Network Protection with Guaranteed Recovery Times using Recovery Domains

Greg Kuperman
MIT LIDS
Cambridge, MA 02139
gregk@mit.edu

Eytan Modiano
MIT LIDS
Cambridge, MA 02139
modiano@mit.edu

***Abstract*—We consider the problem of providing network protection that guarantees the maximum amount of time that flow can be interrupted after a failure. This is in contrast to schemes that offer no recovery time guarantees, such as IP rerouting, or the prevalent local recovery scheme of Fast ReRoute, which often over-provisions resources to meet recovery time constraints. To meet these recovery time guarantees, we provide a novel and flexible solution by partitioning the network into failure-independent "recovery domains", where within each domain, the maximum amount of time to recover from a failure is guaranteed.**

We show the recovery domain problem to be NP-Hard, and develop an optimal solution in the form of an MILP for both the case when backup capacity can and cannot be shared. This provides protection with guaranteed recovery times using up to 45% less protection resources than local recovery. We demonstrate that the network-wide optimal recovery domain solution can be decomposed into a set of easier to solve subproblems. This allows for the development of flexible and efficient solutions, including an optimal algorithm using Lagrangian relaxation, which simulations show to converge rapidly to an optimal solution. Additionally, an algorithm is developed for when backup sharing is allowed. For dynamic arrivals, this algorithm performs better than the solution that tries to greedily optimize for each incoming demand.

I. Introduction

As the importance of time-sensitive internet traffic continues to rise, there is an increasing need to offer network protection that provides guarantees on the amount of time flow can be disrupted after a failure. Examples of time-sensitive traffic include voice-over-IP and video streaming, which would be rendered unusable with high latency delays. Many networks employ protection techniques that offer no recovery time guarantees whatsoever. Alternatively, networks typically provide local recovery schemes, which reroute a connection at the point of failure; such schemes typically over-provision resources to meet time recovery constraints. In this paper, we a present novel solution that provides guarantees on the maximum amount of time that flow can be disrupted after a failure, which is both flexible and efficient. We refer to these time guarantees as the recovery time of the network.

Protection in the internet has traditionally been accomplished using a real-time rerouting approach: after a link failure occurs, the network is updated with the new set of shortest paths between node pairs, and then a new path is selected. This is both slow (sometimes on the order of minutes) [1], and does not necessarily guarantee that bandwidth will be available for the new path [2, 3]. In the past decade, Multi-Protocol Label Switching (MPLS) has been developed to support constraint based routing, which allows connections to be made with guarantees on parameters such as bandwidth, latency, and recovery time [4]. Because of its flexibility and traffic engineering capabilities, MPLS has become the leading packet transport network technology in backbone networks [5]. To handle these fast recovery times, the Fast ReRoute (FRR) framework was developed to be used within MPLS [6]. FRR is a local recovery scheme where traffic is routed away from the node directly preceding a fault, which is known as the point of local repair (PLR), and reconnects with the original path at the merge point (MP). Various implementations of local recovery have been previously examined [7–11].

More recently, the new IETF standard for the MPLS Transport Profile (MPLS-TP) Protection Framework calls for the creation of "recovery domains" [12]. Recovery domains are defined to be non-overlapping path segments, such that after a failure within a segment, flow is restored using a back-up path between the end-points of that segment. Moreover, recovery domains connect to one another via their respective "reference" end-points, forming an end-to-end protected flow. An example is shown in Fig. 1: after the failure of an edge in the primary path located within Recovery Domain 2, the recovery domain's upstream end-point redirects flow onto the backup path, which then reconnects at that recovery domain's downstream end-point, bypassing the failure. The recovery domain model can be used to provide recovery time guarantees.

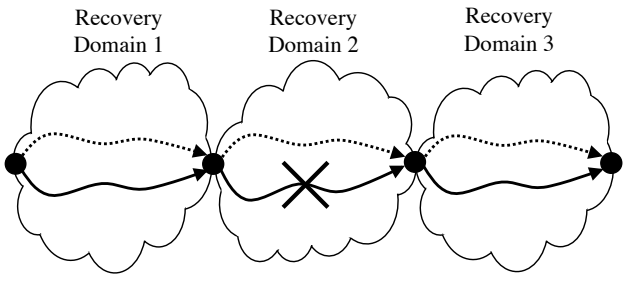


Fig. 1: End-to-end routing using recovery domains

This work was supported by NSF grants CNS-1017800 and CNS-0830961, by DTRA grants HDTRA1-07-1-0004 and HDTRA-09-1-005, and by the Department of the Air Force under Air Force contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

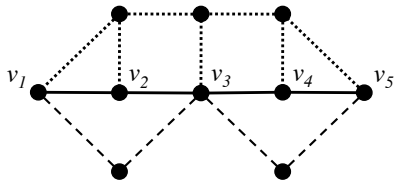


Fig. 2: Time guaranteed recovery examples

In local recovery, since each possible failure has its own dedicated protection path, resources are often over-provisioned beyond what is needed to meet recovery time guarantees. Consider the example shown Fig. 2; the propagation delay for each link is 10 ms, and switching delays are assumed to be negligible. A flow needs to be found from v_1 to v_5 such that the maximum time that the flow can be disripted after a failure is 50 ms, which is the typical recovery time for MPLS networks [1]. A primary path is already allocated on the solid lines from v_1 to v_5. A solution to FRR local recovery is to use all of the links above the primary path: after a link failure in the primary path, a fault notification is sent to the immediate upstream node of that failed link, and the flow is then switched to an alternate path from that node back towards the destination. This protection scheme requires 7 edges to be used for backup.

Now consider an alternative protection routing using the recovery domain model. Two recovery domains are created by using the links below the primary path as the backup paths: one recovery domain between nodes v_1 and v_3, and one between v_3 and v_5. If link $\{v_2, v_3\}$ fails, it would take up to 20 ms for the fault notification to propagate to v_1, and then 20 ms for the data that was switched to the protection route to reconnect with the primary path at node v_3. The other recovery domain will have a similar recovery time after a failure. In this example, only 4 additional links are needed to meet protection guarantees when using recovery domains, as opposed to the 7 needed for FRR.

Previous work has examined providing differentiated reliability for connections, including maximum recovery times. These papers build off of the segment protection framework [13], where segments of a primary path are individually protected, but can overlap by any number of edges. Various heuristics have been considered for segment protection with recovery time considerations [14–16], and an integer linear program (ILP) was presented in [17]. In contrast, recovery domains simplify recovery by separating a flow into disjoint protection regions, where each region guarantees protection for the flow between its end-points. This flow partitioning approach allows the network to be decomposed into a set of individual recovery domains, simplifying capacity allocation and flow control. To the best of our knowledge, the guaranteed recovery time problem using recovery domains has not yet been examined.

Our novel formulation to provide Guaranteed Recovery Times using Recovery Domains (GRT-RD) allows for a general and efficient set of solutions and algorithms. We first present a model of the problem in Section II. We then show in Section III that the recovery domain problem is NP-Hard, and formulate the optimal solution using an MILP. In Section IV, we decompose the end-to-end recovery domain problem into more tractable subproblems, which allows us to more easily construct a solution for the end-to-end problem. This allows for the development of flexible and efficient solutions, including an optimal algorithm using Lagrangian relaxation, which simulations show to converge rapidly to an optimal solution. In Section V, an algorithm is developed for the case when backup sharing is allowed.

II. Model and Problem Description

In this paper, solutions to the problem of Guaranteed Recovery Time using Recovery Domains (GRT-RD) are developed and analyzed. The objective is to provide a primary and protection path for each demand, such that the amount of time flow is disrupted after a failure is guaranteed to be no greater than some maximum value. In order to meet these recovery time guarantees, we separate an end-to-end flow into "recovery domains", where within each recovery domain, the maximum time to recover from a failure cannot exceed a given value. We borrow the definition of recovery domains from [12]: "A recovery domain is defined between two recovery reference end-points which are located at the edges of the recovery domain... To guarantee protection in all situations, a dedicated recovery entity should be pre-provisioned using disjoint resources in the recovery domain, in order to protect against a failure of a working entity."

Adding the constraint for time guaranteed recovery, we implement GRT-RD as follows: an end-to-end recovery domain routing is a set of recovery domains connecting at their respective end-points such that they form a path from the source to the destination, and that the maximum amount of time a flow can be disrupted after any single-link failure is guaranteed. An example was shown in Fig. 1. We implement guaranteed protection within a recovery domain using the 1+1 protection scheme, which provides an edge-disjoint backup path for each primary path, and guarantees the full flow to be available at all times after any single-link failure [18, 19].

The following network model is used for the remainder of the paper. A graph G has a set of vertices V and edges E. We assume a single-link failure model. An end-to-end recovery domain routing will comprise of a set of recovery domains, connected via their reference end-nodes, which form an end-to-end flow from a source to its destination. The maximum recovery time T will be the maximum time data flow can be interrupted: after a link failure of the primary path in some recovery domain, the length of time for the primary flow to be rerouted from the upstream reference end-node to the downstream reference end-node cannot exceed the maximum recovery time of T. This includes the time for fault detection, switching delays, and the time necessary for the flow to reach the end-point of the recovery domain. For ease of exposition, we assume that the link traversal times include all the various switching and detection delays.

Consider the recovery domain in Fig. 3, with link traversal times (in ms) as labeled. After a failure occurs on link $\{j, d\}$, the failure will be seen at j in at most 10 ms, and then it will

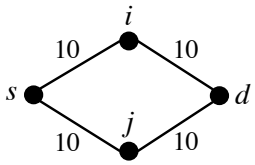


Fig. 3: Recovery domain example

take another 10 ms for that information to reach s. The primary flow will take an additional 20 ms to reach d through i, which gives a total recovery time of 40 ms. Hence, for a recovery domain, the maximum recovery time will be the sum of all the link traversal times within that domain (which includes switching and detection delays).

III. A Minimum-Cost Formulation

This section investigates minimum-cost allocations to find a routing that offers protection after a failure with a guaranteed recovery time by using recovery domains (GRT-RD). Each edge $\{i, j\}$ will have an associated cost c_{ij} and link traversal time t_{ij}. The set of link costs and traversal times will be labeled C and $\mathcal{T}$, respectively. We begin by demonstrating that finding a minimum-cost end-to-end routing using recovery domains with recovery time guarantees is NP-Hard. Subsequently, in Section III-A a mixed integer linear program (MILP) is formulated to find a minimum-cost solution to the end-to-end recovery domain problem with recovery time guarantees. In Section III-B, GRT-RD is compared to the common MPLS local recovery scheme of Fast ReRoute (FRR).

We first show that solving for an individual recovery domain with a guaranteed recovery time is NP-hard. Afterwards, the end-to-end problem is shown to be NP-hard as well.

Theorem 1. *Finding the minimum-cost allocation for a pair of disjoint paths between nodes k and l such that the sum of the link traversals across the two paths does not exceed some maximum time T is NP-hard.*

Proof: We reduce the NP-hard Constrained Shortest Path (CSP) problem [20] to ours. In CSP, each edge $\{i, j\}$ has two associated values: c_{ij} and t_{ij}. In our case c_{ij} is the cost of the edge, and t_{ij} is the link traversal time. If x_{ij} is a binary flow variable for edge $\{i, j\}$, then the objective is to find a path from k to l that minimizes $\sum_{\{i,j\}\in E} c_{ij}x_{ij}$, such that $\sum_{\{i,j\}\in E} t_{ij}x_{ij} \leq T$. To find a minimum-cost solution to CSP by solving GRT-RD with a maximum recovery time of T, add a path between k and l with a total traversal time of zero and a cost of $-(\sum_{\{i,j\}\in E} |c_{ij}| + 1)$. A minimum-cost solution to GRT-RD, if it exists, will always use this path as one of the two disjoint paths, and the other path will be the solution for CSP from k to l with a maximum traversal time of T. In addition, we note that our problem is clearly in NP. ■

Next, we show that finding an end-to-end routing using recovery domains that guarantees the maximum length of flow interruption is NP-hard by using a similar proof to Theorem 1. An end-to-end recovery domain routing can have a series of recovery domains in sequence, each guaranteeing recovery time for its respective flow.

Theorem 2. *Finding the minimum-cost allocation for a protection routing between nodes s and d that guarantees a maximum recovery time of T by using end-to-end recovery domains is NP-hard.*

Proof: We add a path between s and d with a total traversal time of zero and a cost of $-(\sum_{\{i,j\}\in E} |c_{ij}|+1)$. Any minimum-cost solution to the end-to-end GRT-RD will use this negative cost path if a constrained shortest path (CSP) from s to d exists with maximum traversal time of T. Hence, if the minimum-cost solution to the end-to-end problem uses this negative cost path, then the CSP problem has also been solved. If the negative cost path is not used, there exists no solution to CSP on the given network. In addition, we note our problem is clearly in NP. ■

A. MILP to find a Minimum-Cost Solution

Since finding a minimum-cost solution for the guaranteed recovery time problem using recovery domains is NP-hard, in this section a mixed integer linear program (MILP) is developed to solve for the minimum-cost solution. Two versions are developed: one for when backup capacity sharing is not allowed, and one when it is. Backup capacity can be shared between two demands if their primary path edges are disjoint under a single-link failure model. If a failure occurs, and the two primary paths are disjoint, then at most one demand can fail. Hence, at most one demand will need backup protection at a time, and the two demands can share backup protection resources. Due to space constraints, only the MILP for the non-sharing case is presented here. The formulation for when protection sharing is allowed can be found in Appendix A.

Solving for an end-to-end protection routing between nodes s and d using recovery domains relies on the following observations: each pair of nodes in the network is a potential recovery domain, and an end-to-end recovery domain routing will be some subset of these recovery domains. Additionally, in an end-to-end recovery domain routing, recovery domains connect only via their reference end-points, and each recovery domain consists of a pair of edge-disjoint paths. Hence, an end-to-end recovery domain routing is, in fact, a pair of edge-disjoint paths between s and d (potentially connected at certain nodes). Using these observations, the MILP is structured as such: a pair of disjoint paths is found from s to d, where each edge from that pair of disjoint paths is associated with exactly one recovery domain. Additionally, any active recovery domain (i.e., part of the solution) must itself consist of a pair of disjoint paths between the end-nodes of that recovery domain, and the sum of the link traversal times of that recovery domain may not exceed the maximum recovery time. Without loss of generality, a unit demand between s and d is assumed.

The following values are given:

- $G = (V, E)$ is the graph with a set of vertices and edges
- s is the source and d is the destination
- c_{ij} is the cost of link $\{i, j\}$
- t_{ij} is the traversal time for link $\{i, j\}$
- T is the maximum recovery time

The following variables will be solved for:

- x_{ij} is 1 if flow is assigned on link $\{i, j\}$, and 0 otherwise
- R^{kl} is 1 if the recovery domain with end nodes k and l is active (part of the solution), 0 otherwise
- r_{ij}^{kl} is 1 if link $\{i, j\}$ is in recovery domain (k, l), 0 otherwise

The objective is to minimize the total cost of allocating capacity for an end-to-end routing between s and d using recovery domains such that the maximum recovery time of T is not exceeded.

$$\text{minimize} \sum_{\{i,j\}\in E} c_{ij} x_{ij} \tag{1}$$

Subject to the following constraints:

- Find two edge-disjoint paths between the source and destination. Since x_{ij} is strictly 0 or 1, routing two units between s and d will result in two edge-disjoint paths.

$$\sum_{\{i,j\}\in E} x_{ij} - \sum_{\{j,i\}\in E} x_{ji} = \begin{cases} 2 & \text{if } i = s \\ -2 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \ \forall i \in V \tag{2}$$

- If an edge has allocation, it belongs to exactly one recovery domain.

$$\sum_{(k,l)\in(V,V)} r_{ij}^{kl} = x_{ij}, \quad \forall \{i, j\} \in E \tag{3}$$

- Mark active recovery domains
  - If any edge in a recovery domain is active, then that recovery domain is marked as active.

$$\sum_{\{i,j\}\in E} r_{ij}^{kl} \leq |E| \cdot R^{kl}, \quad \forall (k, l) \in (V, V) \tag{4}$$

  - If no edge in a recovery domain is active, then that recovery domain is marked as not active.

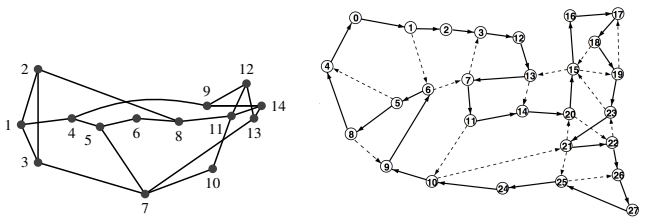
$$\sum_{\{i,j\}\in E} r_{ij}^{kl} \geq R^{kl}, \quad \forall (k, l) \in (V, V) \tag{5}$$

- For each active recovery domain, find two edge-disjoint paths between its respective end-nodes k and l.

$$\sum_{\{i,j\}\in E} r_{ij}^{kl} - \sum_{\{j,i\}\in E} r_{ji}^{kl} = \begin{cases} 2R^{kl} & \text{if } i = k \\ -2R^{kl} & \text{if } i = l \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \ (k, l) \in (V, V) \tag{6}$$

- The sum of the traversed time delays of the edges in a given recovery domain cannot exceed the maximum recovery time.

$$\sum_{\{i,j\}\in E} r_{ij}^{kl} t_{ij} \leq T, \quad (k, l) \in (V, V) \tag{7}$$



(a) NSFNET (b) Lata 'X'

Fig. 4: Network topologies used for simulations

B. Simulation Results for GRT-RD

The minimum-cost solution found by the MILP for the guaranteed recovery time problem using recovery domains is compared to the common MPLS local recovery scheme of Fast ReRoute (FRR). The simulations were run using both the NSFNET and Lata 'X' topologies (Fig. 4) with 100 random unit demands. Each link's traversal time was set to be the propagation delay of that link, plus a 3 ms switching delay. Two versions of the network were tested: one with all unit costs, and one with random integer link costs of uniform distribution between 1 and 5. Both the non-sharing and sharing cases were tested. A dynamic model for routing demands was used: connections are serviced in the order of their arrival (in this case, the 100 demands were randomly ordered), and once a connection is routed, it can no longer be changed. The maximum recovery time was set to the MPLS standard of 50 ms [1]. Fast ReRoute (FRR) is implemented using an MILP, which can be found in Appendix B.

Guaranteed recovery time using recovery domains (GRT-RD) is compared to Fast ReRoute (FRR) for the cases when protection resources can and cannot be shared. The additional cost of spare resources[1] needed to meet protection constraints for the two schemes are compared, with the percent savings of GRT-RD over FRR being shown in Table I.

	No Sharing		Sharing	
Edge Costs	NSFNET	Lata 'X'	NSFNET	Lata 'X'
Random	30%	27%	38%	45%
Deterministic	32%	35%	40%	36%

TABLE I: Percent savings of spare resources of GRT-RD over FRR

As can be seen from Table I, significant savings in the cost of protection resources over the MPLS local recovery scheme of Fast ReRoute are achieved when using GRT-RD. The savings for the case without backup sharing with random edge costs was 30% for NSFNET and 27% for Lata 'X'; with unit edge costs, the savings were 32% and 35% for NSFNET and Lata 'X', respectively. When backup sharing is allowed, the savings were larger: with random edge costs, the savings were 38% for NSFNET and 45% for Lata 'X', and with unit edge cost, 40% and 36%. Further discussion of why backup sharing is more flexible for recovery domain routing (and hence, larger potential savings over other protection schemes) can be found in in Section V. Overall, GRT-RD offers significant savings in cost over FRR, while providing the same level of resiliency.

[1]Spare resources is the capacity needed beyond that of the shortest path routing to meet protection guarantees.

IV. Efficient Algorithms for Guaranteed Recovery Times using Recovery Domains

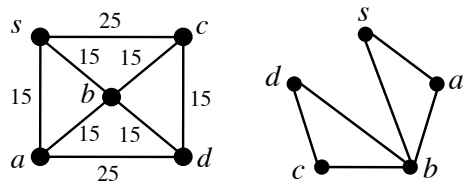
In the previous section, an MILP was presented to find a minimum-cost solution for GRT-RD, which is not generally computationally efficient. In this section, efficient and flexible algorithms are presented to solve GRT-RD. A gradient algorithm is developed that converges rapidly to an optimal solution, and polynomial time heuristics are developed that offer bounds on their solution with respect to the optimal time and cost. In Section IV-A, we demonstrate how finding the optimal solution to the end-to-end recovery domain problem can be solved by decomposing the problem into a set of more tractable and easier to solve individual recovery domain problems. In Section IV-B, a gradient algorithm using Lagrangian relaxation is developed, which simulations show converges rapidly to an optimal solution. In Section IV-C, polynomial timed heuristics that offer bounds with respect to the optimal solution are presented. The different algorithms developed are compared to the optimal solution found by the MILP in Section IV-D. Since it is NP-hard to simply determine if a feasible solution exists for the guaranteed protection problem when backup sharing is used [19], we first consider the case without backup sharing. Recovery domain routing that guarantees recovery times with the use of backup capacity sharing is examined in Section V.

A. Decomposing the End-to-End Recovery Domain Problem

An optimal end-to-end recovery domain routing requires optimizing across the set of possible recovery domains such that a flow from the source to destination is found where the maximum amount of time the flow can be interrupted after a failure is guaranteed. Each individual recovery domain is wholly responsible for protecting against a failure between its respective end-points, and for ensuring that recovery time guarantees are met. This requirement not only allows for easier rerouting after a failure, but, as we demonstrate, allows the end-to-end problem to be simplified by considering only the more tractable individual recovery domains.

The key observation that allows flexibility for finding a solution is that each pair of nodes in the network marks the end-points of a potential recovery domain. Since an end-to-end recovery domain routing will be a series of recovery domains connected via their end-points, the optimal solution will the lowest cost subset of individual recovery domains that form a flow from the source to the destination. We note that there are $O(|V|^2)$ potential recovery domains in a network.

Consider the network G in Fig. 5a, with link traversal times as labeled (in ms), and unit cost links. We wish to find a minimum-cost recovery domain routing from node s to d, with a maximum recovery time of 50 ms. For each pair of nodes in the network, a minimum-cost recovery domain is found. A new network G^R is constructed with the same set of nodes as the original network G, and with an edge being placed between any two nodes i and j if there exists a recovery domain between those nodes that meets recovery time guarantees. The cost of all the edges that comprise the recovery domain between nodes i and j is the cost of edge $\{i, j\}$ in G^R.



(a) Original network G (b) Transformed network G^R

Fig. 5: Decomposing G into its individual recovery domains

The graph transformation G^R is shown in Fig. 5b. An edge between a pair of nodes represents a recovery domain between those same two nodes in G that meets recovery time guarantees. For example, edge $\{s, a\}$ in G^R has a cost of 3, and is a recovery domain consisting of the following edge in G: $\{s, a\}$, $\{s, b\}$, and $\{b, a\}$. It is easy to verify that for this example, each link in G^R has a cost of 3. A shortest path is found in G^R from the source s to the destination d, where each edge of that path represents a recovery domain that is used in the optimal end-to-end recovery domain solution. The shortest path from s to d in G^R is $\{s, b\}$ and $\{b, d\}$, which represents the recovery domains in G between s and b (consisting of the edges $\{s, a\}$, $\{a, b\}$, and $\{s, b\}$ in G), and b and d (consisting of the edges $\{b, c\}$, $\{c, d\}$, and $\{b, d\}$ in G).

This algorithm is labeled `end_to_end_RD`. We note that this approach does not preclude two recovery domains from sharing edges between them, but the essence of recovery domain routing is preserved: the end-nodes of a recovery domain maintain full responsibility for protecting against a failure for the primary flow contained within it, and for guaranteeing that the amount of time that flow can be interrupted does not exceed the maximum allowed. This allows us to now focus on solving the individual recovery domain problem, i.e., for a given a pair of nodes, finding the shortest-pair of disjoint paths between those nodes that meet time constraints. With such an algorithm at hand, one can use the above approach to solve for the end-to-end recovery domain routing with guaranteed recovery times.

B. Optimal Algorithm

In this section, an optimal algorithm using a Lagrangian relaxation is presented. The individual recovery domain problem is first formulated as an MILP, where we wish to find a minimum-cost pair of disjoint paths between a source s and destination d that have some maximum total link traversal time. The MILP is formulated as such: the objective is to find a routing of minimum cost, such that two units of flow are routed between the end-nodes s and d, with flow variables x_{ij} being binary, and the total link traversal time not exceeding the maximum recovery time of T. Since an edge will have strictly a flow of 0 or 1, two paths between s and d cannot overlap; hence, a minimum-cost pair of disjoint paths will be found subject to the time constraints. We refer to this as the constrained shortest-pair of disjoint paths problem.

$$\text{minimize} \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{8}$$

$$\sum_{\{i,j\}\in E} x_{ij} \; - \sum_{\{j,i\}\in E} x_{ji} = \begin{cases} 2 & \text{if } i = s \\ -2 & \text{if } i = d \;,\; \forall i \in V \\ 0 & \text{o.w.} \end{cases} \tag{9}$$

$$\sum_{\{i,j\}\in E} t_{ij} x_{ij} \leq T \tag{10}$$

$$x_{ij} \in \{0,1\}, \quad \forall \{i,j\} \in E \tag{11}$$

The Lagrangian dual is obtained by relaxing the maximum recovery time constraint (Constraint 10), and placing it into the objective:

$$\begin{aligned} L(\mu) &= \min \sum_{\{i,j\}\in E} c_{ij} x_{ij} + \mu \Big(\sum_{\{i,,j\}\in E} t_{ij} x_{ij} - T \Big) \\ &= \min \sum_{\{i,j\}\in E} x_{ij}(c_{ij} + \mu t_{ij}) - \mu T \end{aligned} \tag{12}$$

For a fixed value of μ, the cost of edge $\{i,j\}$ becomes $c_{ij} + \mu t_{ij}$. The recovery domain problem now becomes a *minimum-cost flow* problem, which is defined as finding a flow of lowest cost between a source and destination in a network that has both edge costs and edge capacities [20]. An important characteristic of minimum-cost flows is when given strictly integer inputs (edge costs and capacities), then the solution will always be a set of integer flows [20]; the flow variables no longer need to be constrained to integer values in order to ensure that flows are strictly integer. Hence, by relaxing the binary flow variables x_{ij}, we can write the Lagrangian dual as a linear program, which becomes polynomial time solvable [21].

$$\sum_{\{i,j\}\in E} x_{ij} \; - \sum_{\{j,i\}\in E} x_{ji} = \begin{cases} 2 & \text{if } s = i \\ -2 & \text{if } t = i \;, \\ 0 & \text{o.w.} \end{cases} \quad \forall i \in V$$

$$0 \leq x_{ij} \leq 1, \quad \forall \{i,j\} \in E$$

For a fixed value of μ, the Lagrangian relaxation simply becomes finding the shortest-pair of disjoint paths with respect to the edge costs $c_{ij} + \mu t_{ij}$, $\forall \{i,j\} \in E$. The shortest-pair of disjoint paths can be found in polynomial time using Suurballe's algorithm [22].

To solve the Langrangian relaxation, we wish to find $L^* = L(\mu^*) = \max_{\mu \geq 0} L(\mu)$. The constrained shortest-pair of disjoint paths has similarities to the constrained shortest path (CSP) problem, for which Lagrangian relaxation techniques have been considered [23–26]. In [24], a geometric approach is proposed for solving the Lagrangian relaxation for CSP. While not considered beyond a single constrained path, we demonstrate that the geometric approach can be extended to the constrained shortest-pair disjoint pair of paths problem. $L(\mu)$ can be rewritten as $L(\mu, \mathbf{x}) = f(\mathbf{x}) + \mu g(\mathbf{x})$, where $f(\mathbf{x}) = \sum_{\{i,j\}\in E} c_{ij} x_{ij}$ and $g(\mathbf{x}) = \sum_{\{i,j\}\in E} t_{ij} x_{ij} - T$. Each path becomes a line in $L - \mu$ geometric space, where $g(\mathbf{x})$ is the slope of the line, and $f(\mathbf{x})$ is the L axis crossing point. Furthermore, since $g(\mathbf{x}) = \sum_{\{i,j\}\in E} t_{ij} x_{ij} - T$, the paths associated with $g(\mathbf{x}) \leq 0$ meets time requirements, where $f(\mathbf{x})$ is the cost of those disjoint paths.

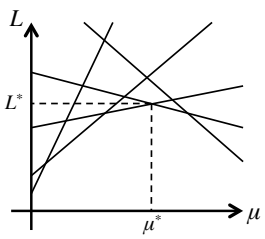


Fig. 6: Pair of disjoint paths mapped to lines in $L - \mu$ space

We see that the mapping to a geometric space is in fact not specific to CSP; any discrete optimization problem that takes the form $L(\mu, \mathbf{x}) = f(\mathbf{x}) + \mu g(\mathbf{x})$ can be represented as a line in $L - \mu$ space. In our case, each pair of disjoint paths becomes a line in $L - \mu$ space, and the lower envelope of lines gives the optimal value for μ^*. A visualization is shown in Fig. 6 [26].

Since any discrete optimization problem taking the form of $L(\mu, \mathbf{x}) = f(\mathbf{x}) + \mu g(\mathbf{x})$ can be represented in geometric space, including the constrained shortest-pair of disjoint paths, a similar method for finding the optimal μ^* as in [24] can be applied. The key is being able to solve for the optimal discrete variable assignments $\mathbf{x}$ for a fixed μ in polynomial time. In the case of constrained shortest-pair of disjoint paths, Suurballe's algorithm [22] can be used, as previously discussed. The algorithm to find the dual optimal solution is labeled `rd_dual_opt`, and is presented in Algorithm 1.

Algorithm 1 $\mu^* = \texttt{rd\_dual\_opt}(s, d, T, V, E, C, \mathcal{T})$

1: Begin with two pairs of disjoint paths from s to d: $L(0)$ is the shortest pair without consideration to time, and $L(\infty)$ is the shortest time pair, without consideration to cost.
2: The intersection point is the first guess for μ (call this μ'). $L(\mu')$ is the new upper bound on the dual-optimal solution. Any intersection point will be defined by two lines: one with $g(\mathbf{x}_1) \leq 0$ and the other $g(\mathbf{x}_2) > 0$. Since $g(\mathbf{x}) = \sum_{\{i,j\}\in E} t_{ij} x_{ij} - T$, the disjoint paths associated with $g(\mathbf{x}_1) \leq 0$ meets the time requirements.
3: Solve for the shortest-pair of disjoints paths with edge costs $c_{ij} + \mu' t_{ij}, \forall \{i,j\} \in E$. This will give a new pair of disjoint paths $\mathbf{x}''$, with some value for $f(\mathbf{x}'')$ and $g(\mathbf{x}'')$.
4: If $f(\mathbf{x}'') + \mu' g(\mathbf{x}'') = L(\mu')$, then the optimal value for μ has been found.
5: **while** $f(\mathbf{x}'') + \mu' g(\mathbf{x}'') \neq L(\mu')$ **do**
6: Add a new line $f(\mathbf{x}'') + \mu' g(\mathbf{x}'')$ in $L - \mu$ space.
7: The next estimate for μ' is the new intersection point on the lower envelope of the lines that gives the max $L(\mu')$.
8: If $f(\mathbf{x}'') + \mu' g(\mathbf{x}'') = L(\mu')$, then the optimal value for μ has been found.
9: **end while**
10: Return $\mu^* = \mu'$. The pair of dual optimal paths $\mathbf{x}^*$ at μ^* that meet time constraints are those associated with $g(\mathbf{x}^*) \leq 0$. The algorithm concludes with an upper and lower bound on solution. Since $f(\mathbf{x}^*)$ represents the cost for feasible paths in the network that meet time constraints, it is an upper bound on the optimal solution. The lower bound is the dual-optimal value $L(\mu^*) = f(\mathbf{x}^*) + \mu^* g(\mathbf{x}^*)$.

The runtime for `rd_dual_opt` is not specified in [24], but a binary search approach for solving the same problem was presented in [26] with a polynomial runtime.

Since the original problem we are trying solve has integer constraints (the flow variables), the dual optimal solution may have a gap in cost between itself and the actual optimal solution, which is known as a duality gap [21]. To close this gap, we iterate through the k-shortest pair of disjoint paths with respect to the dual-optimal edge costs $c_{ij} + \mu^* t_{ij}, \forall \{i,j\} \in E$, closing the gap with each iteration until the optimal solution is found. The authors of [24] step through the k-shortest paths, using an algorithm [27] that cannot be extended to the k-shortest pair of disjoint paths. Instead, we use an alternate technique proposed in [28], which finds the k best solutions for a discrete optimization problem. The run time to find the k^{th} best discrete optimization solution is polynomial with respect to the time to solve the discrete optimization problem, which for the case of shortest-pair of disjoint paths is also polynomial [22]. Details of the algorithm are omitted for brevity, and can be found in [28]. We label the final algorithm, which closes the duality gap, `rd_opt`; it is presented in Algorithm 2.

Algorithm 2 $(P_1, P_2) =$ `rd_opt`$(s, d, T, V, E, C, \mathcal{T})$

1: Find the initial dual optimal solution:
$\mu^* =$ `rd_dual_opt`$(s, d, T, V, E, C, \mathcal{T})$
2: Find the k^{th} shortest pair of disjoint paths $\mathbf{x}^k$ for each $k = 1, 2, ...$ with respect to the dual-optimal edge costs $c_{ij} + \mu^* t_{ij}$.
- For each k, $L(\mu^*, \mathbf{x}^k)$ is the new lower bound on the optimal feasible solution, and is a non-decreasing function with respect to k.
- An upper bound U^k is maintained: $U^k = \min_{1..k} f(\mathbf{x}^k)$ for all $\mathbf{x}^k$ such that $g(\mathbf{x}^k) \leq 0$ (which means that $\mathbf{x}^k$ is a solution that meets time constraints). U^k is a non-increasing upper bound.

3: Continue until $U^k \leq L(\mu^*, \mathbf{x}^k)$, at which point the optimal solution has been found.
- Since we individually step through the shortest-pair of dual-optimal disjoint paths, which are a lower bound on the solution, until their cost is greater than the upper bound, which is a feasible solution, the solution must be optimal. Proof is shown in [24].

4: Return the pair of disjoint paths P_1 and P_2 associated with the upper bound U^k, which meets time constraints.

Since the number of possible disjoint paths can be potentially exponential with respect to the number of edges, the number of iterations to close the duality gap is not necessarily polynomial bounded. But, our simulations indicate that the number of iterations to close the duality gap is in fact minimal: on average, only 1.46 iterations are needed.

C. Polynomial Timed Heuristics

In this section, two algorithms are presented that run in polynomial time to solve the guaranteed recovery time problem using recovery domains. The first is a "fastest paths" algorithm, where link costs are ignored, and paths are found with respect to time only. This algorithm is the simplest to implement, and is most useful when link costs are either not considered, or are proportional to the link traversal times. The second is a fully polynomial time approximation scheme (FPTAS) that guarantees a solution to be within a factor of 1.5 of the optimal cost and $1.5(1+\epsilon)$ of the maximum recovery time.

The fastest paths algorithm is straightforward to implement using the shortest-pair of disjoint paths algorithm [22]. Instead of finding a pair of disjoint paths of minimum-cost, a pair of disjoint paths of minimum time are found. This algorithm is called `rd_fastest`, and has the same complexity as finding the shortest-pair of disjoint paths.

For the approximation scheme, we use an algorithm presented in [29]. In that work, the authors try to solve for disjoint QoS (quality-of-service) paths, such that each path is bounded by some time requirement D. They are not looking at recovery times or recovery domains, but are instead interested in ensuring that the end-to-end primary and backup paths do not exceed some QoS specification. To solve their problem, they relax each path's individual time constraint and try to find a pair of disjoint paths such that the sum of the link traversal times for both paths does not exceed $2D$. By replacing the time requirement $2D$ with the maximum recovery time requirement T, we can solve for a recovery domain with a bound on the optimal cost and time. We label this approximation algorithm `rd_approx`; details of the algorithm are omitted for brevity, and can be found in [29].

D. Algorithm Simulations

In this section, the algorithms developed in the previous sections for guaranteed recovery times using recovery domains are compared to the end-to-end optimal solution found by the MILP from Section III. A similar simulation to Section III-B was run, using the Lata 'X' topology (Fig. 4b). Table II shows the percent that each of the algorithms differed in total cost of resources used over the minimum-cost solution found by the MILP.

Protection Scheme	Unit Edge Costs	Random Edge Costs
`rd_opt`	0	0
`rd_fastest`	1%	6%
`rd_approx`	5%	2%

TABLE II: Difference for the algorithms from optimal

As expected, the optimal algorithm `rd_opt` did not differ from the optimal solution found by the MILP. Additionally, the average number of iterations needed to close the duality gap was 1.46 over all simulated demands. Both the fastest paths and approximation algorithm also performed close to optimal. When edge costs were uniform, the fastest paths approach in fact gave slightly better results. But when edge costs were changed to be random, the approximation algorithm performed better since it tries to optimize with respect to cost, and the fastest paths algorithm does not.

V. Algorithm with Backup Capacity Sharing

In the previous section, efficient algorithms that offer bounds with respect to the optimal solution were presented without the use of backup capacity sharing. These results are useful for a basic understanding of the guaranteed recovery time problem using recovery domains (GRT-RD), and for networks that do not allow protection sharing. But many times, networks do utilize backup sharing, and significant savings can often be achieved. In this section, a time-efficient algorithm for GRT-RD using backup capacity sharing is presented.

If two primary flows for two different demands are edge-disjoint from one another, then under a single-link failure model, at most one can be disrupted at any given point in time. Since at most one demand will need to be restored after a failure, two failure-disjoint flows can share backup capacity. An interesting feature of recovery domain routing is that two demands can share backup capacity even if their two primary flows are not failure disjoint. Traditionally, in path protection schemes, two paths can only share protection resources if their primary paths are disjoint. However, in the recovery domain setting, sharing can take place between two recovery domains, so long as the primary segments in those recovery domains are disjoint. Thus, end-to-end primary paths that are not entirely disjoint may still share backup resources.

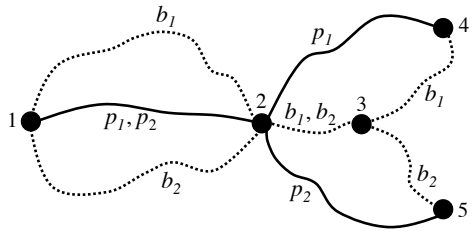


Fig. 7: Sharing protection resources in a recovery domain

An example of an end-to-end recovery domain routing for two demands is shown in Fig. 7. Demand 1 is routed from node 1 to node 4, and demand 2 is routed from node 1 to node 5. The primary paths are labeled p_1 and p_2 for demands 1 and 2, respectively, and backup paths are labeled b_1 and b_2. Two recovery domains are used for each demand: demand 1 uses recovery domains with the end-points of $(1, 2)$, and $(2, 4)$; demand 2 uses recovery domains with the end-points of $(1, 2)$, and $(2, 5)$. The two primary paths overlap in the first recovery domain on the path segment between nodes 1 and 2; hence, they cannot share protection resources within that domain, and each demand has its own dedicated backup path. The primary segments for the two recovery domains starting at node 2 and going to their respective destinations are failure disjoint. Even though the two primary paths overlap between nodes 1 and 2, the two demands can share protection resources for their recovery domains after node 2. On the path segment between nodes 2 and 3, only one unit of backup capacity allocation is needed to protect against a failure for either demand's primary path in recovery domains $(2, 4)$ or $(2, 5)$.

Since a failure is local to a recovery domain, and the primary flow outside of that recovery domain is not affected, a similar approach can be used as was done previously for the algorithms without backup capacity sharing: for every pair of nodes, we find the recovery domain routing that guarantees recovery time and utilizes backup capacity sharing. Then, an end-to-end recovery domain routing is constructed from a subset of those recovery domains using `end_to_end_RD` (Section IV-A).

Conflict sets are used to determine how much backup capacity can be shared by each incoming demand [19]. A conflict set indicates how much backup sharing is possible on an edge by examining how much backup capacity it already has to protect against any particular edge failure. If some edge has more backup capacity already assigned to it than is needed to protect against a particular edge failure, then those resources can be used at no additional cost. For example, let some edge $\{i, j\}$ have one unit of backup capacity allocated to it to protect against the failure of $\{k, l\}$, and with edge $\{i, j\}$ not being scheduled to protect against any other link failures. Now consider some new connection with a primary flow that uses some other edge $\{u, v\}$. Edges $\{k, l\}$ and $\{u, v\}$ can never fail simultaneously under a single-link failure model; thus, the new connection can use the backup capacity allocated to $\{i, j\}$ for protecting against the failure of $\{u, v\}$ without incurring additional cost. Further details of protection routing using conflict sets can be found in [19].

We consider routing demands dynamically (one-at-a-time), where once a connection is routed, it can no longer be changed; this model is similar to those used in path protection schemes [15, 19]. These path protection schemes offer heuristics to jointly optimize the primary and backup path for each incoming demand. We instead choose the simple strategy of using the shortest path for the primary route. After the primary path is found, the cost to use the remaining edges to protect that path can be determined (i.e., find out if edges can utilize protection resource sharing at no additional cost). If the maximum recovery time is T, and the shortest path has traversal time t_s, then a backup path is a constrained shortest path (CSP) that has a traversal time of at most $(T - t_s)$. To solve for the CSP, an optimal solution can be found in pseudo-polynomial time [30]; if T is rational and polynomial bounded with respect to the input parameters, the algorithm becomes polynomial. We label this algorithm `rd_sharing`. Our simulations show that using the shortest path for the primary route in fact performs better than jointly optimizing the primary and backup paths for each incoming demand.

To test the performance of `rd_sharing`, a similar simulation to Section III-B was run. The simulations were run using both the NSFNET and Lata 'X' topologies (Fig. 4) with 75 random unit demands. Each link's traversal time was set to be the propagation delay of that link, plus a 3 ms switching delay. Edges have random integer cost with uniform distribution between 1 and 5, and the maximum recovery time is set to the MPLS standard of 50 ms [1]. Three different schemes were tested: the optimal recovery domain routing with sharing, which jointly optimizes the primary and backup path for each incoming demand (using the MILP from Appendix A), local

Protection Scheme	NSFNET	Lata 'X'
Local Recovery with Sharing	583	3459
Optimal Recovery with Sharing	351	2677
End-to-end `rd_sharing`	349	2605

TABLE III: Cost of allocation for different protection schemes

recovery (FRR) with sharing (found in Appendix B), and the end-to-end `rd_sharing` algorithm developed in this section. A dynamic model was used: connections are serviced in the order of their arrival, and once a connection is routed, it can no longer be changed. Table III shows the total cost of allocation needed to route all 75 demands using the aforementioned protection schemes.

As anticipated, the optimal recovery scheme performs better than the local recovery scheme. Interestingly, the end-to-end `rd_sharing` algorithm performs *better* than the supposed optimal recovery with sharing. This can be explained by observing that the algorithm takes the simple strategy of the shortest path as the primary for each connection. This is as opposed to the optimal recovery scheme that jointly optimizes the primary and backup routes for each incoming demand, which may take a longer primary path to take advantage of backup sharing. By greedily optimizing every incoming demand, the potentially longer primary path makes it more difficult for future demands to find failure disjoint routes, lowering their ability to share protection resources, and thus increasing the overall cost.

VI. Conclusion

In this paper, we examined the problem of providing network protection with guaranteed recovery times using recovery domains (GRT-RD). The network is partitioned into failure-independent "recovery domains", where within each domain, the time to recover from a failure is guaranteed. To meet these guarantees, we provide an optimal solution in the form of an MILP. We demonstrate that the network-wide optimal solution can be decomposed into a set of more tractable and easier to solve subproblems. This allows for the development of flexible and efficient solutions, including an optimal algorithm using Lagrangian relaxation, which simulations show to converge rapidly to an optimal solution. Low complexity heuristics are developed for both with and without backup sharing. For dynamic arrivals, the algorithm utilizing backup sharing and using shortest paths performs better than the solution that greedily tries to optimize for each incoming demand.

Appendix

A. Guaranteed Recovery Time using Recovery Domains with Backup Capacity Sharing

The following values are given:

- V is the set of vertices, and E is the set of edges
- f^{sd} is the amount of flow that need to be routed from s to d, assumed to be integer
- c_{ij} is the cost of link $\{i,j\}$
- t_{ij} is the traversal time for link $\{i,j\}$
- T is the maximum recovery time

The following variables will be solved for:

- x_{ij}^{st} is 1 if flow is assigned on link $\{i,j\}$ or demand (s,t), and 0 o.w.
- $p_{ij,kl}^{st}$ is 1 if protection flow is assigned on link $\{i,j\}$ after the failure of link $\{k,l\}$ for demand (s,t), and 0 o.w.
- $y_{ij,kl}^{st}$ is 1 if spare capacity is assigned on on link $\{i,j\}$ for failure of link $\{k,l\}$ for demand (s,t) , and 0 o.w.
- w_{ij} is total primary flow on link $\{i,j\}$, $w_{ij} \geq 0$
- s_{ij} is total spare allocation on link $\{i,j\}$, $s_{ij} \geq 0$
- R_{kl}^{sd} is 1 if the recovery domain with end nodes k and l for demand (s,d) is active (part of the solution), 0 o.w.
- $r_{ij,kl}^{sd}$ is 1 if link $\{i,j\}$ is in recovery domain (k,l) for demand (s,d), 0 o.w.

The objective is to:

- Minimize the cost of allocation over all links:

$$\min \sum_{\{i,j\}\in E} c_{ij}(w_{ij}+s_{ij}) \tag{13}$$

Subject to the following constraints:

- Route primary traffic between s and d:

$$\sum_{\{i,j\}\in E} x_{ij}^{sd} - \sum_{\{j,i\}\in E} x_{ji}^{sd} = \begin{cases} 1 & \text{if } i=s \\ -1 & \text{if } i=d \\ 0 & \text{o.w.} \end{cases}, \quad \forall i \in V,\ \forall (s,d) \in (V,V) \tag{14}$$

- Route flow to protect after the failure of link $\{k,l\}$:

$$\sum_{\substack{\{i,j\}\in E \\ \{i,j\}\neq\{k,l\}}} p_{ij,kl}^{sd} - \sum_{\substack{\{j,i\}\in E \\ \{j,i\}\neq\{k,l\}}} p_{ji,kl}^{sd} = \begin{cases} 1 & \text{if } i=s \\ -1 & \text{if } i=d \\ 0 & \text{o.w.} \end{cases}, \quad \forall i \in V,\ \forall \{k,l\} \in E,\ \forall (s,d) \in (V,V) \tag{15}$$

- Primary capacity on link $\{i,j\}$ must meet all primary flows before a link failure

$$\sum_{(s,d)\in(V,V)} f^{sd} x_{ij}^{sd} \leq w_{ij},\ \forall \{i,j\} \in E \tag{16}$$

- Primary and spare capacity on link $\{i,j\}$ must be sufficient for all protection flows after failure of link $\{k,l\}$:

$$p_{ij,kl}^{sd} \leq x_{ij}^{sd} + y_{ij,kl}^{sd}, \quad \begin{matrix} \forall \{i,j\}\in E,\ \forall \{k,l\}\in E \\ \forall (s,d)\in(V,V) \end{matrix} \tag{17}$$

- Spare capacity on link $\{i,j\}$ satisfies all protection flows after failure of link $\{k,l\}$:

$$\sum_{(s,d)\in(V,V)} f^{sd} y_{ij,kl}^{sd} \leq s_{ij}, \quad \begin{matrix} \forall \{i,j\}\in E \\ \forall \{k,l\}\in E \end{matrix} \tag{18}$$

- Mark active recovery domains
  - If any edge in a recovery domain is active, then that recovery domain is active

$$\sum_{\{i,j\}\in E} r_{ij,kl}^{sd} \leq |E| \cdot R_{kl}^{sd}, \quad \begin{matrix} \forall (k,l)\in(V,V) \\ \forall (s,d)\in(V,V) \end{matrix} \tag{19}$$

  - If no edge in a recovery domain is active, then that recovery domain is not active

$$\sum_{\{i,j\}\in E} r_{ij,kl}^{sd} \geq R_{kl}^{sd}, \quad \begin{matrix} \forall (k,l)\in(V,V) \\ \forall (s,d)\in(V,V) \end{matrix} \tag{20}$$

- For each active recovery domain, find two disjoint paths between its respective end nodes k and l

$$\sum_{\{i,j\}\in E} r^{sd}_{ij,kl} - \sum_{\{j,i\}\in E} r^{sd}_{ji,kl} = \begin{cases} 2R^{sd}_{kl} & \text{if } i = k \\ -2R^{sd}_{kl} & \text{if } i = l \\ 0 & \text{o.w.} \end{cases},$$
$$\forall i \in V,\ (k,l) \in (V,V),\ \forall (s,d) \in (V,V) \quad (21)$$

- The sum of the traversed time delays of the edges in a given recovery domain cannot exceed the maximum recovery time

$$\sum_{\{i,j\}\in E} r^{sd}_{ij,kl} t_{ij} \leq T, \ \substack{\forall (k,l)\in (V,V) \\ \forall (s,d)\in (V,V)} \quad (22)$$

B. MILP for Optimal Local Recovery (Fast ReRoute)

The following is an MILP for optimal local recovery for a single demand requiring unit flow. Sharing can be accomplished by using similar techniques as the MILP for GRT-RD with sharing, shown in Appendix A.

The following values are given:

- V is the set of vertices and E is the set of edges
- c_{ij} is the cost of link $\{i,j\}$
- (s,d) is the source and destination

The following variables will be solved for:

- x_{ij} is 1 if primary flow is assigned to link $\{i,j\}$, and 0 otherwise
- f^{ij}_{kl} is 1 flow is assigned to link $\{i,j\}$ to protect after the failure of link $\{k,l\}$, 0 otherwise
- s_{ij} is 1 if protection flow is assigned to link $\{i,j\}$, and 0 otherwise

The objective is to minimize the total cost of allocation:

$$\text{minimize} \sum_{\{i,j\}\in E} c_{ij}(x_{ij} + s_{ij}) \quad (23)$$

Subject to the following constraints:

- Route primary traffic to meet demand before a failure

$$\sum_{\{i,j\}\in E} x_{ij} - \sum_{\{j,i\}\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \ \forall i \in V \quad (24)$$

- Route flow from point of local repair k to destination d after the failure of link $\{k,l\}$ if link $\{k,l\}$ carried flow

$$\sum_{\substack{j\in V \\ j\neq l}} f^{ij}_{kl} - \sum_{\substack{j\in V \\ j\neq l}} f^{ji}_{kl} = x_{kl}, \ \forall \{k,l\} \in E \quad (25)$$

$$\sum_{\substack{j\in V \\ j\neq k}} f^{jd}_{kd} - \sum_{\substack{j\in V \\ j\neq k}} f^{tj}_{kd} = -x_{kd}, \ \forall \{k,d\} \in E \quad (26)$$

$$\sum_{\substack{\{i,j\}\in E \\ \{k,l\}\neq\{i,j\} \\ i\neq k\vee d}} f^{ij}_{kl} - \sum_{\substack{\{j,i\}\in E \\ \{k,l\}\neq\{j,i\} \\ i\neq k\vee d}} f^{ji}_{kl} = 0, \quad \substack{\forall i\in V \\ \forall \{k,l\}\in E} \quad (27)$$

- Primary and spare capacity assigned on any link meets flow requirements after the failure of link $\{k,l\}$

$$f^{ij}_{kl} \leq x_{ij} + s_{ij}, \quad \substack{\forall \{i,j\}\in E \\ \forall \{k,l\}\in E} \quad (28)$$

REFERENCES

[1] V. Sharma and F. Hellstrand, "Framework for multi-protocol label switching (MPLS)-based recovery," IETF RFC 3469, Tech. Rep., 2003.

[2] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proceedings of the 2nd ACM SIGCOMM*. ACM, 2002, pp. 237–242.

[3] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic Engineering with MPLS in the Internet," *Network, IEEE*, vol. 14, no. 2, pp. 28–33, 2002.

[4] E. Rosen, A. Viswanathan, R. Callon *et al.*, "Multiprotocol label switching architecture," IETF RFC 3031, Tech. Rep., 2001.

[5] "Understanding mpls-tp and its benefits," White paper, Cisco, 2009. [Online]. Available: http://www.cisco.com/en/US/technologies/tk436/tk428/white_paper_c11-562013.pdf

[6] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," IETF RFC 4090, Tech. Rep., 2005.

[7] M. Kodialam and T. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2001, pp. 376–385.

[8] S. Raza, F. Aslam, and Z. Uzmi, "Online routing of bandwidth guaranteed paths with local restoration using optimized aggregate usage information," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 201–207.

[9] L. Li, M. Buddhikot, C. Chekuri, and K. Guo, "Routing bandwidth guaranteed paths with local restoration in label switched networks," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 110–120.

[10] R. Cohen and G. Nakibly, "Maximizing restorable throughput in MPLS networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 2, pp. 568–581, 2010.

[11] C. Huang and D. Messier, "A fast and scalable inter-domain MPLS protection mechanism," *Journal of Communications and Networks*, vol. 6, no. 1, pp. 60–67, 2004.

[12] N. Sprecher and A. Farrel, "MPLS-TP Survivability Framework," IETF RFC 6372, Tech. Rep., 2011.

[13] D. Xu, Y. Xiong, and C. Qiao, "Novel algorithms for shared segment protection," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 8, pp. 1320–1331, 2003.

[14] K. Wu, L. Valcarenghi, and A. Fumagalli, "Restoration schemes with differentiated reliability," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 3. IEEE, 2003, pp. 1968–1972.

[15] C. Ou, S. Rai, and B. Mukherjee, "Extension of segment protection for bandwidth efficiency and differentiated quality of protection in optical/mpls networks," *Optical Switching and Networking*, vol. 1, no. 1, pp. 19–33, 2005.

[16] S. Arakawa, J. Katou, and M. Murata, "Design method of logical topologies with quality of reliability in wdm networks," *Photonic Network Communications*, vol. 5, no. 2, pp. 107–121, 2003.

[17] J. Tapolcai, P. Ho, D. Verchére, T. Cinkler, and A. Haque, "A new shared segment protection method for survivable networks with guaranteed recovery time," *Reliability, IEEE Transactions on*, vol. 57, no. 2, pp. 272–282, 2008.

[18] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM Mesh Networks," *Journal of Lightwave Technology*, vol. 21, no. 4, p. 870, 2003.

[19] C. Ou, J. Zhang, H. Zang, L. Sahasrabuddhe, and B. Mukherjee, "New and Improved Approaches for Shared-Path Protection in WDM Mesh Networks," *Journal of Lightwave Technology*, vol. 22, no. 5, 2004.

[20] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, New Jersey, 1993.

[21] D. Bertsimas and J. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997.

[22] J. Suurballe and R. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, 1984.

[23] Y. Aneja and K. Nair, "The constrained shortest path problem," *Naval Research Logistics Quarterly*, vol. 25, no. 3, pp. 549–555, 1978.

[24] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, no. 4, pp. 293–309, 1980.

[25] J. Beasley and N. Christofides, "An algorithm for the resource constrained shortest path problem," *Networks*, vol. 19, no. 4, pp. 379–394, 1989.

[26] M. Ziegelmann, "Constrained shortest paths and related problems," Ph.D. dissertation, Universitat des Saarlandes, 2001.

[27] J. Yen, "Finding the k shortest loopless paths in a network," *management Science*, pp. 712–716, 1971.
[28] E. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, pp. 401–405, 1972.
[29] A. Orda and A. Sprintson, "Efficient algorithms for computing disjoint qos paths," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1. IEEE, 2003.
[30] H. Joksch, "The shortest route problem with constraints," *Journal of Mathematical analysis and applications*, vol. 14, pp. 191–197, 1966.