Oblivious Routing Using Learning Methods

Ufuk Usubütün*, Murali Kodialam†, T.V. Lakshman†, Shivendra Panwar*
*New York University Tandon School of Engineering, Brooklyn, NY
Email: {usubutun, panwar}@nyu.edu
†Nokia Bell Labs, Murray Hill, NJ
Email: {murali.kodialam, tv.lakshman}@nokia-bell-labs.com

***Abstract*—Oblivious routing of network traffic uses predetermined paths that do not change with changing traffic patterns. It has the benefit of using a fixed network configuration while robustly handling a range of varying and unpredictable traffic. Theoretical advances have shown that the benefits of oblivious routing are achievable without compromising much capacity efficiency. For oblivious routing, we only assume knowledge of the ingress/egress capacities of the edge nodes through which traffic enters or leaves the network. All traffic patterns possible subject to the ingress/egress capacity constraints (also known as the hose constraints) are permissible and are to be handled using oblivious routing. We use the widely deployed segment routing method for route control. Furthermore, for ease of deployment and to not deviate too much from conventional shortest path routing, we restrict paths to be 2-segment paths (the composition of two shortest path routed segments). We solve the 2-segment oblivious routing problem for all permissible traffic matrices (which can be infinitely-many). We develop a new adversarial and machine-learning driven approach that uses an iterative gradient descent method to solve the routing problem with worst-case performance guarantees. Additionally, the parallelism involved in descent methods allows this method to scale well with the network size making it amenable for use in practice.**

***Index Terms*—Oblivious Routing, Segment Routing, Gradient Descent, Machine Learning, Adversarial Learning**

I. Introduction

Classical network optimization literature provides various methods for solving multi-commodity flow routing problems when the network graph, link capacities and the demands are all known. However, in today's networks with a constantly growing number of users and with diversified applications to support, accurately predicting traffic demand is often a challenge. Virtualization and continuous migration of Internet services and occasional failures in adjacent networks can also lead to unpredictable demand fluctuations. Despite the changing network traffic conditions, for operational simplicity it is preferable to avoid frequent configuration changes. One approach to accomplish this while still making good use of network resources is traffic oblivious routing.

With traffic oblivious routing the goal is to find a fixed routing that is agnostic to the current traffic demand while not leading to poor network performance. Work on developing routing approaches that work over a wide variety of traffic instances was initiated by [1]. More recently, [2] outlined a routing scheme that provides relative congestion performance guarantee (compared to the optimal routing scheme) for any traffic matrix. Our paper follows the approach taken in [3]–[5] where the set of permissible traffic patterns are constrained by natural network capacity constraints termed the hose constraints. Instead of obtaining relative performance guarantees, we want to obtain absolute performance guarantees over this class of traffic matrices. To achieve this goal, instead of solving the routing problem for a single traffic matrix, we need to find a single routing that works well (provides a worst case bound) for every permissible traffic matrix subject to the hose constraints since it only requires knowing the ingress/egress capacities of the network's edge nodes. However, we need to address the challenge of finding the optimized routing for the infinitely-many traffic matrices that are permissible within the hose constraints. This oblivious routing solution has the potential to avoid over-provisioning while robustly handling fluctuating traffic demands. [6].

Segment Routing (SR) [7] has been widely deployed in networks for route control. With segment routing, a source-destination path in the network is composed of multiple segments where each segment uses shortest-path routes, including possibly equal-cost multi-path (ECMP) routes, computed by the underlying routing protocol. Control of the routing path is accomplished by an appropriate choice of the segment end-points. In this paper, we use segment routing as the method of choice for route control and solve the corresponding oblivious routing problem. A similar approach was taken by [8]; however our method provides a higher degree of freedom for routing and does not have any restrictions on the in/out constraints. For ease of deployment, we consider only 2 segment paths, where each network path is the composition of at most two segments determined by choosing a single intermediate node where the first segment ends and the second segment begins.

Solving our oblivious routing problem involves optimizing over infinitely many traffic matrices. For this aspect, we see a parallel with machine learning methods. Adversarial learning approaches are a class of methods that emerged within the field of deep learning for improving the robustness of neural networks [9]. In adversarial learning, given a training solution, new data points are synthetically generated in order to fail the system. These adversarial data points are then fed back into the training in order to improve the robustness of the solution. The idea, therefore, is to gradually find a solution

This research was supported by the New York State Center for Advanced Technology in Telecommunications and Distributed Systems (CATT), NYU Wireless, the Cisco University Research Program Fund and by the National Science Foundation (NSF) under Grant No. CNS-2148309 and OAC-2226408.

by constantly improving against adversarially generated worst case scenarios. In this paper we propose a novel technique inspired by adversarial learning to be used for oblivious routing problems. Building on this idea, we switch back and forth between solving the routing problem for a small set of traffic matrices and generating the worst case traffic matrices for our current solution until we converge to an optimal solution. To solve the constrained optimization problems, we reshape them in ways that allow us to use descent based methods that are highly parallelized and efficient when compared with linear programming (LP) based approaches to oblivious routing that face computation and scalability issues [6]. We also provide two relaxations of the oblivious routing problem that use upper bounds on the traffic demand and do not require an adversarial method. We implement a solver and provide a proof of concept. Our contributions in this work can be listed as follows:

- Formulate a traffic oblivious, 2-SR problem that follows from the ingress/egress capacities, i.e., hose constraints.
- Propose an adversarial approach to efficiently solve this problem using gradient descent.
- Propose two relaxations of the problem that yield solutions that provide bounds on the optimal solution.
- Show the proof of concept with an implementation.

After setting up the notation, in Section III we introduce oblivious routing with hose constraints and then in Section IV we formulate the 2-SR problem with the hose constraints. In Section V, we present our adversarial method to solve the oblivious routing problem. We present relaxations of the formulated problem in Section VI. Next, in Section VII, we present performance benchmarks. Concluding remarks are in section VIII.

II. Notation

Let us represent a network using a directed graph $G(N, E)$ with a set of nodes N and a set of directed edges E. We let each node be both a source and a destination for the traffic to be routed. The nodes in N are labeled $\{1, 2, \ldots, n\}$. We denote each directed edge from i to j with (i, j) where $i, j \in N$ and $(i, j) \in E$. For simplifying the notation we also use $e \in E$ to refer to an edge. We use $|.|$ as the cardinality operator. We express the capacity of link e as u_e. We define the utilization of a link as the total load on that link divided by its capacity.

III. Oblivious Routing with Hose Constraints

Unlike challenges involved with characterization of expected traffic patterns in a network, the network topology and link capacities are more easily available from the network. As was done in [5], one can use some of this information about the network, to characterize all observable traffic on that network. Let us assume that network G is connected with the outside world at every node i and that we expect all traffic to originate and terminate outside our network. Observe for the connection with the outside world that the egress capacity of each node provides an upper limit on the amount of traffic that can leave that node at a given moment. Likewise, the ingress capacities limit, how much traffic can be observed to enter our network at a given time. Let us denote the ingress capacity of a node i with R_i and the egress capacity with C_i. We can then claim the following for each traffic matrix $[t_{ij}]$ that can be observed at network G:

$$\sum_{j \in N, j \neq i} t_{ij} \leq R_i \qquad \sum_{j \in N, j \neq i} t_{ji} \leq C_i \qquad \forall i \in N \tag{1}$$

This corresponds to having limits on the row and column sums of any observable traffic matrix imposed by physical specifications of the network. We can use this information to describe all observable, or hose-feasible, traffic matrices $[t_{ij}]$ on network G with the following set:

$$\mathcal{T}(\mathcal{R}, \mathcal{C}) = \left\{ [t_{ij}] \middle| \sum_{j \neq i} t_{ij} \leq R_i \text{ and } \sum_{j \neq i} t_{ji} \leq C_i, \ \forall i \in N \right\} \tag{2}$$

Observe that we are characterizing all observable traffic in a network and not using any current measurements or estimations about the traffic to be carried. In the next section we formulate a flow placement problem that aims to provide a fixed optimal solution for every traffic matrix in set $\mathcal{T}(\mathcal{R}, \mathcal{C})$.

IV. System Model with Segment Routing

Segment routing (SR) is deployed in networks and can be used for controlling routing in the network to be along specific paths. In SR, routing paths are broken down into multiple segments. For traffic engineering, one needs to pick which intermediate nodes are to be visited on the way to the final destination. These intermediate nodes are known at the source node, which is in charge of appending each of these intermediate destinations to the packet header. Packets are forwarded through the chosen segments and used labels are popped at each intermediate node. Segment routing is defined to work with any number of intermediate nodes. However, most of the benefits in terms of optimizing network utilization can be achieved by just using two segments [10] (more segments may be needed if traffic needs to be routed through several mid-boxes). We therefore choose to limit our attention to 2-segment routing in this work, that is for each flow traveling through the input-output pair $i \rightsquigarrow j$ we pick one intermediate node $k \in N$. Our proposed method however, can easily be extended to n-segment routing.

We define our model following a similar approach to [10]. Let $SP(i, j)$ be the set of links e on the shortest path from the node i to j. With 2-segment routing, if we are given a flow from i to j, and we chose to route this flow over an intermediate node k, this flow will travel through the links in $SP(i, k)$ and then through $SP(k, j)$. In order to map a shortest path $SP(i, j)$ to a given link e on graph G, define the *mapping coefficient function* $f_{ij}(e) \in [0, 1]$. If the shortest path is unique with respect to the cost metric, then $f_{ij}(e) = 1$ for each link $e \in SP(i, j)$, and $f_{ij}(e) = 0$ for all other links $e \notin SP(i, j)$. If the there exists multiple equal cost paths, assuming flows are splittable, values of $f_{ij}(e)$ can also be fractional in order to represent an ECMP splitting [11]. An

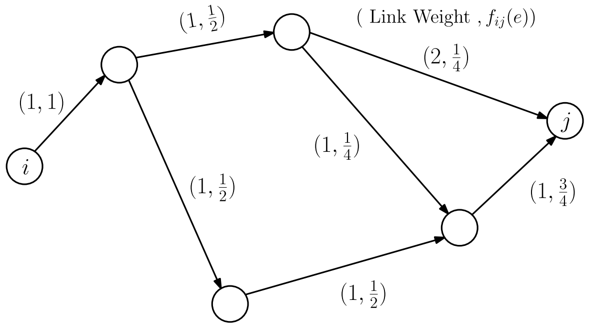

Fig. 1. Definition of $f_{ij}(e)$ with ECMP. The first number next to the link represents the link weight and the second number is $f_{ij}(e)$. The shortest path length is 4 and there are three shortest paths.

example scenario for the ECMP case is demonstrated in Figure 1. This mapping can naturally be extended to segment routing, where each segment is routed through the shortest path. Let $g_{ij}^k(e)$ be the *mapping coefficient function* of the 2 segment route $i \rightsquigarrow k \rightsquigarrow j$ to some link e. It is equal to the following:

$$g_{ij}^k(e) = f_{ik}(e) + f_{kj}(e) \tag{3}$$

The values of *mapping coefficient functions* are fixed for a given network G and only have to be calculated once, which can be done fairly easily.

For the purpose of traffic engineering, let us define *traffic split ratios* $\alpha_{ij}^k \in [0,1]$. Assuming flows are splittable, α_{ij}^k describes the ratio of traffic from i to j that will be routed over the intermediate node k. In order to distribute all traffic we must have $\sum_k \alpha_{ij}^k = 1, \forall i,j$. In case an arbitrary traffic matrix $[\tilde{t}_{ij}] \in \mathcal{T}(\mathcal{R},\mathcal{C})$ is observed at the network, the amount of traffic traveling from i to j routed over node k will be equal to $\tilde{t}_{ij}\alpha_{ij}^k$. Therefore, the amount of traffic from i to j routed over node k will cause $\tilde{t}_{ij}\alpha_{ij}^k g_{ij}^k(e)$ amount of load on a link e. The total load $F(e)$ on some link e can then be found using:

$$F(e) = \sum_{ijk} \tilde{t}_{ij}\alpha_{ij}^k g_{ij}^k(e) \tag{4}$$

Using the defined symbols, we formulate the following traffic oblivious 2-SR routing problem. The formulation aims to minimize the maximum link utilization by tuning *traffic split ratios*.

$$\min_{\alpha_{ij}^k} \mu \tag{5}$$

$$\text{s.t.} \quad \sum_{ijk} t_{ij}\alpha_{ij}^k g_{ij}^k(e) \le \mu u_e \quad \forall [t_{ij}] \in \mathcal{T}(\mathcal{R},\mathcal{C}), \quad \forall e \in E \tag{6}$$

$$\sum_k \alpha_{ij}^k = 1 \quad \forall i,j \in N \tag{7}$$

$$\alpha_{ij}^k \ge 0 \quad \forall i,j,k \in N \tag{8}$$

Constraints in (6) aim to bound the maximum link utilization and constraints on (7) and (8) make sure that all traffic is routed. If a solution $\mu \le 1$ can be found, this means, there exists a feasible fixed routing solution that can optimally handle all permissible traffic matrices.

Note that the utilization constraints in (6) are infinitely many as these relations have to be satisfied for every matrix from the set $\mathcal{T}(\mathcal{R},\mathcal{C})$. This brings a fair amount of complexity to the problem since solving the problem involves calculating of the worst case traffic matrix at each evaluation. There exist approaches that utilize LP dual formulations to find a solution to different variants of the oblivious routing problem with hose constraints. Those approaches however are computationally expensive and scale poorly with network size [5]. In the next section, we propose a novel adversarial approach, inspired by machine learning, that allows us to simplify the problem and solve it.

V. Adversarial Learning Approach

The oblivious routing problem with hose constraints presented in Section IV involves satisfying infinitely many demand constraints and presents a challenge. To approach the problem differently, we propose a novel adversarial approach, inspired by machine learning, that allows us to capture the dynamics of the infinitely dimensioned set $\mathcal{T}(\mathcal{R},\mathcal{C})$ of hose-feasible traffic matrices without having to iterate the infinitely many constraints in (6). We achieve this by breaking the original problem down into two simpler and competing problems and tackle them in alternation. In doing this, we reshape the problem to make it compatible with gradient optimization tools. This new formulation allows us to exploit the parallelism capabilities of modern gradient descent tools.

The optimization problem presented in Section IV, can instead be solved by tackling the following two sub-problems in alternation:

(i) Generating the worst case traffic matrices for a given routing solution α_{ij}^k.
(ii) Solving for the optimal *traffic split ratios* α_{ij}^k for the finitely many matrices generated so far.

We first define the two sub-problems and discuss how they can individually be solved. We then present our methodology for obtaining the optimal solution for the oblivious routing problem. Let us begin with sub-problem (ii) which we call the **Optimization Step**: Assume that we are given a finite set $\mathcal{L}$ of traffic matrices and we would like to tune the split ratios α_{ij}^k to obtain the fixed routing solution that results in the lowest maximum link utilization. This corresponds to replacing $\mathcal{T}(\mathcal{R},\mathcal{C})$ with $\mathcal{L}$ in equation (6) and results in $|\mathcal{L}| \cdot |E|$ constraints. By reshaping the problem into a $\min\max$ formulation, we can eliminate all of the link utilization constraints by embedding them into the objective function. To simplify the notation, assume all matrices in set $\mathcal{L}$ are indexed and define the set $\mathcal{L}_I$ to represent these indices. We obtain the following problem of finding the optimal routing solution α_{ij}^k over the set $\mathcal{L}$:

$$\min_{\alpha_{ij}^k} \left[\max_{e \in E, l \in \mathcal{L}_I} \frac{\sum_{ijk} t_{ij}^{(l)} \alpha_{ij}^k g_{ij}^k(e)}{u_e} \right] \tag{9}$$

$$\text{s.t.} \quad \sum_k \alpha_{ij}^k = 1 \qquad \forall i,j \in N \tag{10}$$

$$\alpha_{ij}^k \ge 0 \qquad \forall i,j,k \in N \tag{11}$$

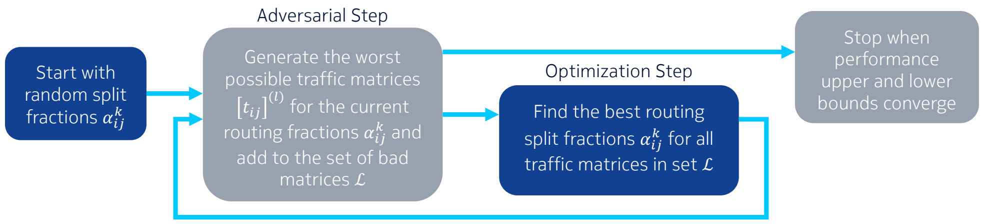
Adversarial Step
Start with random split fractions α_{ij}^k
Generate the worst possible traffic matrices $[t_{ij}]^{(l)}$ for the current routing fractions α_{ij}^k and add to the set of bad matrices $\mathcal{L}$
Optimization Step
Find the best routing split fractions α_{ij}^k for all traffic matrices in set $\mathcal{L}$
Stop when performance upper and lower bounds converge

Fig. 2. Adversarial Optimization Flowchart

Observe that remaining constraints involve keeping α_{ij}^k non-negative and ensuring the terms add up to 1 when summed over k. The *softmax* function, widely used in machine learning, is a function that takes in n real numbers and maps them to a discrete probability distribution with n outcomes, i.e., n numbers adding up to 1. This function provides a great opportunity for eliminating the remaining constraints in (10) and (11). Let us define a new dummy variable y_{ij}^k corresponding to each α_{ij}^k and relate them to one another using the *softmax* function, denoted $SM(\cdot)$, as follows:

$$\alpha_{ij}^k = SM(y_{ij}^k) = e^{y_{ij}^k} / (\textstyle\sum_{\tilde{k}} e^{y_{ij}^{\tilde{k}}}) \tag{12}$$

Observe that, regardless of the values of y_{ij}^k, the α_{ij}^ks obtained through (12) always satisfy the constraints in (10) and (11). We can, therefore, update our formulation once again to include the *softmax* function and eliminate all constraints.

$$\min_{y_{ij}^k} \left[\max_{e \in E, l \in \mathcal{L}_I} \frac{\sum_{ijk} t_{ij}^{(l)} SM(y_{ij}^k) g_{ij}^k(e)}{u_e} \right] \tag{13}$$

By taking advantage of the *softmax* function, we have been able to reduce the problem of finding the optimal split ratios for finitely many matrices into a non-linear unconstrained optimization problem. This formulation in (13) can easily be fed into any optimization tool used for machine learning that is able calculate gradients and be solved with the desired choice of descent algorithm.

Now, with knowledge of solving the optimal routing problem for finitely many matrices from set $\mathcal{L}$, let us formulate sub-problem (i) of generating worst case traffic matrices to build the set $\mathcal{L}$. We call this the **Adversarial Step**: As our aim is to minimize the maximum link utilization in the network for all permissible traffic, a worst case traffic matrix for link e can be defined to be a matrix $[t_{ij}] \in \mathcal{T}(\mathcal{R}, \mathcal{C})$ that results in the maximum utilization of link e, given split choices α_{ij}^k. As link capacities u_e are fixed, this corresponds to maximizing the total load on the link $F(e)$ as defined in (4) and can be formulated as the following LP:

$$[t_{ij}]^{(l)} = \arg \max_{[t_{ij}]} \sum_{ijk} t_{ij}^l \alpha_{ij}^k g_{ij}^k(e) \tag{14}$$

$$\text{s.t.} \quad \sum_{i, i \neq j} t_{ij}^l \leq C_j \qquad \forall j \in N \tag{15}$$

$$\sum_{j, j \neq i} t_{ij}^l \leq R_i \qquad \forall i \in N \tag{16}$$

$$t_{ij}^l \geq 0 \qquad \forall i, j \in N \tag{17}$$

This problem may yield infinitely many solutions as certain entries of t_{ij}^l may not contribute to traffic on link e either due to the split choices or due to the mapping function. We choose any one of such solutions as the the free terms have no effect on the worst case utilization of link e. In order to cover all links in the network, this problem should be solved $|E|$ times for each link $e \in E$, producing $|E|$ new traffic matrices to be appended to set $\mathcal{L}$.

Having both sub-problems formulated, we now present our adversarial approach for traffic oblivious 2-SR routing with hose constraints. Our approach, which involves switching back and forth between the Adversarial and Optimization Steps, is summarized in a flowchart presented in Fig 2. Given a network graph $G(N, E)$, link capacities u_e and ingress/egress capacities R and C, we start with randomly generated dummy variables y_{ij}^k and obtain the corresponding split fractions α_{ij}^k through the *softmax* function presented in (12). This allows us to start the first execution of the Adversarial Step in which we generate $|E|$ traffic matrices $[t_{ij}]^{(l)}$ which maximize the load on their respective links e for the current split fractions α_{ij}^k. We append all the matrices generated in this step to the precedently empty set $\mathcal{L}$. At the end of each Adversarial Step, before y_{ij}^ks are modified, the value of the objective function of the Optimization Step, i.e.,

$$\max_{e \in E, l \in \mathcal{L}_I} \frac{\sum_{ijk} t_{ij}^{(l)} SM(y_{ij}^k) g_{ij}^k(e)}{u_e} \tag{18}$$

will be equal to the worst case performance performance of current split fractions α_{ij}^k over the entire set $\mathcal{T}(\mathcal{R}, \mathcal{C})$, i.e., all permissible traffic matrices. Therefore, this defines an upper bound on the worst case routing performance for the current split fractions. We then move on to the Optimization Step. We tune the dummy variables y_{ij}^k with gradient descent in order to generate the routing solution that leads to the lowest maximum link utilization over all the traffic matrices in set $\mathcal{L}$. Once the descent converges, we save the new split fractions. If evaluated with the new split fractions, equation (18) this time gives us a lower bound on the worst case performance over the set $\mathcal{T}(\mathcal{R}, \mathcal{C})$. We continue going back and forth between the two steps, continually growing the set $\mathcal{L}$ and tuning α_{ij}^ks while noting down the upper and lower bound values.

In Figure 3 an example optimization run is presented where the worst case link utilization of each link on the network is depicted over iterations. Before the first iteration begins, split fractions are randomly picked and the worst case traffic matrices corresponding to each link are calculated. These $|E|$

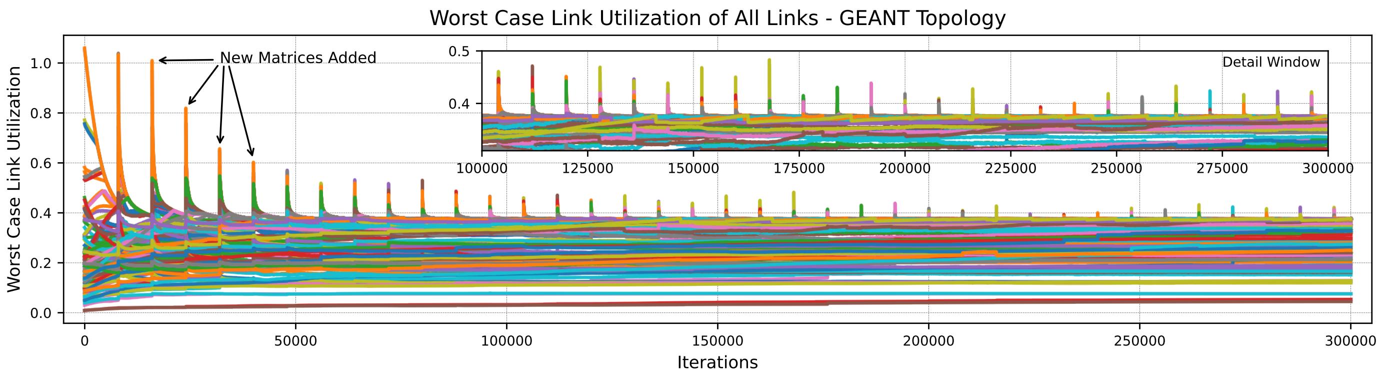

Fig. 3. The worst case utilization of each link is shown while the active traffic split fractions α_{ij}^k are being tuned with gradient descent over the iterations of Optimization Steps. The worst case of each link is found over the traffic matrices from the set $\mathcal{L}$ and therefore the maximum value corresponds to the value in equation (18). Each Optimization Step is followed by an Adversarial step where the set $\mathcal{L}$ is appended with traffic matrices producing the worst link utilization for the solution α_{ij}^k at every 8000th iteration. The results are shown for the GEANT topology with parameters in Section VII.

matrices, which constitute the set $\mathcal{L}$, lead to poor utilization values. Then, the the Optimization Step begins and tunes split fractions over iterations. At each 8000th iteration, the Adversarial Step is called to expand the set $\mathcal{L}$. This leads to peaks that are visible in the worst case. The Optimization Step then works to find a new solution. As can be seen, the upper bound does not strictly decrease at each Adversarial Step as the optimization step taken over the finite set $\mathcal{L}$ may push the solution away from the optimum value. However, when executed for sufficiently long the two quantities are expected to converge. Therefore, the procedure can be terminated when the lower and upper bound values are within a desired ϵ away from each other. In Section VII we provide performance statistics for select real backbone network topologies.

In the next section we present two relaxations of the oblivious routing problem that does not require running adversarial episodes. These formulations provide bounds on the worst case performance of our Adversarial Method and are further able to provide sub-optimal routing solutions.

VI. Relaxations for Obtaining Performance Bounds

In this section, we present two simpler formulations that are inspired by [12] that do not require generation of worst case traffic matrices. These methods are instead able to provide a sub-optimal routing solution using only the ingress/egress capacity information R and C. As they are computationally much simpler than the adversarial method, they provide easy to compute upper bounds on the worst case performance of the optimal solution. In order to derive these methods, let us use a slightly different but equivalent expression to express the total load $F(e)$ from what we defined in (4). Let us first define ϕ_{ij} to be the total amount of traffic induced on an SR segment from node i to j. Keep in mind that an SR segment corresponds to a logical mapping and that real traffic goes through all nodes on the shortest path $SP(i,j)$. In that case, we can express the total load $F(e)$ as:

$$F(e) = \sum_{ij} \phi_{ij} f_{ij}(e) \tag{19}$$

As 2-SR is used, any segment is either the first or the second along the way. Therefore, the quantity ϕ_{ij} consists of the aggregate of all traffic that uses the segment $i \rightsquigarrow j$ either as the first or as the second segment. Assuming an arbitrary traffic matrix $[\tilde{t}_{ij}] \in \mathcal{T}(\mathcal{R},\mathcal{C})$ is observed at the network, the total load on logical segment $i \rightsquigarrow j$ can be expressed as the following:

$$\phi_{ij} = \sum_k \alpha_{ik}^j \tilde{t}_{ik} + \sum_k \alpha_{kj}^i \tilde{t}_{kj} \tag{20}$$

where the first term captures first segments and second term captures second segments. Note that ϕ_{ij} depends on the traffic matrix. However the ingress/egress capacity information R and C can be used to obtain an upper bound on the term that does not depend on the individual traffic matrix. We present two such methods. The first one that we call **Non-Uniform Splitting**, uses the same system model introduced in Section IV. By following the inequalities below we obtain an upper bound on the quantity ϕ_{ij}:

$$\phi_{ij} \leq (\max_k \alpha_{ik}^j) \sum_k \tilde{t}_{ik} + (\max_k \alpha_{kj}^i) \sum_k \tilde{t}_{kj} \tag{21}$$

$$\leq (\max_k \alpha_{ik}^j) R_i + (\max_k \alpha_{kj}^i) C_j \tag{22}$$

Replacing the term ϕ_{ij} in equation (19) with the upper bound in (22), provides an upper bound on $F(e)$, the total load on link e.

We formulate the following new oblivious routing problem using this upper bound on $F(e)$. Using the same *softmax* approach introduced in Section III, allows us to eliminate the non-negativity and summation constraints of (10) and (11).

$$\min_{y_{ij}^k} \max_{e \in E} \frac{1}{u_e} \sum_{ij} f_{ij}(e) \Big[\max_k SM(y_{ik}^j) R_i + \max_k SM(y_{kj}^i) C_j \Big] \tag{23}$$

Note that this relaxation can directly be solved with gradient descent as it does not require generation of traffic matrices, or tackling competing problems. The solution to this problem also provides a worst case performance guarantee.

We formulate a simpler problem that we call **Uniform Splitting** by reducing the split choice space. We let $\alpha_{ij}^k = \alpha^k$, that is, we map all traffic over node k regardless of origin and destination. Following a similar approach, we obtain an upper bound on the quantity ϕ_{ij}:

$$\phi_{ij} = \alpha^j \sum_k \tilde{t}_{ik} + \alpha^i \sum_k \tilde{t}_{kj} \tag{24}$$

$$\leq \alpha^j R_i + \alpha^i C_j \tag{25}$$

By using the same *softmax* approach, we obtain the following optimization problem:

$$\min_{y_{ij}^k} \max_{e \in E} \frac{1}{u_e} \sum_{ij} f_{ij}(e) \left[SM(y^i) C_j + SM(y^j) R_i \right] \tag{26}$$

Like the Non-Uniform Splitting formulation, this formulation can also be solved with gradient descent and provide worst case performance guarantees.

VII. Results

We implemented all three methods in `pytorch` in order to demonstrate a proof of concept. For all three methods we used the Adam optimizer to descend over the gradient. Our implementation of the Adversarial Method uses fixed length descent episodes in its Optimization Step that does not involve fine tuning with respect to stop conditions. In each run, we executed the algorithm for a total of 300000 iterations, used fixed 8000 iteration episodes of gradient descent for the Optimization Step and took advantage of the `CBC` LP solver in `pulp` to conduct the Adversarial Step. After a run, we pick the *traffic split solution* α_{ij}^k that leads to the best worst-case performance and report the worst-case utilization corresponding to that solution. We tested our method over multiple real life backbone network topologies obtained through The Network Zoo [13]. We used uniform link capacities $u_e = 1, \forall e$ and uniform ingress/egress capacities $R_i = C_i = 0.1, \forall i$.

The transient behavior of our adversarial method is presented in Figure 3 for the GEANT topology. This transient behavior was discussed in detail in Section V. Table I presents the achieved worst case link utilization of each method. These values are normalized with respect to the worst case link utilization when the shortest path ECMP routing was used. These ratios allow us to compare the worst case link utilization of the proposed methods to that of shortest path routing with ECMP, and their comparison is only meaningful within the same topology. Thanks to the achieved load-balancing, we see notable improvements with respect to the shortest path scenario for all cases. We also see that the Adversarial Method always performs better than the Non-Uniform Splitting method which in turn always performs better than the Uniform Splitting method as expected. Both of the relaxation methods, which are much simpler to solve compared to the Adversarial approach, obtain a similar performance and provide an upper bound as expected.

TABLE I
WORST CASE LINK UTILIZATION NORMALIZED WITH RESPECT TO SHORTEST PATH WORST CASE LINK UTILIZATION

			Normalized Worst Case Utilization		
Topology	**Nodes**	**Edges**	Unif. Sp.	N-Unif Sp.	Adversarial
Sprint	11	18	61.1%	60.6%	56.0%
GoodNet	17	31	38.8%	37.2%	33.6%
GEANT	40	61	51.3%	49.9%	46.3%
GARR	61	89	57.7%	56.5%	52.3%
Intellifiber	73	97	82.3%	79.3%	71.6%

VIII. Conclusions and Future Work

We have demonstrated that adversarial approaches that originated within machine learning can also be used to solve traffic oblivious routing problems. The technique, unlike previous attempts, scales to large scale problems and can be used to derive routing schemes that provide worst case guarantees over the range of hose-feasible traffic matrices. Furthermore, we provided relaxations to the problem that can be easily solved and that can be used to obtain performance estimates of the Adversarial Method. The approach in this paper can be extended to the problem of routing with latency bounds as well routing through middle-boxes by restricting the set of points through which traffic can be routed.

References

[1] L. G. Valiant and G. J. Brebner, “Universal schemes for parallel communication,” in *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, 1981, pp. 263–277.

[2] H. Racke, “Minimizing congestion in general networks,” in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 43–52.

[3] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, “A flexible model for resource management in Virtual Private Networks,” in *Proc. ACM SIGCOMM*, 1999, pp. 95–108.

[4] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, “Algorithms for provisioning Virtual Private Networks in the hose model,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 565–578, 2002.

[5] M. Kodialam, T. Lakshman, and S. Sengupta, “Traffic-oblivious routing in the hose model,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 774–787, 2011.

[6] M. Kodialam, T. Lakshman, J. B. Orlin, and S. Sengupta, “Oblivious routing of highly variable traffic in service overlays and IP backbones,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 459–472, 2008.

[7] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, “Segment Routing Architecture,” RFC 8402, Jul. 2018.

[8] M. Antic and A. Smiljanic, “Oblivious routing scheme using load balancing over shortest paths,” in *Proc. IEEE ICC*, 2008, pp. 5783–5787.

[9] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, “Recent advances in adversarial training for adversarial robustness,” in *30th International Joint Conference on Artificial Intelligence*, 2021, pp. 4312–4321.

[10] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, “Optimized network traffic engineering using Segment Routing,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 657–665.

[11] C. Hopps, “Analysis of an Equal-Cost Multi-Path Algorithm,” RFC 2992, Nov. 2000.

[12] M. Kodialam, T. V. Lakshman, and S. Sengupta, “Efficient and robust routing of highly variable traffic,” in *3rd Workshop on Hot Topics in Networks*. ACM, November 2004.

[13] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The internet topology zoo,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.