$

All links in J must be deactivated at least once for maintenance, and then must be reactivated. s_{initial} is given by $(G, \Omega, w_0, \mathcal{I}_{initial})$, where $\mathcal{I}_{initial}(e) = 0 \ \forall e \in J$. Similarly, s_{final} is given by $(G, \Omega, w_0, \mathcal{I}_{final})$, where $\mathcal{I}_{final}(e) = 2 \ \forall e \in J$.

We consider a link activation or deactivation to be an atomic network operation. Therefore, $(s_a = (G, \Omega_a, w_a, \mathcal{I}_a), s_b = (G, \Omega_b, w_b, \mathcal{I}_b)) \in \mathcal{A}$ implies that:

- $w_a(e) = w_b(e)$ for all links except for a single link $j \in J$, for which either $w_a(j) = w_0(j)$ and $w_b(j) = \infty$, or $w_a(j) = \infty$ and $w_b(j) = w_0(j)$.
- $\mathcal{I}_a(e) = \mathcal{I}_b(e)$ for all links other than j, for which:

$$\mathcal{I}(j) = \begin{cases} 1 & \text{if } w_a(j) = w_0(j) \text{ and } w_b(j) = \infty \\ 2 & \text{if } w_a(j) = \infty \text{ and } w_b(j) = w_0(j) \end{cases} \tag{8}$$

The initial and final state both correspond to the starting weight setting w_0. The definition of s_{initial}, s_{final}, and $\mathcal{A}$ ensures that all links in J are deactivated and re-activated at least once. We set $B = 2 \times |J|$, which is the minimum number of transitions required to migrate from the initial state to the final state.

B. Network Disruption Metric

We now detail the overall cost function $\Gamma(x)$, where x is the sequence of state transitions $(s_0, s_1, s_2, ...s_{n_1}, s_n)$. We define $(s_i, s_{i+1}) \in x$ if and only if there exists a transition from s_i to s_{i+1} in x. Section III-A defined the disruption function $d : S \times S \rightarrow \Re$ that gives the disruption cost to transition from one network state to another. $\Gamma(x)$ is basically a function of the individual disruption costs across all transitions in x, i.e., $d(s_i, s_{i+1})$ for all $(s_i, s_{i+1}) \in x$. The disruption function can be defined in a variety of ways. In our context of OSPF/IS-IS, moving from one network state to another can result in transient routing loops and packets losses during the time it takes the routing protocol to converge. Once the routing protocol has converged, the traffic distribution across the network may change. Our problem formulation and solution is independent of the choice of disruption function. For this paper we are primarily interested in looking at measures of link utilization seen on the network after the routing protocol has converged. In other words, the disruption cost $d(s_a, s_b)$ of a transition from $s_a = (G, \Omega_a, w_a, \mathcal{I}_a)$ to $s_b = (G, \Omega_b, w_b, \mathcal{I}_b)$ is a function of the link utilizations resulting from routing Ω_b on G according to w_b. However, we also look at another metric that roughly corresponds to the transient conditions prior to convergence. The metrics used to model the disruption cost ($d(s_a, s_b)$) in this paper are detailed as follows:

- **Maximum Link Utilization (MLU)**: refers to the utilization of the most congested link after the routing protocol has converged according to w_b.
- **Fortz & Thorup Metric (F&T)**: is also a measure of link utilizations after the routing protocol has converged according to w_b. It is the widely used metric proposed in [10] that represents a piece-wise increasing linear convex envelope of a non-linear link cost function.
- **Routing Churn (CHURN)**: is a measure of the volume of traffic that is routed differently between one state and the other. For each OD pair we count the number of links for which one of two things is true: a) the link had traffic for the given OD pair incident on it as per w_a and does not carry traffic for the given OD pair as per w_b, or b) the link did not have traffic for the given OD pair incident on it as per w_a, and carries traffic for the given OD pair as per w_b. CHURN is the sum of such counts over all OD pairs weighted by the traffic volume associated with the OD pair. A high value for this metric roughly corresponds to higher transient disruptions such as routing loops and packet drops before the routing protocol converges.

Our model can easily be extended to account for other and more complex measures of network disruption.

Similarly there exist a number of ways to aggregate the individual disruption costs to compute the overall cost $\Gamma(x)$ for a solution x. To conserve space we restrict ourselves to the *mean* of the individual disruption costs. Hence,

$$\Gamma((s_0, s_1, s_2, ...s_{n-1}, s_n)) = \frac{\sum_{i=0}^{n-1} d(s_i, s_{i+1})}{n} \tag{9}$$

IV. GNSM Solution Framework

A. Dynamic Programming Solution

We define a dynamic programming formulation to compute the optimal sequence for GNSM. We exploit a special property of our problem: the transition cost from one state to another state *only* depends upon the two states (independent of how we arrived at the current state). Hence we can reduce the number of stages by half by breaking up the dynamic programming formulation for our optimal scheduling problem.

Let $P_k(x, z)$ denote the minimum cost of going from state x to state z in k steps. We can, therefore, define the following recurrence relation:

$$P_{2k}(x, z) = \min_{y \in S} \left(P_k(x, y) + P_k(y, z) \right) \tag{10}$$

Eq. 10 implies that the minimum cost of going from state x to state z in $2k$ steps can be obtained by the minimum cost of going from state x to a possible state y in k steps and

then going from state y to state z in k steps. The boundary condition is given by Eq. 11.

$$P_1(x,z) = \begin{cases} \frac{d(x,z)}{B} & \text{if } (x,z) \in \mathcal{A}, \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

The solution of the optimal cost is given by $P_B(s_{initial}, s_{final})$ if B is a multiple of 2. Else, it is given by $\min_{y \in \mathcal{N}(s_{intial})}(d(s_{initial}, y) + P_{B-1}(y, s_{final}))$. In Eq. 11 we divide $d(x,z)$ by B so that, corresponding to Eq. 9, the final cost represents the mean of the individual disruption costs.

Note that the state-space S grows exponentially with the size of the job set $|J|$. The LWRS problem has $2^{|J|}$ states corresponding to the $2^{|J|}$ unique values the progress vector $\mathcal{I}$ can take (Eq. 6). Similarly, the LMS problem has $3^{|J|}$ states corresponding to the $3^{|J|}$ unique values the progress vector $\mathcal{I}$ can take (Eq. 7). Hence the utility of our dynamic programming solution is restricted to those job-sets for which the dynamic programming solution remains tractable. The next section presents an approximation algorithm based on Ants Colony Optimization that can be used as a heuristic for large problem sizes.

B. Ant Colony Optimization based Scheduling

Ant Colony Optimization (ACO) algorithms have been used to produce near-optimal solutions to combinatorial optimization problems, such as the traveling salesman problem [3]. ACO is motivated by the foraging behavior of ants in nature. Ants traveling from nests to food sources deposit a chemical, *pheromone*, along their routes. Ants following them choose routes based upon the deposited pheromone and deposit pheromone themselves along their routes. Shorter pheromone trails get reinforced since the pheromone along less attractive routes evaporates. The ant colony is, therefore, able to converge to the optimal route. A detailed exposition of ACO can be found in [7].

1) Adapting ACO for GNSM: Algorithm 1 represents the ACO meta-heuristic adapted for GNSM. In each iteration of the outer loop n_{ant} ants independently explore a sequence. This process is detailed in Algorithm 2. We let W represent the set of all atomic operations. Hence, for the LWRS problem, since an atomic operation entails switching a link weight, W contains an operation for each link in the job-set. For LMS, since an atomic operation entails either deactivating or reactivating a link, W contains one activate and one deactivate operation for each link in the job-set. We define the set W^T to represent the set of permissible operations that can be performed next. For LWRS, $W^T = W$. However, for LMS, W^T is defined as having all operations in W except for link reactivation for links whose corresponding deactivation operations are also in W. This is because one can only reactivate a link after it has been deactivated.

Algorithm 2 progressively chooses operations to perform until W is empty. Two values $p(c,x)$ and $k(x)$ are used to guide the choice of the next operation x at any stage. $p(c,x)$ represents the amount of pheromone on the route segment (c,x) where c is the last operation performed. $k(x)$ represents

Algorithm 1 AOS

initialize pheromone values
$seq_{global} \Leftarrow$ NULL
for $l = 1$ to l_{max} **do**
  $seq_{local} \Leftarrow$ NULL
  for $i = 1$ to n_{ant} **do**
    $seq \Leftarrow$explore()
    $seq_{local} \Leftarrow \min(seq, seq_{local})$
  end for
  update pheromone values based on seq_{local}
  $seq_{global} \Leftarrow \min(seq_{local}, seq_{global})$
end for

Algorithm 2 explore

$seq \Leftarrow$ NULL
randomly choose c *from* W^T
append c *to* seq *and remove from* W
while $W \neq \emptyset$ **do**
  $q \Leftarrow$ uniform$(0,1)$
  if $q \leq q_0$ **then**
    $n \Leftarrow \max_{x \in W^T}(p(c,x) \times k(x)^{-\beta})$
  else
    $n \Leftarrow n'$ *with probability* $\frac{p(c,n') \times k(n')^{-\beta}}{\sum_{x \in W^T}(p(c,x) \times k(x)^{-\beta})}$
  end if
  append n *to* seq *and remove from* W
  $c \Leftarrow n$
end while
return seq

the network disruption cost to go from the current network state to the next state if we carry out operation x (as defined in Section III-B). Initially, all $p(u,v)$ are initialized to a common value. We assign a score to each operation $x \in W^T$ that measures the attractiveness of choosing x to be our next operation. The score is set to $p(c,x) \times k(x)^{-\beta}$. It should be evident that an operation $x \in W^T$ has a higher score if the pheromone along (c,x) is high and the associated network disruption cost is low. β is a configurable parameter that determines the relative weight to be given to the pheromone value with respect to the network disruption cost. The next operation is chosen to be the operation in W^T with the highest score, with probability q_0. To avoid getting stuck in local optima, the next operation is randomly chosen in proportion to its score, with probability $1 - q_0$.

At the end of each iteration of the inner loop in Algorithm 1, the best sequence is used to update pheromone values according to Eq. 12.

$$p(u,v) = \begin{cases} p(u,v)(1-e_f) + \frac{e_f}{cost(seq_{local})} & (u,v) \in seq_{local} \\ p(u,v)(1-e_f) & (u,v) \notin seq_{local} \end{cases} \tag{12}$$

Here, $cost(seq_{local})$ represents the cost of seq_{local} and e_f represents the evaporation factor that determines the weight to be given to previous pheromone values. seq_{global} represents the best sequence returned by our algorithm.

2) Tuning ACO Parameters: As evident from the preceding description, l_{max}, n_{ant}, q_0, β, and e_f are tunable parameters of

Parameter	Description
l_{max}	The number of iterations of the outer loop in Algorithm 1.
n_{ant}	The number of ants, or the number of iterations of the inner loop in Algorithm 1.
β	Determines the relative importance of pheromone deposition and the disruption cost in choosing the next operation (Algorithm 2: Lines 7 and 9)
q_0	$q0$ is the probability of choosing the next operation with the best pheromone value (Algorithm 2: Line 7); $1 - q0$ is the probability of choosing the next operation in proportion to the respective pheromone values (Algorithm 2: Line 9)
e_f	The evaporation rate of pheromone values (Eq. 12)

TABLE III
ACO PARAMETERS

		n_{ant}	β	e_f	q_0	l_{max}
Abilene	**LWRS**	20	3	0.1	0.7	20
	LMS	20	3	0.1	0.7	20
ISP A & ISP B	**LWRS**	20	2	0.3	0.9	20
	LMS	80	2	0.3	0.7	20

TABLE IV
ACO PARAMETER SETTINGS

our algorithm. Table III contains a brief description of these parameters. We predominantly use three network topologies for our simulation study (see Section V-A2). We tune the ACO parameters separately for each topology. [7] proposes ideal values of the ACO parameters. We select a discrete set of values for each parameter, similar to the ones proposed in [7], and search for the best combination from within these sets. Table IV summarizes the selected parameter setting for each topology used in our simulation experiments.

V. SIMULATION EXPERIMENTS

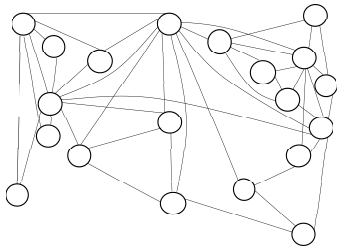
A. Simulation Setup

This section describes a detailed simulation study to evaluate the performance of GNSM scheduling algorithms.

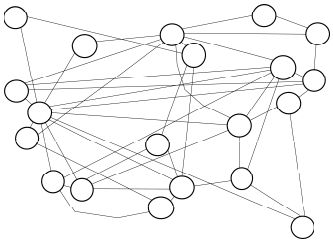
1) GNSM Schemes:

- **Naive Scheduling (NS)**: NS represents the crude heuristic currently employed. In the context of LMS, the NS solution entails failing at most one link at a time. It should be evident from Eq. 9 that, if we fail at most one link at a time, the overall cost is independent of the order in which links are failed. The intuition behind NS is that since the number of simultaneous failures is never greater than one, it would serve to keep the overall network disruption low. In the context of LWRS, NS computes a random permutation of links in the job-set which represents the order in which they are reassigned weights. For LWRS we also define a scheme **NS+** that entails computing 10 such random permutations and selecting the one that yields the minimum network disruption cost.
- **Optimal Scheduling (OS)**: OS computes schedules that yield the optimal network disruption cost using dynamic programming as described in Section IV-A. OS is feasible for small problem sizes and we use it to benchmark the performance of our heuristic algorithm.
- **ACO based Scheduling (AOS)**: AOS uses the ACO meta-heuristic as described in Section IV-B.

2) Simulation Topologies: We essentially use three networks for our simulations. The **Abilene** network has 11 nodes and 28 directional links with 10 Gbps capacity. The other two networks are Tier 1 POP-level topologies **ISP A** and **ISP B**, shown in Fig. 3. The link capacity for each link is set as 1000 units in each direction, modeling the capacity of OC-192 circuits. There are 110 ingress-egress pairs in the Abilene network, and 190 ingress-egress pairs in each of the other two networks. For generality, our problem formulation in Section III presented the job-set J as a set of unidirectional links. However, in practice failure of a link in one direction implies failure of a link in the other direction [18]. Therefore, in this section we simplify the notation and let the definition of J depend on the context. Specifically, J represents a set of bidirectional links if the problem under consideration is LMS, and it represents a set of unidirectional links when we discuss the LWRS problem. Section V-C presents results for some larger topologies as well.



(a) ISP A



(b) ISP B

Fig. 3. Simulation Topologies

3) Synthetic Traffic Matrix Generation: We use three synthetic traffic matrix generation methods employed by [23].

- **Gravity Model (GM)**: GM models the observed characteristics of PoP-to-PoP traffic matrices in Sprints IP backbone [19]. GM specifies three volume categories for traffic demands [4]. The fan-out of traffic originating at a given node is then determined as per the observations in [19].
- **NegativeExponential Model (NegExp)**: NegExp generates traffic matrix entries according to a negative exponential distribution. NegExp is motivated by studies revealing that a fraction of demands in the Sprint IP Network can be explained by the negative exponential distribution [4].
- **LogNormal Model (LogNorm)**: LogNorm generates traffic matrix entries according to a log-normal distribution. [24] finds that LogNorm describes a subset of

traffic demands seen in actual networks. The estimated parameters for the distribution from [24] were $\mu = 16.06$ and $\sigma = 1.04$. Therefore, we use log-normal distributions where $\sigma = \frac{1.04}{16.06}\mu$.

B. GNSM for the Abilene Network

We first conduct a preliminary evaluation using the Abilene network. Our choice of Abilene is influenced by the fact that problem sizes for Abilene are small enough to be solved by OS. We use the GM model to synthetically generate traffic demands for the Abilene network. We set the mean traffic for the GM model such that it yields a maximum link utilization of approximately 33%. We use the well known local-search meta-heuristic proposed in [9, 10] to optimize link weights with respect to a given traffic matrix.

1) Link Weight Reassignment Scheduling (LWRS): We first look at the LWRS problem. An LWRS experiment comprises generating an *initial* and a *perturbed* traffic matrix, that represents how the initial traffic matrix has evolved over time. Let the optimized weight setting for the initial traffic matrix be $w_{initial}$. We introduce random perturbation in the initial traffic matrix by multiplying the demand between each node pair by a fraction uniformly distributed between $1-p_f$ and $1+p_f$, where p_f is the perturbation factor used to model change in traffic demands. For the preliminary evaluation we set $p_f = 0.5$. We optimize the weights for Abilene links with respect to the *perturbed* traffic matrix to get w_{final}. Both $w_{initial}$ and w_{final} are computed using the well known local-search meta-heuristic proposed in [9, 10]. The LWRS problem is to find the optimal schedule to migrate from $w_{initial}$ to w_{final} given that Ω is equal to the perturbed traffic matrix.

Fig. 4 shows the performance of OS, AOS, and NS+. The results are averaged over 100 experiments. We use the cost of NS as a benchmark and report the reduction in overall network disruption cost with respect to NS achieved by the other three schemes. Results are presented for all the three disruption metrics defined in Section III: MLU, F&T and CHURN. For each metric the three GNSM schemes result in a reduction in the overall disruption cost incurred by NS. We report both the average(Fig. 4(a)) and the maximum (Fig. 4(b)) cost reduction over the experiments. The maximum cost reduction is significant since it shows the opportunity cost of not using judicious scheduling in the worst case. Fig. 4 also shows that AOS performs very close to the optimal given by OS.

2) Link Maintenance Scheduling (LMS): We now look at the LMS problem. For each LMS experiment, we have a job size that represents the number of bidirectional links that need to be maintained. We construct the job-set using the **Random Selection (RS)** model. RS randomly selects links to include in the job-set. This is motivated from prior failure characterization studies [18] that observed no correlation between links included in the same maintenance window.

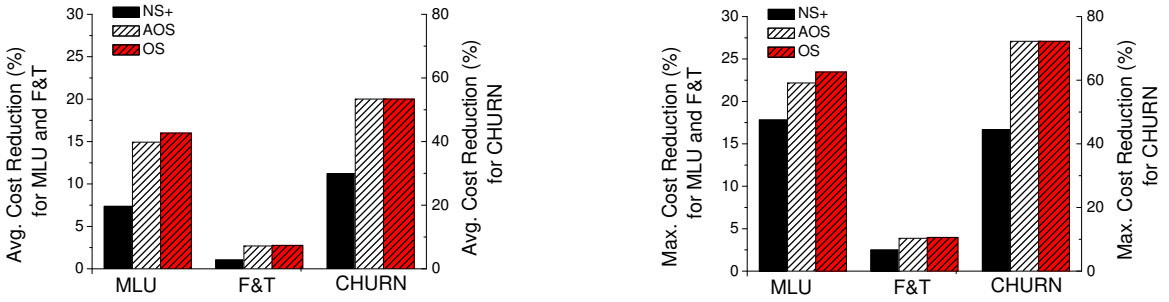
Analogous to Fig. 4, Fig. 5 presents results for all three disruption metrics for LMS in the Abilene network. The results in Fig. 5 use the RS link selection model and the job size is set to 11. As for LWRS, we see that not only OS and AOS result in a performance improvement over NS, but AOS performs very closely to OS. Also, CHURN shows the most significant performance improvement. One notable difference from the results for LWRS is the difference between the average Fig. 5(a) and the maximum Fig. 5(b) cost reduction over NS achieved.

We expect NS to work well for LMS on average, since it makes intuitive sense that if there is no more than one link deactivated at a time, the resulting performance degradation will be low. What we want to evaluate is whether there exist cases where this intuition fails and a different scheduling yields better results.

The performance improvement for F&T is very sensitive to the average traffic load because of the way F&T is defined, i.e., at higher utilization even small differences in utilization result in higher differences in the F&T cost. Since our preliminary experiments consider a traffic load that on average corresponds to 33% of maximum link utilization, difference in the F&T costs are not remarkable. CHURN shows the most significant performance improvement. However, for reasons of conciseness and in line with our primary intention to evaluate GNSM in the the context of traffic engineering disruptions we restrict ourselves to the MLU disruption metric for all subsequent LWRS and LMS results.

Fig. 6(a) shows the percentage of times OS and AOS result in an improvement over NS. We also report the results for **MinHop** routing where instead of optimizing weights as is the case in **TE** routing [9, 10], we assign unit weights to each link. The horizontal axis plots the size of the job-set and the vertical axis gives the percentage of times that a scheme resulted in lower disruption cost than NS. Fig. 6 shows that GNSM scheduling schemes yield better results than NS. Furthermore, in almost all experiments where OS registers an improvement over NS, AOS also does so. The improvement is greater for MinHop routing, which suggests that if the initial weights are suboptimal, the advantage of employing GNSM scheduling is greater. Fig. 6(b) and 6(c) give the average and maximum percentage cost reduction of OS and AOS over NS, across all the experiments where OS results in an improvement over NS. We see that the average improvement is small but the maximum improvement can be as much as 7%. We also see that both the average and the maximum cost reduction increase with increase in the job size i.e., the number of links that are included in the job-set. Section V-C will show how the improvement is more significant for larger networks and job sizes.

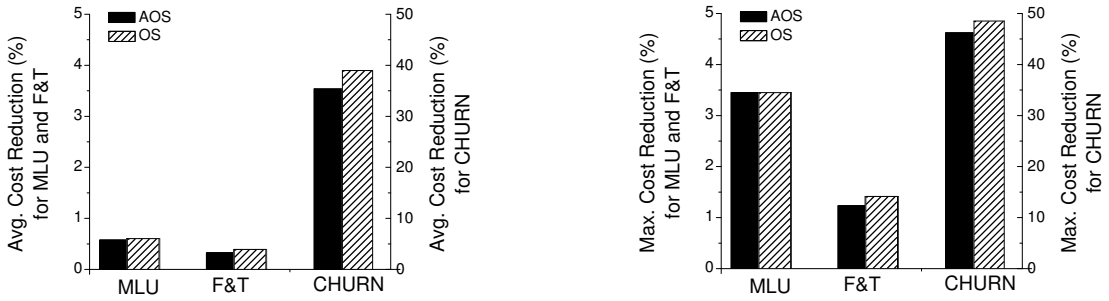
If a GNSM schedule represents an improvement over NS, then there is at least one stage with more than one link deactivated in that schedule. Let κ represent the maximum number of links that are simultaneously down in a schedule. Fig. 7(a) and 7(b) plot the average and maximum κ for OS and AOS, across all the experiments where an improvement over NS is achieved. We expected to see no more than two simultaneous deactivations ($\kappa = 2$) for the Abilene network, but were surprised to see that the a judicious schedule can have as many as four simultaneous deactivations ($\kappa = 4$).



(a) Avg. Cost Reduction across all experiments

(b) Max. Cost Reduction across all experiments

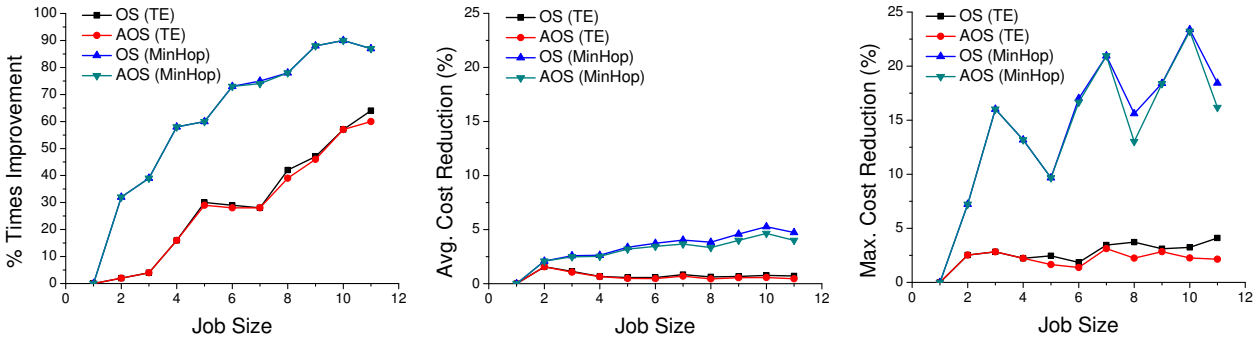
Fig. 4. LWRS: Different Metrics [Network = Abilene; # Experiments = 100; Perturbation Factor (p_f) = 0.5; Traffic Matrix = GM]



(a) Avg. Cost Reduction across all experiments

(b) Max. Cost Reduction across all experiments

Fig. 5. LMS: Different Metrics [Network = Abilene; # Experiments = 100; Job Size = 11; Link Selection = RS; Traffic Matrix = GM]

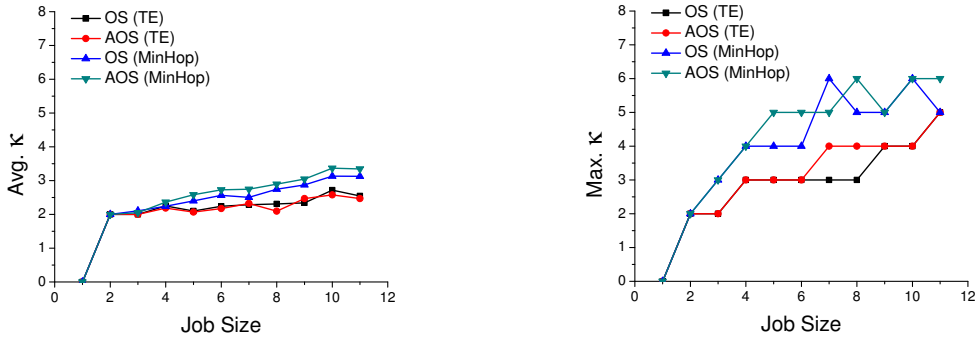


(a) % Times improvement across all experiments

(b) Avg. Cost Reduction across all experiments

(c) Max. Cost Reduction across all experiments

Fig. 6. LMS: Detailed Results [Network = Abilene; # Experiments = 100; Disruption Metric = MLU; Link Selection = RS; Traffic Matrix = GM]



(a) Avg. κ across all experiments

(b) Max. κ across all experiments

Fig. 7. LMS: Simultaneous Deactivations [Network = Abilene; # Experiments = 100; Disruption Metric = MLU; Link Selection = RS; Traffic Matrix = GM]

C. GNSM for Tier 1 Pop Level Topologies

We now evaluate the performance of AOS for our larger networks representing Tier 1 POP level topologies. The results in Section V-B indicated that AOS performs close to OS. All our subsequent experiments do not evaluate OS since it is intractable to do so. We also use MLU as our disruption metric for the remainder of this paper. However, for reference Fig. 8 gives a snapshot of the LMS and LWRS performance for different metrics for ISP A.

1) Link Weight Reassignment Scheduling (LWRS): Fig. 9 gives the reduction in the disruption cost over NS of AOS and NS+ for LWRS in ISP A and ISP B for different values of the perturbation factor p_f. The results are presented as a box-and-whisker diagram summarizing the minimum, lower quartile, median, upper quartile, and maximum cost reduction. The lines depict the average disruption cost. Fig. 9 shows that the improvement is more or less independent of the perturbation factor. Even for low values of p_f (for which w_{final} might be close to $w_{initial}$) we see significant performance gains. We can observe that AOS results in approximately 50% reduction in the disruption cost that would have ensued if we used NS. Also NS+ only yields a cost reduction of around 20%, which is significantly worse than that achieved by AOS. An important observation is that the reductions for ISP A and ISP B are greater than what was observed for the Abilene network. This can be attributed to the larger solution space that exists in the case of the larger topologies. Hence, the difference between a random schedule and one computed through an approximation algorithm can be significant. The results in Fig. 9 use the GM traffic model. Fig. 10 show that our performance gains are similar when different traffic models (Section V-A3) are used.

2) Link Maintenance Scheduling (LMS): Fig. 11 shows the LMS performance of AOS for ISP A and ISP B. As was the case with LWRS, the performance gains are more significant than for the smaller Abilene network. We see that AOS can result in modest reductions in the average disruption cost achieved with NS. We also see AOS performing much better for MinHop routing as compared to TE routing. This is consistent with our previous observations for the Abilene network. We conjecture that if the original weights are not optimal, we witness a greater advantage of employing AOS. We also see greater performance gains for larger job sizes. The modest average cost reduction for TE routing is expected since we expect NS to perform well for the average case. Fig. 12 shows that like LWRS, the performance gain of AOS persists across different distributions of traffic demands. We see that the maximum cost reduction achieved by AOS can be as large as 7% and 40% for TE and MinHop routing, respectively. The maximum cost reduction seen in LMS is more significant since it underscores the opportunity cost of not having a systematic and well-founded mechanism of computing the maintenance schedule.

An important question is whether our results our specific to the way links are chosen. As stated in Section V-B, the RS link selection model is motivated from prior failure characterization studies [18] that observed no correlation between links included in the same maintenance window. However, for completeness we consider a **Topological Correlation (TC)** model that considers such correlations. In TC we start with an empty job-set. In each successive iteration we select a node which has the highest number of links already included in the job-set (and has additional candidate links for addition), and select a bidirectional link incident to the selected node that has not yet been added to the job-set. Fig. 13 shows, as for RS, AOS can deliver significant performance gains for job-sets that correspond to TC.

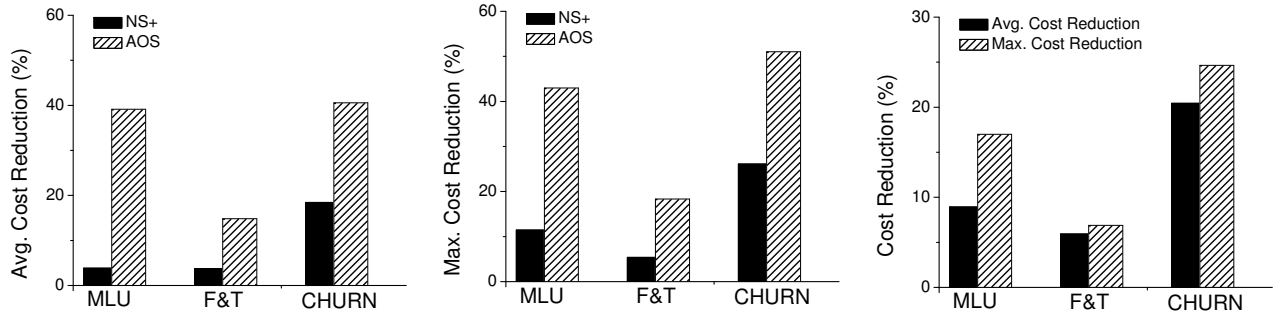
We also evaluate GNSM for a few larger topologies. Specifically, we use topologies drawn from Rocketfuel [30]. The topologies we use correspond to POP-level topologies of AS3967 (79 nodes, 294 links), AS1755 (87 nodes, 322 links), and AS1221 (104 nodes, 302 links). We configure ACO-parameters for these topologies in a manner analogous to Section IV-B2. The job size for LMS for all three Rocketfuel topologies is set to 200 and to 50 for ISP A. We use the RS link selection model for LMS. The perturbation factor p_f used for the LWRS experiment is set to 0.5. Fig. 14 plots the efficiency of the different GNSM schemes for the different network topologies and for ISP A. We see that for both LWRS and LMS, the larger the size of the topology/job-set the greater is the improvement over NS.

D. Hot-Potato Routing

Previous studies [26, 29] on interaction between link configurations and BGP routes show that link events can change BGP routes, thus changing the traffic matrix within the same AS. Such *hot-potato* routing occurs when there are multiple egress points for an external prefix and the IGP distance is used as a tie-breaker. In this scenario, a link deactivation or link weight reassignment may change the IGP distances and cause traffic to shift from the original ingress-egress pair to a new ingress-egress pair.

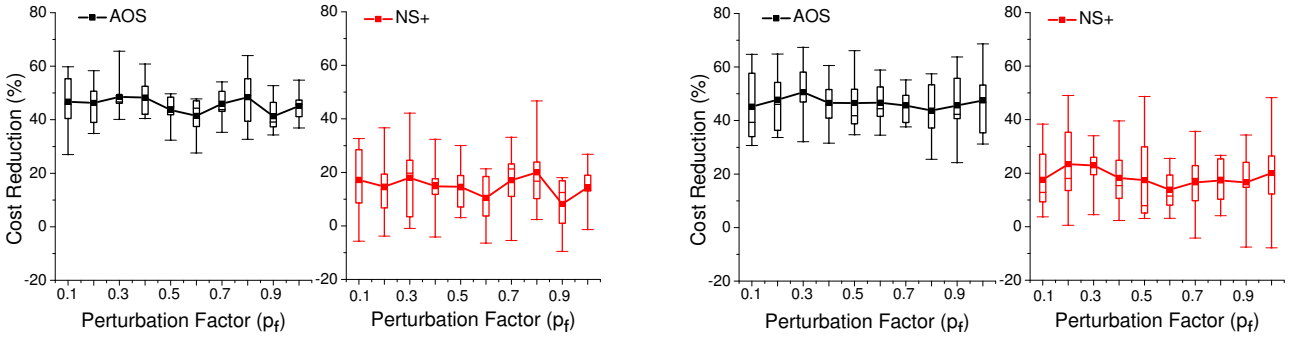
In this section, we stress-test our GNSM framework in the presence of hot-potato routing. We use a fixed traffic demand matrix (TDM) as input and we maintain a dynamic traffic matrix (TM) during the scheduling process. For traffic arriving at an ingress point of an AS and destined for an external prefix, there may exist multiple egress points for leaving the AS. We assume that the traffic demand between an ingress point and its "egress-set" is fixed and we use TDM to denote these demands. On the other hand, a specific egress point is selected from the egress set according to the IGP distance between the ingress and the candidate egress nodes. The TM represents these ingress-to-egress traffic demands for a specific IGP weight setting. When a link is deactivated or its weight changes, we re-select the egress point for each prefix at its ingress point, and get a new ingress to egress traffic matrix (TM). Therefore, during the process of a scheduling, the TDM remains static while the TM can change.

We compute the TDM, from BGP dumps and Netflow records of the Abilene network. The IGP weights are set to the IGP weights corresponding to the period of the Netflow record obtained from the Internet2 Observatory [1]. For LWRS we construct a job-set by randomly selecting links, and randomly perturb the link weights (not necessarily optimizing the



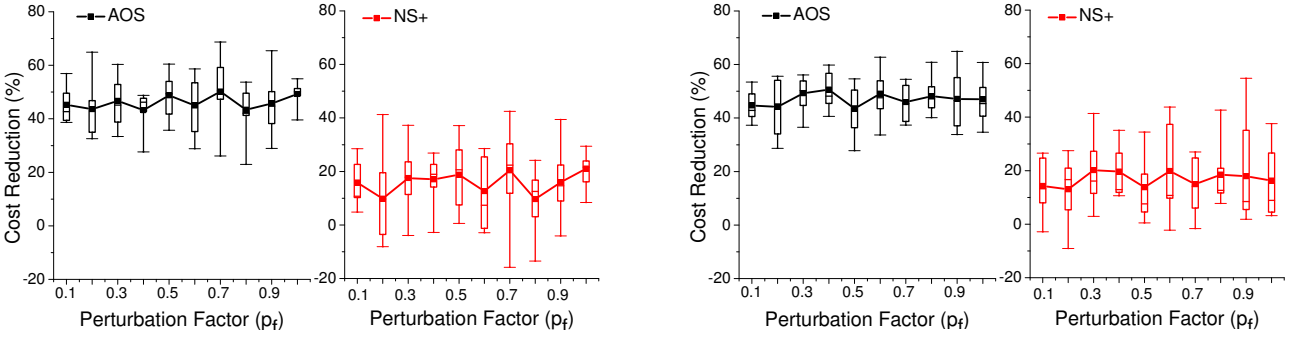
(a) LWRS: Avg. Cost Reduction (b) LWRS: Max. Cost Reduction (c) LMS: Avg. and Max. Cost Reduction

Fig. 8. Different Disruption Metrics [Network=ISP A; # Experiments=100; Traffic Matrix=GM; LWRS $p_f = 0.5$; LMS Job Size=50; LMS Link Selection=RS]



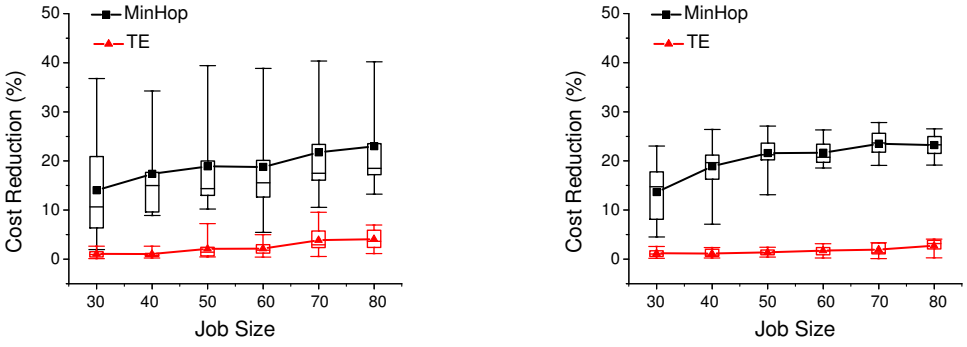
(a) ISP A (b) ISP B

Fig. 9. LWRS: Performance for ISP A & ISP B [# Experiments = 100; Disruption Metric = MLU; Traffic Matrix = GM]



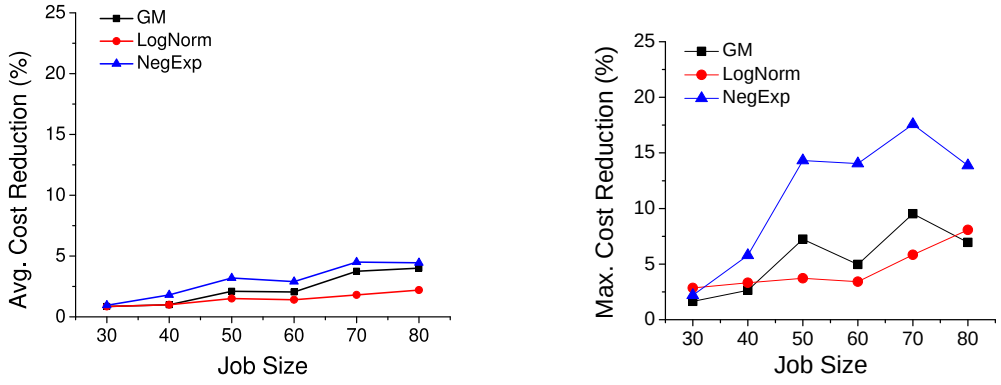
(a) LogNorm (b) NegExp

Fig. 10. LWRS: Performance for Different Traffic Models [Network = ISP A; # Experiments = 100; Disruption Metric = MLU]



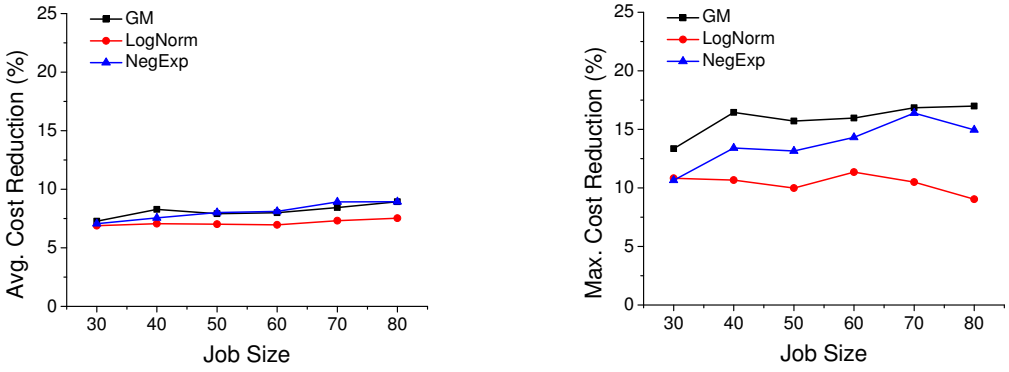
(a) ISP A (b) ISP B

Fig. 11. LMS: Performance for ISP A & ISP B [# Experiments=100; Disruption Metric=MLU; Job Size=50; Link Selection=RS; Traffic Matrix=GM]



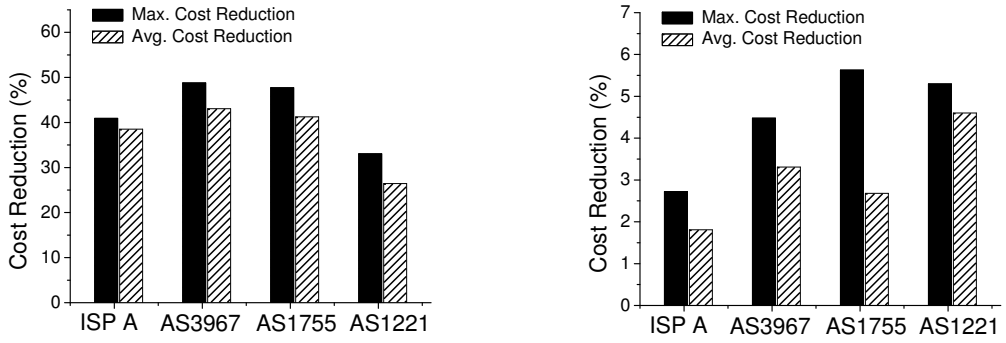
(a) Avg. Cost Reduction (b) Max. Cost Reduction

Fig. 12. LMS: Different Traffic Models [Network = ISP A; # Experiments=100; Disruption Metric=MLU; Job Size=50; Link Selection=RS]



(a) Avg. Cost Reduction (TE) (b) Max. Cost Reduction (TE)

Fig. 13. LMS: Topological Correlation in Job-Set [Network = ISP A; # Experiments=100; Disruption Metric=MLU; Job Size=50]



(a) LWRS (b) LMS

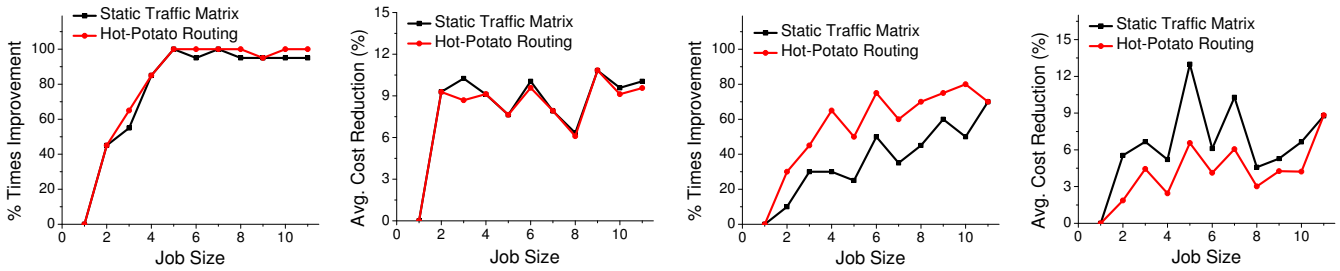
Fig. 14. Rocketfuel Topologies [# Experiments=20; Disruption Metric=MLU; LWRS Perturbation Factor ($p_f = 0.5$); LMS Link Selection=RS; Traffic Matrix=GM]

weights for traffic engineering). We, however, keep perturbing link weights until we arrive at a weight setting that results in a better MLU than the original Abilene weight setting. This represents the final weight setting to migrate to. For the LMS problem we construct the job-set according to the RS model. Fig. 15 shows the performance of GNSM in the presence of hot-potato routing for the LWRS and LMS problems. Fig. 15(a) and Fig. 15(c) show the percentage of experiment runs in which AOS performs better than NS. We can see that with hot-potato routing, AOS shows improvement over NS in more experiments, compared to the case with static traffic matrices.

Similarly, Fig. 15(b) and Fig. 15(d) show the average cost reduction of AOS over NS for the experiments where AOS exceeds NS. The average cost reduction is similar for hot-potato routing compared to the case with static traffic matrices. We, therefore, see that GNSM performances are expected to persist with dynamic traffic matrices.

E. Solution Computation Times

Table V compares the solution computation times of OS and AOS for ISP A. Our algorithms are implemented in Java are run on a PC with a 3 GHz CPU and 3 GB memory, with the maximum Java heap size set to 512 MB. Our



(a) LWRS: Times Improvement (b) LWRS: Avg. Cost Reduction (c) LMS: Times Improvement (d) LMS: Avg. Cost Reduction

Fig. 15. Hot-Potato Routing: Performance for the Abilene network using real traffic traces

Job Size	2	6	10	13	20	22	50	80
LWRS (OS)	0.02	0.1	0.7	5	924	6624		
LWRS (AOS)	0.3	2	5	7	16	20	93	234
LMS (OS)	0.04	0.2	4	144				
LMS (AOS)	0.7	4	13	22	43	58	316	523

TABLE V
SOLUTION COMPUTATION TIMES (SECONDS)

implementation is not optimized for running time, and we report the computation times to give a rough picture of their rate of growth as a function of job size. Table V shows that the average computation time for OS grows much faster than that for AOS, and quickly becomes intractable. We can see that on average it takes 331 times longer to solve an LWRS problem with a job size of 22 using OS as compared to AOS. Similarly the average solution computation time for an LMS problem with a job size of 13 is 6.5 times greater for OS as compared to AOS.

VI. CONCLUSION & FURTHER APPLICATIONS

Our work was motivated by the observation that a significant fraction of network failures, topology changes, and parameter configurations stem from deliberate and premeditated management and operational tasks. Since network operators have the prerogative to decide the exact sequence of such operations, judicious scheduling can minimize network performance disruption. We formulated GNSM as a general class of problems that study how to migrate from an initial to a final network state by executing a series of atomic operations. We presented LWRS and LMS, two problems routinely encountered by network operators, as case studies for GNSM. We also presented an Ants Colony Optimization inspired heuristic for LWRS and LMS. Our simulation study demonstrates that our solution delivers major improvements over current practices for LWRS. For the LMS problem we saw that, although current practice perform well on average, there exist cases where the difference between current practice and our solution can be significant.

We believe that a major strength of this work is the generality of the GNSM problem formulation and solution framework. It is independent of the underlying network and link layer mechanisms, and can incorporate different metrics for network disruption. The following outlines some potential applications of the GNSM framework.

Network Evolution & Upgrade: Enterprise, data-center, and backbone networks continually need to evolve and be upgraded. This involves capacity building by adding new links and nodes and may involve decommissioning older ones. Alternatively, it may involve firmware upgrades on existing nodes It is easy to see how the problem can be formulated within the GNSM framework. The current topology of the network represents the initial state, and the final topology is the desired state after the upgrade activities. An example of an atomic operation is commissioning a single link (changing its weight from ∞ to some positive value). GNSM can chart out a disruption minimizing trajectory of such upgrade operations in order to reach the desired final state.

MPLS Route Re-Optimization: We can leverage the GNSM framework to discover the most efficient rerouting sequence for label switched paths (LSPs) in Multi-Protocol Label Switching (MPLS) networks. Such networks often use dynamic LSP placement, wherein LSP demands are routed one-by-one, with no priori knowledge of future demands. This represents the GNSM initial state. The network bandwidth utilization may become suboptimal after some time as a result of such one-by-one LSP placement. In such cases, network operators can compute a globally optimal routing of LSPs using knowledge of existing LSPs and their bandwidth demands. This represents the GNSM final state. The migration from the initial state to the final state can be realized by reconfiguring the route of each LSP from its initial route to its new route in a make-before-break fashion. This represents an atomic operation. We can use the GNSM solution framework to discover the optimal sequence in which LSP demands should be switched from their existing to their desired routes.

Data Center Power Management: We may also use the GNSM framework to determine the optimal states-of-being at discrete time slots across an optimization interval. This can be accomplished by having placeholder states corresponding to the initial and final states with zero cost transitions out of and into them, respectively. Device power consumption and associated cooling costs are major drivers of costs incurred while operating data centers. Such costs can be curtailed by shutting down computational or networking devices under low load conditions or configuring power saving mechanisms such as dynamic voltage scaling. However, doing so has implications on job latency and other performance metrics. GNSM can be used to determine the optimal trajectory of job-scheduling and power-management decisions to realize an

acceptable latency-energy tradeoff.

The above are only a few examples of how network operators can leverage our framework to deal with a host of problems specific to their operational contexts. We speculate that extending the GNSM framework to deal with a host of similar problems within different operational contexts is a rich area for future work.

REFERENCES

[1] "The Internet2 Network," http://www.internet2.edu, June 2009.

[2] F. Aslam, S. Raza, F. R. Dogar, I. Ahmad, and Z. A. Uzmi, "NPP: A facility based computation framework for restoration routing using aggregate link usage information," in *Proc. International Workshop on QoS in Multiservice IP Networks (QoS-IP)*, Catania, Italy, Feburary 2005.

[3] J. L. Bentley, "Fast algorithms for geometric traveling salesman problems," in *ORSA J. Comput.*, 1992.

[4] S. Bhattacharyya, N. Taft, J. Jetcheva, and C. Diot, "POP-level and access-link-level traffic dynamics in a tier-1 POP," in *Proceedings of 1st ACM Sigcomm Internet Measurement Workshop (IMW)*, Nov. 2001.

[5] S. Bryant and M. Shand, "a framework for loop-free convergence," IETF Draft, December 2006.

[6] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," *Network, IEEE*, vol. 19, no. 6, pp. 5–11, 2005. [Online]. Available: http://dx.doi.org/10.1109/MNET.2005.1541715

[7] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," in *IEEE Trans. Evol. Comput*, April 1997.

[8] N. Dubois, B. Fondeviole, and N. Michel, "Fast convergence project," presented at RIPE47, www.ripe.net/ripe/meetings/ripe-47/presentations/ripe47-routing-fcp.pdf, January 2004.

[9] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[10] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *IEEE INFOCOM*, March 2000.

[11] ——, "Optimizing OSPF/IS-IS weights in a changing world," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 4, pp. 756–767, 2002.

[12] P. Francois and O. Bonaventure, "Avoiding transient loops during IGP convergence in IP networks," 2005. [Online]. Available: citeseer.ist.psu.edu/francois05avoiding.html

[13] P. Franois, M. Shand, and O. Bonaventure, "Disruption-free topology reconfiguration in OSPF networks," in *IEEE INFOCOM*, May 2007.

[14] R. Keralapura, A. Moerschell, C. nee Chuah, G. Iannaccone, and S. Bhattacharyya, "A case for using service availability to characterize ip backbone topologies," *IEEE/KCIS Journal of Communications and Networks*, 2006.

[15] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, also available at http://planning.cs.uiuc.edu/.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[17] S. Lee, S. N. Y. Yu, Z.-L. Zhang, and C. N. Chuah, "Proactive vs. reactive approaches to failure resilient routing," in *IEEE INFOCOM*, March 2004. [Online]. Available: http://www.ece.ucdavis.edu/ chuah/paper/infocom04-fir.pdf

[18] A. Markopoulou, G. Iannaconne, S. Bhattacharrya, C. N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *IEEE INFOCOM*, March 2004. [Online]. Available: http://www.ece.ucdavis.edu/ chuah/paper/infocom04-failures.pdf

[19] A. Medina, N. Taft, S. Battacharya, C. Diot, and K. Salamatian, "Traffic matrix estimation: Existing techniques compared and new directions," in *ACM SIGCOMM*, Aug. 2002. [Online]. Available: citeseer.ist.psu.edu/medina02traffic.html

[20] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[21] S. Nelakuditi, Z. Zhong, J. Wang, R. Keralapura, and C.-N. Chuah, "Mitigating transient loops through interface-specific forwarding," *Comput. Netw.*, vol. 52, no. 3, pp. 593–609, 2008.

[22] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, "IGP Link Weight Assignment for Transient Link Failures," in *ITC*, 2003.

[23] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot, "IGP link weight assignment for operational tier-1 backbones." *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 789–802, 2007.

[24] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating ip traffic matrices: Initial recommendations," in *Computer Communications Review*, July 2005.

[25] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142, Feb. 1990.

[26] T. G. R. Teixeira, A. Shaikh and G. M. Voelker, "Network sensitivity to hot potato disruptions," in *ACM SIGCOMM*, Aug. 2004.

[27] S. Raza, F. Aslam, and Z. A. Uzmi, "Online routing of bandwidth guaranteed paths with local restoration using optimized aggregate usage information," in *Proc. IEEE Int. Conf. on Comm. (ICC)*, Seoul, Korea, May 2005.

[28] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful Network Operations," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

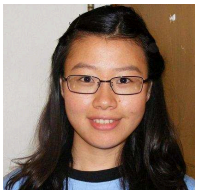
[29] S. B. S. Agarwal, C. N. Chuah and C. Diot, "Impact of bgp dynamics on intra-domain traffic," in *ACM SIGMETRICS*, June 2004.

[30] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," in *ACM SIGCOMM*, 2002, pp. 133–145.

[31] R. Teixeira and J. Rexford, "Managing routing disruptions in internet service provider networks," *IEEE Communications Magazine*, March 2006.

[32] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C. N. Chuah, "Failure Inferencing based Fast Rerouting for Handling Transient Link and Node Failures," *Global Internet Symposium, Miami*, 2005. [Online]. Available: http://www.ece.ucdavis.edu/ chuah/paper/2005/gi05-fifr.pdf

Saqib Raza recieved the BS degree in Computer Science form the Lahore University of Management Sciences in 2004, and the Ph.D and MS degrees in Computer Science from the University of California, Davis in 2007 and 2010, respectively. He is a software engineer at the Data Center Switching Technology Group in Cisco Systems, San Jose. His research interests span network algorithms, traffic engineering, network measurement, optimization and graph theory. He has a specific interest in intra-domain and inter-domain routing algorithms, design and analysis of fault tolerant network systems, and network anomaly detection. He is also interested in characterization of online social network based applications.



Yuanbo Zhu is currently a Ph.D. candidate in the computer Science department at University of California, Davis. She received her BS and ME degree in Computer Science from Tsinghua University, China in 2005 and 2007. Her research interests focus on the area of design and analysis of efficient and reliable network, with emphasize on routing. She is specifically interested in data centers and power related issues.

Chen-Nee Chuah is currently a Professor in the Electrical and Computer Engineering Department at the University of California, Davis. She received her B.S. in Electrical Engineering from Rutgers University, and her M. S. and Ph.D. in Electrical Engineering and Computer Sciences from the University of California, Berkeley. Her research interests lie in the area of computer networks and wireless/mobile computing, with emphasis on Internet measurements, network anomaly detection, network management, multimedia, online social networks, and vehicular ad hoc networks. She received the NSF CAREER Award in 2003, and the Outstanding Junior Faculty Award from the UC Davis College of Engineering in 2004. In 2008, she was selected as a Chancellors Fellow of UC Davis. She has served on the executive/technical program committee of several ACM and IEEE conferences and is currently an Associate Editor for IEEE/ACM Transactions on Networking.