Traffic Engineering with Precomputed Pathbooks

Mathieu Leconte, Apostolos Destounis, and Georgios S. Paschos
Huawei Technologies, Mathematical and Algorithmic Sciences Laboratory, Paris, France

***Abstract*—This paper addresses a major challenge of traffic engineering; the selection of a set of paths that minimize routing cost for a random traffic matrix. We introduce the concept of *pathbook*, a small set of paths to which we restrict routing. The use of pathbook accelerates centralized traffic engineering algorithms, and therefore is appealing for instantiating, configuring, and optimizing large software-based networks. However, restricting routing to a few paths may lead to higher cost or infeasibility. To this end, we introduce the problem of *pathbook design*, wherein we search for a pathbook of constrained size that minimizes the expected routing cost of the random traffic matrix, which represents a prediction of the future traffic. The pathbook design problem is of combinatorial nature, and we show that it is NP-hard. For its convex relaxation, we derive an optimal solution based on a projected subgradient method. The optimal convex pathbook can be used to provide a suboptimal integral pathbook. For large networks, the subgradient vector is large and challenging to compute, hence we propose a coordinate-descent method using the Gauss-Southwell rule, which prescribes a move along the direction of largest subgradient element. We test the performance of our solution on available data for dynamic traffic matrices in the GEANT network and find benefits as much as one and two orders of magnitude in terms of runtime with respect to standard routing methods.**

I. Introduction

Future wireless and wired networks will be based on software. Technologies such as Software Defined Networking (SDN) decouple control from data plane and allow to manage and optimize the network from a central point, the network controller. During deployment, the controller receives the application requirements and solves large traffic engineering problems in order to determine the desirable configuration of the virtual network, which is then instantiated by means of protocols like OpenFlow [1]. Similarly, during operation, the controller continuously receives real-time monitoring information about the network state and solves large traffic engineering problems in order to compute the desirable network reconfiguration [2]. These optimization problems are of combinatorial nature, and are often the bottleneck in execution time of the entire network (re)configuration. For example, the LP-relaxation of global routing in a 500-node IP network may take hours to compute with a standard optimization tool like CPLEX. *This paper proposes the acceleration of traffic engineering algorithms by means of precomputing paths based on data monitoring.*

In order to compute desirable network configurations, we focus on the *Minimum Cost Multicommodity Flow* (MCF) problem on a static topology with link capacities and costs. MCF is one of the most important and challenging problems in traffic engineering []–the importance of MCF for future virtualized networks is further explained in Section II. For each source-destination pair, a demand with certain bandwidth must be routed through the network over multiple paths, subject to link capacity constraints. The goal is to route the flow through cheap links in order to minimize the total cost. The instance of this problem is determined by the network graph (and its attributes), and the *traffic matrix*: the collection of bandwidth demands of each source-destination pair. Typically, traffic matrices fluctuate over time [3]. In SDN, past monitorned traffic matrices can be processed with machine learning tools to extract a distribution of traffic matrices as a prediction model for future traffic. The goal of this paper is to find a path set restriction of limited size that achieves the smallest expected routing cost under a distribution of traffic matrices.

More specifically, we make two observations: (1) solving MCF for large networks is slow, but can be accelerated if we restrict the problem to a small set of paths, (2) experimental data from GEANT network show that the union of solutions over 11642 traffic matrices scaled at 99% loading, requires no more than six (6) paths per commodity. This naturally leads to the pathbook design problem: *find a small set of paths that minimizes the expected routing cost over a distribution of traffic matrices*. Solving this problem optimally allows to predetermine the set of paths of constrained size that best serves the forecast traffic. Then, online, the network controller receives a specific traffic matrix as input and solves the routing problem restricted to this precomputed set of paths to determine quickly the optimal routing.

A. Related work

Perhaps the simplest way to jointly route multiple demands under capacity constraints is to use successive shortest paths on a graph with updated remaining link capacities. This greedy procedure quickly leads to suboptimal configurations, so previous SDN techniques use a central multicommodity flow optimization to globally reconfigure the network [4], [5]. Since the multicommodity flow solver is typically slow, the network remains suboptimally configured for some time. In the special case that the demands are known a priori, we can find the optimal routing cost by creating a *Bandwidth Calendar* and solving a time-expanded version of the mincost multicommodity flow problem [6], [7]. The optimization approaches above require as input the traffic matrix and assume that it remains fixed.

Related to our pathbook design problem is the prior art on robust optimization. Past works on QoS routing have focused on *robust network design*, where the goal is to design network capacities to withstand a variety of traffic matrices [8], and *robust routing* where the goal is to find the best path allocation to each commodity to withstand unpredictable future variations of the traffic matrix [9], [10], [11]. Our work is a generalization

of this concept since we preselect a *set* of paths to optimize performance under uncertainty.

A line of work has focused on preselecting paths to accelerate the difficult problem of constrained shortest paths; [12] computes the *all hops optimal path*, i.e., the best path of length h, for $h = 1, 2, \ldots$, while [13] uses caching of good paths. These schemes focus on QoS constraints but do not take into account online congestion in an optimal manner, which is the focus of this paper. In section V-C we adapt our framework to include QoS constraints.

The most closely related line of work is *oblivious routing* [14], [15], [16], [17], which aims at computing offline sets of paths and load-balancing weights for every possible commodity, so that arriving demands can be routed without even looking at the state of the network. To make the oblivious routing more reactive to network state, and hence more efficient, statistics of the typical traffic matrices are incorporated in [18], [19]. Also, [20], [21] depart from the original oblivious approach and use simple congestion metrics on the pre-selected paths to take their online decisions: this is called *semi-oblivious* routing. *We extend the idea of oblivious routing to the problem of minimizing routing cost, offering in this way a novel framework for traffic engineering.*

Finally, we point out that the pathbook design problem is a generalization of the $k-$ splittable multicommodity flow problem [22], where each commodity has to be routed over at most k paths. The $k-$ splittable multicommodity max flow problem is NP-hard in general [22]; [23] gives approximation results for different cases. The min cost $k-$ splittable multicommodity flow is introduced in [24], where a heuristic with no approximation guarantees is presented. The latter problem can be seen as a special case of the pathbook design problem, for only one traffic matrix.

B. Our contribution

- We formulate the *pathbook design problem* for selecting a subset of paths per commodity to statistically minimize routing cost for a distribution of traffic matrices. The pathbook design problem is shown to be NP-hard and hard to approximate, and hence it is a challenging problem.
- We provide a subgradient method for optimally solving the convex pathbooks, i.e. the relaxation of the pathbook design problem. The method is based on identifying a subgradient vector as a function of easy-to-solve linear subproblems. We show that the SPDA converges logarithmically to an optimal convex pathbook. This convex pathbook is then rounded to find an integral pathbook for the original problem.
- When the paths are too many to enumerate, the subgradient vector is of very high dimensions, and hence impractical. In this case, we propose a coordinate-descent method based on Gauss-Southwell rule. Although the method has not guarantees for non-smooth constraint sets, it is shown to perform well in our test data, discovering quickly useful paths.

We show by experimentation on the GEANT network that we can efficiently learn the statistics over a week, design a small pathbook, and then solve the min cost MCF problem, restricted to these pre-selected paths, for each traffic matrix at near-optimal cost and much faster than solving the unrestricted min cost MCF.

II. Introduction to Minimum Cost Routing

Our network is an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \{c_e\}, \{b_e\})$, where $\mathcal{N}$ is the set of nodes, $\mathcal{E}$ the set of links, and each link e has an available bandwidth capacity b_e and a cost c_e per unit of bandwidth usage. The k^{th} source-destination pair corresponds to commodity k which has a bandwidth requirement of T_k, and we denote by $\boldsymbol{T}$ the $N \times N$ traffic matrix.[1]

Next we provide the path formulation of the well-known *minimum cost multicommodity flow* (MCF) problem, commonly used to optimize routing for a traffic matrix $\boldsymbol{T}$. Let $\mathcal{P}_k$ denote the set of all paths of $\mathcal{G}$ connecting the source of k to its destination. We introduce flow variables $\boldsymbol{f}$, where f_{pk} denotes the fraction of T_k that flows over path $p \in \mathcal{P}_k$. To route the entire commodity we require:

$$\sum_{p \in \mathcal{P}_k} f_{pk} = 1, \ \forall k, \quad \text{(path split)}. \tag{1}$$

Additionally, the total flow crossing a link must not exceed its capacity:

$$\sum_k T_k \sum_{p \in \mathcal{P}_k:\, e \in p} f_{pk} \leq b_e, \ \forall e, \quad \text{(capacity const.)}. \tag{2}$$

A flow $\boldsymbol{f} \geq 0$ satisfying (1)-(2) is said to be a *feasible multicommodity flow*. The MCF problem is then to find a feasible multicommodity flow of minimum cost:

$$\min_{\boldsymbol{f} \geq 0, (1)\text{-}(2)} \sum_k T_k \sum_{p \in \mathcal{P}_k} f_{pk} C_p, \tag{3}$$

where C_p is the cost of path p, calculated as $C_p = \sum_{e \in p} c_e$.

A. Why MCF is important?

The MCF problem inherently captures the complexity of simultaneously routing multiple commodities under the capacity constraints (2), where the latter couple together the routing allocations of different commodities. Since MCF searches for a feasible flow that economizes the resource usage, for a traffic matrix with low congestion the commodities are routed over shortest paths, but for a traffic matrix with high congestion certain commodities are rerouted over more expensive paths to avoid congested links. The linear costs are general enough to model (i) bandwidth usage costs, (ii) network congestion, (iii) link latency, or more generally (iv) routing prices that implement a specific policy. We note that our work extends almost immediately to convex objectives, *less the issue of computing the subgradient of the expectation, which depending on the chosen objective may be difficult or not*. However, this paper focuses on linear costs which are commonly used in practice.

[1] Our static bandwidth model directly applies to MPLS tunnel reservations and IP aggregated traffic [25], layer-2 WDM networks [26], and network slicing [27]; in these applications it is customary to reserve bandwidth by slightly overprovisioning the peak hour demand. Additionally, our model can be used as a snapshot model for a time-varying online routing problem.

Practical traffic engineering typically requires generalizations of MCF, such as (i) single-path (integral) routing, or equal-cost multipath routing, (ii) survivable routing with backup paths, (iii) Quality of Service constraints in the form of linear constraints on additive link latencies, (iv) distributed solutions of MCF across multiple controllers, (v) online MCF adapting to time-varying traffic matrices, and combinations. These generalizations can be solved by extending MCF algorithms in certain ways. Therefore, *to achieve efficient traffic engineering in future virtual networks, it is of paramount importance to be able to quickly solve MCF for large networks.*

B. Existing solutions of MCF

MCF is a linear program, and hence it can be solved in finite steps polynomial to the input by the ellipsoid algorithm. In practice, the best known technique is the *Column Generation* (CG), typically yielding a $\times 10$ performance in terms of runtime with respect to the common simplex method [28]. In CG, we exploit the linear programming property of the existence of a *basic solution* to maintain a small set of non-zero columns, here a column corresponds to a path and a basic solution must use $K + E$ paths. At each iteration, we solve the restricted linear program with the active paths as variables. Then, using properties from Lagrangian theory and strong duality, we can determine which path is added/removed at each iteration. When CG terminates it provides an optimal and sparse solution of MCF (3), as well as the optimal dual variable for each constraint (1)-(2) [28]; we will use this property in order to compute the subgradient and its highest change coordinate in Sections IV and V, respectively.

The total number of optimization variables of (3) is $K\left|\cup_{k\in K}\mathcal{P}_k\right|$, where $\cup_{k\in K}\mathcal{P}_k$ is the set of all routable paths. In most graphs if we allow all possible paths to be routable, $\left|\cup_{k\in K}\mathcal{P}_k\right|$ becomes an immense number. On the other hand, the non-zero variables used in CG are at most $K + E$ (one variable for each constraint), making the solution of large MCF instances manageable, and CG the best known algorithm for solving this problem. Nevertheless, our optimized c++ implementation of CG running on a state-of-the-art server needs approximately 10 minutes to solve congested MCF instances for networks of 500 nodes and 10k demands. In the future network architectures, the controllers will need to orchestrate large networks with complicated constraints, and the orchestration should occur in a few seconds. Therefore, we are motivated to accelerate the solution of MCF.

C. Accelerating MCF with pathbooks

Definition 1 (Pathbook)**.** *Let $\tilde{\mathcal{P}}_k \subset \mathcal{P}_k$ be a (small) subset of the commodity k paths. The set $\cup_{k\in K}\tilde{\mathcal{P}}_k$ is called pathbook.*

By replacing in (1)-(3) $\mathcal{P}_k$ with $\tilde{\mathcal{P}}_k$ for all k, we may restrict MCF to a pathbook. This is useful because if $\tilde{\mathcal{P}}_k$s are small, the restricted problem can be solved much faster; we can directly use the paths in the pathbook as the only active columns in CG and solve the problem in one iteration. Essentially, the precomputed pathbook replaces the effort of CG to search for good paths. Our experiments explained in Sections VI show a 250-500 and 20-40 times acceleration of the MCF solution with and without QoS constraints, respectively, yielding the desired result in tens of milliseconds, see Section VI and Table I.

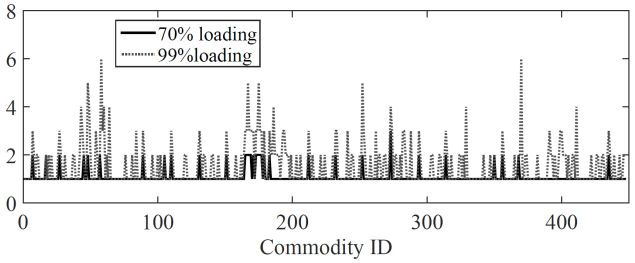

Fig. 1: Total number of paths used in the optimal solutions in 3 months of GEANT data (11460 traffic matrices with 15 minutes granularity) for mincost routing under medium (70%) and heavy (99%) loading.

Therefore, this paper proposes to accelerate MCF solution by restricting feasible paths to a precomputed pathbook. Note that the restriction can lead to lack of feasibility or increased cost. For illustration, consider the simple case where the pathbook contains only the shortest paths. As explained above, this will lead to very fast optimal solutions at low congestion, but also to infeasibility at high congestion, since the solver will be unable to use reroutes to avoid congested links. Therefore an important question is *how can we restrict MCF to a small pathbook without loss of optimality?*

D. Useful observations on GEANT data

To amplify our hopes for finding a small pathbook we make the following practical observation: *a small set of paths is sufficient to provide a joint solution to multiple traffic matrices representing realistic data*. Indeed, we experimented with dynamic traffic matrices available at [29] for the GEANT network. We selected the same capacity per link so that the load is 70% and 99% of the maximum feasible, and computed the optimal MCF solution for each of the available 11460 matrices. The union of paths used in all optimal solutions is shown in fig. 1, where we observe that each commodity only utilizes a very small number of paths during the entire 4 month duration. This motivates a novel approach to QoS routing: *instead of repeatedly computing shortest paths and multicommodity flows, which is time consuming, we propose a mechanism where (offline) a pathbook is identified that performs well for a distribution of traffic matrices, and then (online) the controller solves a restricted linear program to derive the optimal routing for the new traffic matrix query.* This paper addresses a main challenge that arises in this scheme: how to design the *optimal* pathbook of a constrained size.

III. The Pathbook Design Problem

In this section we design a pathbook that optimizes the statistical cost under a given distribution of traffic matrices.

A. Handling uncertain traffic matrices

The design of a pathbook is a slow process that occurs offline, before the actual MCF query. In general the traffic

matrix is unknown and time-varying [30]. Hence, the pathbook design is done by assuming that the queried traffic matrix is random described by a probability distribution $F(x)$ over $\mathbb{R}_+^K$, such that

$$F(\boldsymbol{x}) = \mathbb{P}(T_1 \leq x_1, \ldots, T_K \leq x_K)\,.$$

The distribution F can be estimated in an SDN application efficiently using monitored data [30]. For practical purposes, F is typically available in the form of a histogram of quantized traffic matrices. To simplify exposition below, we assume a finite collection of M traffic matrices $\boldsymbol{T}^1, \ldots, \boldsymbol{T}^M$, indexed by m, and an associated probability distribution $[p_m]$. However, we remark that our approach also applies to the continuous case using sampling techniques.

B. Problem formulation

We introduce binary variables P_{pk}, such that $P_{pk} = 1$ if and only if $p \in \tilde{\mathcal{P}}_k$. Restricting to a pathbook $\tilde{\mathcal{P}}_k$ will hereinafter be described by the vector $\boldsymbol{P} = [P_{pk}]_{p\in\mathcal{P}_{\text{all}}, k\in K}$. With this notation, the minimum routing cost for traffic matrix $\boldsymbol{T}^m$ is given by:

$$C^m(\boldsymbol{P}) = \min_{\boldsymbol{f}\geq 0} \sum_k T_k^m \sum_p f_{pk} C_p \tag{4}$$

$$\text{s.t.} \sum_k T_k^m \sum_{p:\, e\in p} f_{pk} \leq b_e, \quad \forall e \in E, \tag{5}$$

$$\sum_p f_{pk} = 1, \qquad \forall k \in K, \tag{6}$$

$$f_{pk} \leq P_{pk}, \qquad \forall k \in K,\ p \in \mathcal{P}_{\text{all}} \tag{7}$$

where we have rewritten (3) for $\boldsymbol{T}^m$, and additionally used (7) to restrict the paths to the pathbook $\boldsymbol{P}$. Since the traffic matrix is random, the total routing cost we are interested in optimizing will also be random. Thus, our objective is to minimize the expected cost $\mathbb{E}[C(\boldsymbol{P})] = \sum_m p_m C^m(\boldsymbol{P})$ incurred by using a pathbook $\boldsymbol{P}$ of limited size.

Pathbook design problem:

$$\min_{P_{pk}\in\{0,1\}^{\mathcal{P}_{\text{all}}\times K}} \mathbb{E}\left[C(\boldsymbol{P})\right] \tag{8}$$

$$\text{s.t.} \sum_p P_{pk} \leq P_{\max}^k, \qquad \forall k \in K. \tag{9}$$

where $P_{\max}^k$ is a commodity-dependent pathbook size parameter. Hereinafter, we will focus on studying feasible instances of problem (8).

Because of its combinatorial nature, problem (8) is hard to solve as the next theorem establishes.

Theorem 2 (Complexity)**.** *The pathbook design problem* (8) *is NP-hard, and hard to approximate.*

Proof: The first part is proven by reduction to the k-splittable flow, where we are asked to find a flow over at most k paths that minimizes the cost. The k-splittable flow is known to be NP-hard and hard to approximate [23], [22]. Indeed, any instance of the k-splittable flow problem can be solved by an algorithm that produces a solution to (8), by creating a corresponding instance with $M = 1$, $K = 1$, and $P_{\max}^1 = k$. ∎

We additionally note that the pathbook design problem is a generalization of the the integral multicommodity flow; it can be seen by selecting instances with $M = 1$, and $P_{\max}^k = 1, \ \ \forall k$. The decision version of the integral multicommodity flow is known to be strongly NP-complete [31]. Furthermore, the networks we are interested in are typically large, hence the pathbook design problem is computationally very challenging.

In the remaining of the paper we focus on solving the fractional relaxation problem and getting an integral solution through it.

IV. Convex pathbooks

In this section w6e focus on the fractional relaxation of the pathbook design problem (8), called *convex pathbooks*. The fractional relaxation can provide a lower bound on the optimal cost of (8), but it can also be used as a heuristic approximation.

We first relax the path selection variables P_{pk} allowing them to take real values, i.e., $\boldsymbol{P} \in [0,1]^{\mathcal{P}_{\text{all}}\times K}$ instead of $\{0,1\}^{\mathcal{P}_{\text{all}}\times K}$. In this context, $P_{pk} = 0.5$ means that the path p can be used by commodity k to route up to $0.5T_k^m$ traffic for each realization m.

Lemma 3. *The function $\boldsymbol{P} \in [0,1]^{\mathcal{P}_{all}\times K} \mapsto \mathbb{E}[C^m(\boldsymbol{P})]$ is convex.*

Proof: The proof is given in Appendix A. ∎

Convex pathbooks problem:

$$\min_{P_{pk}\in[0,1]^{\mathcal{P}_{\text{all}}\times K}} \mathbb{E}\left[C^m(\boldsymbol{P})\right] \tag{10}$$

$$\text{s.t.} \sum_p P_{pk} \leq P_{\max}^k, \qquad \forall k \in K. \tag{11}$$

The convex pathbooks can be solved to optimality using an iterative algorithm. We note that the function $\mathbb{E}[C^m(\boldsymbol{P})]$ is linear by parts, and not everywhere differentiable. Hence, we will design an iterative algorithm based on its subgradients [32].

In order to compute the subgradient of $\mathbb{E}[C^m(\boldsymbol{P})]$ at a given pathbook $\boldsymbol{P}$, we focus first on the *subproblem* (4) for a single traffic matrix m. Let us introduce the Lagrange multipliers (α_e), (β_k) and (γ_{pk}) associated to the constraints (5)-(7) of problem (4), as well as the Lagrangian function L_P^m at pathbook $\boldsymbol{P}$:

$$L_{\boldsymbol{P}}^m(\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \sum_{pk} f_{pk} T_k C_p + \sum_{pk} \gamma_{pk}\left(f_{pk} - P_{pk}\right)$$

$$+ \sum_k \beta_k \left(1 - \sum_p f_{pk}\right) + \sum_e \alpha_e \left(\sum_{p,k:\, e\in p} f_{pk} T_k - b_e\right).$$

Above, γ_{pk} is the price of using a path more than it is prescribed by the pathbook, β_k is the price of not routing the entire commodity, and α_e is the price of exceeding the capacity of a link. From strong duality of convex programming, the value $C^m(P)$ of problem (4) satisfies

$$C^m(\boldsymbol{P}) = \max_{\boldsymbol{\alpha},\boldsymbol{\gamma}\geq 0, \boldsymbol{\beta}\in\mathbb{R}} \min_{\boldsymbol{f}\geq 0} L_{\boldsymbol{P}}^m(\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \tag{12}$$

$$= L_{\boldsymbol{P}}^m\left(\boldsymbol{f}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*\right),$$

where $\boldsymbol{f}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*$ denote optimal primal and dual values of (4) for a specific m and $\boldsymbol{P}$. This observation leads to the computation of the subgradient of $C^m(\boldsymbol{P})$:

Lemma 4. *For traffic matrix m and pathbook $\boldsymbol{P} \in [0,1]^K$, the vector $-\boldsymbol{\gamma}^* = -\boldsymbol{\gamma}^*(m, \boldsymbol{P})$ is a subgradient of C^m at $\boldsymbol{P}$, i.e., for any $\boldsymbol{P}' \in [0,1]^K$ we have*

$$C^m(\boldsymbol{P}') \geq C^m(\boldsymbol{P}) - \boldsymbol{\gamma}^{*T}(\boldsymbol{P}' - \boldsymbol{P}). \tag{13}$$

Proof: The proof is given in Appendix B. ∎

Then the subgradient vector of the objective function $\mathbb{E}[C^m(\boldsymbol{P})]$ at $\boldsymbol{P}$ is given by $-\sum_m p_m \boldsymbol{\gamma}^*(m, \boldsymbol{P})$. Next, we explain how to compute $\boldsymbol{\gamma}^*(m, \boldsymbol{P})$. As explained in Section II, linear programs such as (4) are efficiently solved using the Column Generation method [28], which also provides the optimal Lagrange multipliers $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$, and the values of $\gamma^*_{pk}(m, \boldsymbol{P})$ for $P_{pk} > 0$. However the values of $\gamma^*_{pk}(m, \boldsymbol{P})$ for $P_{pk} = 0$ are not provided directly as these paths are not included as active columns. To compute the full subgradient $-\boldsymbol{\gamma}^*(m, \boldsymbol{P})$ we need the following result:

Lemma 5. *For a given traffic matrix $\boldsymbol{T}^m$ and pathbook $\boldsymbol{P}$, let $\boldsymbol{\alpha}^*$, $\boldsymbol{\beta}^*$ be optimal values for the Lagrangian multipliers in* (12) *and let $\alpha^*_p = \sum_{e \in p} \alpha^*_e$. Then,*

$$\gamma^*_{pk}(m, \boldsymbol{P}) = \max\{0, \beta^*_k - T^m_k(C_p + \alpha^*_p)\}. \tag{14}$$

Proof: The proof is given in Appendix C. ∎

We note that all the dual variables above depend on $(m, \boldsymbol{P})$ as the discussion is made for a specific subproblem (4), however for simplicity, we explicitly denote the relation only for variables γ_{pk}. Additionally, since the subproblems (4) must be solved for different traffic matrices $\boldsymbol{T}^m$, they are independent and can be solved in parallel, exploiting modern multi-core processors.

Based on the discussion above, we propose the following subgradient pathbook design algorithm:

Subgradient pathbook design algorithm (SPDA)

Input: Network $\mathcal{G}$; Collection of traffic matrices $(\boldsymbol{T}^m)$; probability distribution $[p_m]$; pathbook size constraints $(P^k_{\max})$.

Initialization: add to $\boldsymbol{P}$ the shortest path for each commodity.

Convex pathbook computation: Until convergence of $\boldsymbol{P}$, iterate the following:

Computation of dual variables for each traffic matrix: For each m, solve the MCF problem (4) restricted to the current pathbook $\boldsymbol{P}$. Obtain the values of the optimal Lagrange multipliers $\boldsymbol{\alpha}^*(m, \boldsymbol{P})$ and $\boldsymbol{\beta}^*(m, \boldsymbol{P})$.

Subgradient pathbook update: For each path p, compute $\mathbb{E}\left[\gamma^*_{pk}(m, \boldsymbol{P})\right] = \sum_p p_m \gamma^*_{pk}(m, \boldsymbol{P})$ using (14); update P_{pk} according to

$$P_{pk} \leftarrow P_{pk} + s(t)\mathbb{E}\left[\gamma^*_{pk}(m, \boldsymbol{P})\right],$$

where $s(t)$ is a step-size at iteration t defined below. Project $\boldsymbol{P}$ on the feasible space: (maybe here consider an orthogonal projection on simplex)

$$P_{pk} \leftarrow \max\left\{0, P_{pk} / \sum_p P_{pk}\right\}.$$

Rounding: For each commodity k, sort the paths p by decreasing order of P_{pk}. Keep only the first $P^k_{\max}$ paths.

The performance of SPDA algorithm is provided by the next theorem.

Theorem 6 (Performance of SPDA)**.** *Choose step sizes $s(t) = 1/\sqrt{t}$, SPDA converges to an optimal convex pathbook solution of* (10)*.*

Proof: The Theorem follows from Lemmas 3-5 and classical results on subgradient algorithms (see [32]). This is our main result, consider adding a proof here. ∎

V. Gauss-Southwell pathbooks

The dimension of the pathbook vector $\boldsymbol{P}$ is equal to the total number of paths in the graph, which grows superexponentially with the size for most types of graphs. Hence, in large networks the computation of the subgradient vector at each iteration of SPDA may be expensive. Additionally, it is expected that most paths are not actually used by an optimal, or even near-optimal, solution of (8). For this reason, in this section we develop a *coordinate subgradient method* which does not need to compute the entire subgradient vector at each iteration, but instead is based on changing the pathbook vector $\boldsymbol{P}$ along a single dimension, which corresponds to the increase/decrease of a pathbook variable P_{pk}.

Our constraint set described by (4)-(7) and (9) is non-smooth, and coordinate methods do not necessarily provide descent directions if the set is not smooth [33]. Hence our proposed method will not descend monotonically. Nevertheless, our experiments show a satisfactory performance, whereby the distance between the current best solution found and the neighborhood of the optimal solution gradually decreases.

The classical coordinate descent method iterates over all coordinates in a round robin fashion. Prior work shows that the optimal convergence of coordinate descent methods can also be attained by visiting each coordinate at random [34]. Here, we propose a less known rule called *Gauss-Southwell* [], which moves along the direction of *the steepest subgradient coordinate*. Intuitively, choosing the coordinate with the greatest value is a wiser move as it uses more information from the subgradient vector, especially so if most of the coordinates are zero. Prior work [35] analyzes the Gauss-Southwell coordinate descent rule, showing that theoretically it achieves the optimal convergence rate but additionally it practically outperforms the randomized rule. The reason for choosing Gauss-Southwell rule is more than just the good convergence properties. We show below that Gauss-Southwell rule corresponds to a shortest path problem.

A. Approximately computing the Gauss-Southwell rule

Our algorithm increases the pathbook along the direction of the path p^* that corresponds to the largest subgradient coordinate $\mathbb{E}[\gamma^*_{pk}]$. Hence, the algorithm will increase the corresponding value P_{pk} by a step-size parameter $s(t)$ at iteration t, and rescale $\boldsymbol{P}$ to satisfy the constraints (7). It remains to explain how to find the path p^*.

Using Lemma 5 and writing $\widetilde{\beta}^* = \beta^*_k/T^m_k$, we observe that the Gauss-Southwell rule can be computed by solving the following path problem:

$$p^* \in \arg\min \mathbb{E}\Big[\min\Big\{0, C_p + \alpha^*_p - \widetilde{\beta}^*\Big\}\Big]. \qquad (15)$$

Therefore, we are searching for a shortest path according to the modified cost $\mathbb{E}\Big[\min\Big\{0, C_p + \alpha^*_p - \widetilde{\beta}^*\Big\}\Big]$. Searching for shortest path on cost-modified graphs is often encountered when solving multicommodity flow problems [28], where however the path cost modifications are equivalent to link cost modifications. Here, due to the expectation of a minimum, the cost is not additive in the edges used in the path. Therefore, it cannot be captured through link costs, and so the path p^* cannot be obtained by classical shortest path algorithms.

Instead, we can equivalently cast problem (15) as a constrained submodular minimization. We define the function $\phi^m_{\boldsymbol{P}} : 2^{\mathcal{E}} \to \mathbb{R}$,

$$\phi^m_{\boldsymbol{P}}(A) = \min\left\{0, \sum_{e\in A}\Big(C_e + \alpha^*_e(m, \boldsymbol{P}) - \widetilde{\beta}^*\Big)\right\} \qquad (16)$$

for $A \subseteq \mathcal{E}$. It is easy to check that the functions $\phi^m_{\boldsymbol{P}}$ are submodular [36], i.e., for any two sets $A, B \subseteq \mathcal{E}$ we have $\phi^m_{\boldsymbol{P}}(A \cup B) + \phi^m_{\boldsymbol{P}}(A \cap B) \leq \phi^m_{\boldsymbol{P}}(A) + \phi^m_{\boldsymbol{P}}(B)$. It is then immediate that the function $\mathbb{E}[\phi^m_{\boldsymbol{P}}]$ is also submodular. Ultimately, the Gauss-Southwell rule can be computed by finding a set A minimizing $\mathbb{E}[\phi^m_{\boldsymbol{P}}]$ under the constraint that A is the link-set of a valid path for commodity k:

Minimum submodular cost path problem:

$$\min_{A\subseteq\mathcal{E}} \mathbb{E}[\phi^m_{\boldsymbol{P}}(A)] \qquad (17)$$
$$\text{s.t. } A \text{ is a path of } k.$$

Problem (17) is a special case of constrained submodular minimization known to be NP-hard and with a worst-case approximation ratio $\Omega(|\mathcal{N}|^{2/3})$ [37]. However, prior work reports encouraging practical performance obtained with convex relaxation via the Lovasz extension of the submodular function [38]. We next describe an adapted approximate algorithm for our specific problem.

Approximate submodular-cost shortest path

General idea: Consider the Lovász extension $\mathbb{E}\Big[\widetilde{\phi}^m_{\boldsymbol{P}}\Big]$ of $\mathbb{E}[\phi^m_{\boldsymbol{P}}]$, which is a convex function over $[0,1]^{|\mathcal{E}|}$ and agrees with $\mathbb{E}[\phi^m_{\boldsymbol{P}}]$ on all the extreme points $\{0,1\}^{|\mathcal{E}|}$.
1) Find an optimizer $\boldsymbol{f} \in [0,1]^{|\mathcal{E}|}$ of $\mathbb{E}\Big[\widetilde{\phi}^m_{\boldsymbol{P}}\Big]$ by using projected subgradient method.
2) Round the solution $\boldsymbol{f}$ into a path p.

We now detail steps 1-2:

1) Computation of a subgradient of $\mathbb{E}\Big[\widetilde{\phi}^m_{\boldsymbol{P}}\Big]$ at $\boldsymbol{f}$:

1) Sort the coordinates of $\boldsymbol{f}$ in decreasing order; let $\sigma : \{1,\dots,|\mathcal{E}|\} \to \mathcal{E}$ be the permutation such that $f_{\sigma(1)} \geq f_{\sigma(2)} \dots \geq f_{\sigma(|\mathcal{E}|)}$.
2) Define the sets $A_i = \{\sigma(j) : j \leq i\}$, for all $i \in \{0,\dots,|\mathcal{E}|\}$.
3) For all i, set the value of the coordinate $\sigma(i)$ of the subgradient $\boldsymbol{y}$ of $\mathbb{E}\Big[\widetilde{\phi}^m_{\boldsymbol{P}}\Big]$ at $\boldsymbol{f}$ as follows:

$$y_{\sigma(i)} = \mathbb{E}[\phi^m_{\boldsymbol{P}}(A_i)] - \mathbb{E}[\phi^m_{\boldsymbol{P}}(A_{i-1})]\,.$$

2) Rounding of the fractional flow $\boldsymbol{f} \in [0,1]^{|\mathcal{E}|}$ into a path p:

1) Sort the coordinates of $\boldsymbol{f}$ in decreasing order, and define the permutation σ and the sets (A_i) as above.
2) Find the smallest $i \in \{1,\dots,|\mathcal{E}|\}$ such that A_i contains a valid path p for commodity k, and output p.

We propose the Gauss-Southwell pathbook design algorithm GSPD, in which we replace the subgradient update in the SPDA by a small step in the direction given by an approximate submodular-cost shortest path.

B. Validation of GSPD

To prove the concept of our Gauss-Southwell heuristic we conducted experiments on a random directed network of 50 nodes and 362 links shown in fig.2 (left). On this network we generate 100 different traffic matrices, where 380 commodities are active (sources-destinations shown with blue). We scale each traffic matrix m so that it induces on the network a load ρ very close to 95%, i.e., the MCF for $T^m/0.95$ is feasible but saturates the network.

Solving the traffic matrices optimally, we observe that on average the MCF solutions with all paths use 22.01 paths per commodity and admit 44.25 units of traffic for a cost of 101.81. Comparatively, note that the shortest paths alone can support a traffic of 15.88 at a cost of 10.5. Figure 2 shows the performance of the pathbooks obtained using the GSPD for various size limits $P^k_{\max}$. The cost ratio shown is the average cost of pathbook-limited routing divided by the optimal for the demand that our scheme successfully admits.

We observe that the obtained pathbooks allow us to route all the traffic as soon as 6 paths per commodity are used. However, the limited pathbook induces an extra cost for routing. Yet, this cost is significant only when the pathbook size limit is barely sufficient to sustain the traffic, and it becomes negligible as we increase slightly the pathbook size. In this case, a pathbook of 10 paths per commodity is sufficient to yield very close to optimal behavior.

C. Dealing with constrained paths and failure scenarios

The proposed framework is suitable to deal with more complex traffic engineering rules where each commodity has a set of constraints on the paths it can be routed over. For example, in QoS routing applications it is customary to incorporate latency constraints or to search for pairs of paths with

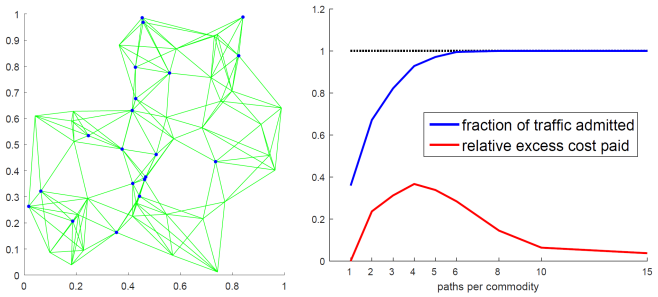

Fig. 2: Performance of pathbooks generated by the GSPD on a random network of 50 nodes and 362 links.

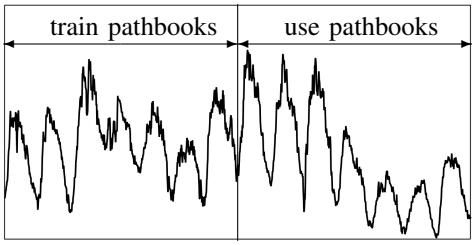

Fig. 3: Our proposed traffic engineering methodology: Collected data are used to train the pathbook design in an offline manner while pathbooks are used to restrict the MCF problem in real time.

certain disjointness properties to anticipate against failures of some network element. The pathbook design algorithms need only be modified in the part which updates the current path-selection variables: SPDA will only keep the paths which satisfy the constraints, and DFPD will only be modified in the submodular cost shortest path procedure, where the rounding part will select a valid constrained path. Finding constrained paths online can be a hard task for some constraints (it is already NP-complete to find a shortest path with latency constraints [39]). Therefore, it is an important advantage of our approach that these operations become entirely offline.

In addition, robustness to failures or to different network states can be incorporated in our framework. Indeed, this amounts to considering different network states G^i in addition to different traffic matrices $\boldsymbol{T}^m$. Then, we would solve the slave problem (4) for different networks and different traffic matrices. As an example, to create pathbooks resilient to (distributed) Denial-of-Service attacks, we can incorporate traffic matrices with additional artificial large demands directed at particular nodes in the network. This would create congestion in some regions of the network, and the pathbooks obtained would offer some paths avoiding these regions.

VI. A SCALABLE TRAFFIC ENGINEERING APPROACH FOR SDN

In this Section, we use GSPD together with min cost MCF to obtain a novel traffic engineering methodology for dynamic traffic matrices. The novel traffic engineering proposed works as follows. Traffic statistics are continuously collected in order to obtain a distribution of traffic matrices. A recent summary is fed into the pathbook design problem which repeatedly provides pathbook updates at a slow time-scale. Then, when the traffic matrix changes, a min cost MCF is executed, restricted to only use paths from the most recent pathbook.

We use data from the SNDlib library for robust network design [29]. The available data include 11460 traffic matrices for a duration of 4 months on the GEANT network [40]. The link capacities are not specified on the dataset.

We focus on two consecutive weeks, corresponding to traffic matrices indexed from 8603 to 9948, shown in figure 3. We assign equal latency demands for each commodity and a random capacity and delay to each link. Then, we scale the traffic matrices such that we have two operating regimes: (i) a "medium load" regime, where all traffic matrices, if rescaled by $10/7$ are feasible when the min cost MCF is solved by a Column Generation algorithm which uses the LARAC algorithm [41] to choose new columns (paths) that satisfy the latency constraints of each commodity and (ii) a "high load" regime, where the aforementioned algorithm admits 95% of all traffic.

As seen in Fig. 3, we use the first week of data to train our pathbooks; we apply GSPD on the data for different pathbook sizes, as explained in V-C in order to ensure that only paths that satisfy the latency constraints are kept. For the next week of data, we use solve a min cost MCF problem that is restricted to use only paths on the derived pathbooks for each traffic matrix.

We compare with the algorithm that, for each traffic matrix, solves the min cost MCF using Column Generation with LARAC to find paths (columns) with minimum augmented cost and satisfying the latency constraints at each iteration (which was also used to scale the traffic matrices as explained before). We term this algorithm "full MCF". Results are shown in Fig. 4. We can see from Fig. 4a that a pathbook size of 10 paths per commodity is already enough to accept almost all traffic that full MCF accepts. In addition, as Fig. 4b shows, there is almost no difference in the cost by restricting to a pathbook of 20 paths (the pathbook-restricted MCF has lower cost than the full MCF at small pathbook sizes because a sizeable portion of the traffic is not admitted). Finally, restricting to a pathbook can significantly speed up the routing decision: Fig. 4c depicts the speedup gained by using a pathbook, i.e. the ratio of the runtime of the full MCF over the MCF constrained to the pathbook, and we can see that using pathbooks can speed up the routing decision by some hundreds of times. This is because the full MCF needs to search for delay-constrained paths for each CG iteration, which is a time-consuming operation, while in our approach good columns for the min cost MCF problem have already been selected offline, during the pathbook construction phase.

Overall, these results suggest that using pathbooks we can significantly speed up the routing process with essentially no loss of optimality: For 25 paths per commodity the proposed framework achieves almost identical performance with full MCF, but is 500 and 250 times faster in high and medium loading, respectively.

When the commodities do not have latency constraints, we observe a similar behavior with respect to a Column

Generation algorithm for the min cost MCF at every traffic matrix. Detailed runtimes for medium loaded networks are showin in Table I. In the case of no latency constraints, the pathbook MCF is one order of magnitude faster than the full MCF, because the CG uses unconstrained shortest paths, which are less time consuming that constrained ones.

Pathbook size	Pathbook MCF, latency	full MCF, latency	Pathbook MCF, no latency	full MCF, no latency
1	0.0187	15.2534	0.0148	1.3083
2	0.0297	15.2534	0.0281	1.3083
4	0.0381	15.2534	0.0449	1.3083
6	0.0423	15.2534	0.0489	1.3083
10	0.0507	15.2534	0.0576	1.3083
15	0.0587	15.2534	0.0594	1.3083
20	0.0897	15.2534	0.0581	1.3083
25	0.0917	15.2534	0.0616	1.3083

TABLE I: Average runtimes (in seconds) for medium load.

VII. Conclusions

We decomposed online minimum cost routing into (a) offline pathbook design, using past traffic data and (b) solving the MCF for the current traffic matrix, restricting it on the paths already present in the pathbook. We have proposed a pathbook generation method, based on the Gauss-Southwell rule and relaxations for the resulting minimum submodular shortest path problem at each iteration based on Lovasz extensions. Our experimental results on the GEANT network show that, with this decomposition we can get a solution of the min cost MCF problem one to two orders of magnitude faster than solving the MCF for each traffic matrix, with essentially no loss of optimality. Interesting could be results on the proposed pathbook design algorithms (and such results for the pathbook design problem in general) and studying the impact on traffic matrix prediction on the performance of pathbook-based traffic engineering.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] K. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent network updates," *ArXiV e-prints, abs/1609.02305*, 2016.

[3] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning," in *ACM IMW*, 2002.

[4] S. Paris, A. Destounis, L. Maggi, G. S. Paschos, and J. Leguay, "Controlling Flow Reconfigurations in SDN," in *Proc. of IEEE INFOCOM*, 2016.

[5] S. Brandt, K. T. Förster, and R. Wattenhofer, "On consistent migration of flows in sdns," in *IEEE INFOCOM*, 2016.

[6] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," *Proc. of ACM SIGCOMM*, 2014.

[7] L. Gkatzikis, S. Paris, I. Steiakogiannakis, and S. Chouvardas, "Bandwidth calendaring: Dynamic services scheduling over software defined networks," in *Proc. of ICC*, 2016.

[8] C. Chekuri, "Routing and network design with robustness to changing or uncertain traffic demands," *SIGACT News*, pp. 106–129, 2007.

[9] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: Traffic engineering in dynamic networks," *Proc. of ACM SIGCOMM*, 2006.

[10] V. Tabatabaee, A. Kashyap, B. Bhattacharjee, R. J. La, and M. A. Shayman, "Robust routing with unknown traffic matrices," in *Proc. of IEEE INFOCOM*, 2007.

[11] A. Gunnar and M. Johansson, "Robust load-balancing under statistical uncertainty: Models and polynomial-time algorithms," in *Proc. of NGI*, 2009.

[12] A. Orda and A. Sprintson, "Precomputation schemes for qos routing," *IEEE/ACM Transactions on Networking*, pp. 578–591, 2003.

[13] M. Peyravian and A. D. Kshemkalyani, "Network path caching," *Computer Communications*, pp. 605–614, 1997.

[14] H. Räcke, "Minimizing congestion in general networks," in *Proc. of IEEE FoCS*, 2002.

[15] S. Nelakuditi, Z.-L. Zhang, and D. H. C. Du, "On selection of candidate paths for proportional routing," *Computer Networks*, pp. 79–102, 2004.

[16] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal oblivious routing in polynomial time," in *Proc. of ACM STOC*, 2003.

[17] M. Englert and H. Räcke, "Oblivious routing for the lp-norm," in *Proc. of IEEE FOCS*, 2009.

[18] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs," in *Proc. of ACM SIGCOMM*, 2003.

[19] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Towsley, "On optimal routing with multiple traffic matrices," in *Proc. of IEEE INFOCOM*, 2005.

[20] M. Hajiaghayi, R. Kleinberg, and T. Leighton, "Semi-oblivious routing: lower bounds," in *Proc. of ACM SODA*, 2007.

[21] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, and R. Soulé, "Kulfi: Robust traffic engineering using semi-oblivious routing," *arXiv preprint:1603.01203*, 2016.

[22] G. Baier, E. Köhler, and M. Skutella, "The k-splittable flow problem," *Algorithmica*, vol. 42, no. 3, pp. 231–248, 2005.

[23] R. Koch, M. Skutella, and I. Spenke, "Approximation and complexity of k–splittable flows," in *Proc. of WAOA*, 2005.

[24] M. Gamst, P. N. Jensen, D. Pisinger, and C. Plum, "Two- and three-index formulations of the minimum cost multicommodity k-splittable flow problem," *European Journal of Operational Research*, vol. 202, no. 1, pp. 82 – 89, 2010.

[25] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys Tutorials*, vol. 10, no. 1, pp. 36–56, First 2008.

[26] R. Dutta and G. N. Rouskas, "Traffic grooming in wdm networks: past and future," *IEEE Network*, vol. 16, no. 6, pp. 46–56, Nov 2002.

[27] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 171–177.

[28] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Prentice Hall, 1993.

[29] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," *Networks*, pp. 276–286, 2010.

[30] Y. Gong, X. Wang, M. Malboubi, S. Wang, S. Xu, and C.-N. Chuah, "Towards accurate online traffic matrix estimation in software-defined networks," in *Proc. of SOSR*, 2015.

[31] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, Oct 1975, pp. 184–193.

[32] A. Nedi and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[33] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, Sep. 1999.

[34] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[35] J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, and H. Koepke, "Coordinate Descent Converges Faster with the Gauss-southwell Rule Than Random Selection," in *ICML*, 2015.

[36] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming The State of the Art*. Springer, 1983, pp. 235–257.

[37] R. Iyer, S. Jegelka, and J. Bilmes, "Monotone closure of relaxed constraints in submodular optimization: Connections between minimization and maximization: Extended version," in *Proc. of UAI*, 2014.

[38] ——, "Fast semidifferential-based submodular function optimization: Extended version," in *Proc. of ICML*, 2013.

[39] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1979.

[40] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, pp. 83–86, 2006.

[41] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in *IEEE INFOCOM*, 2001.

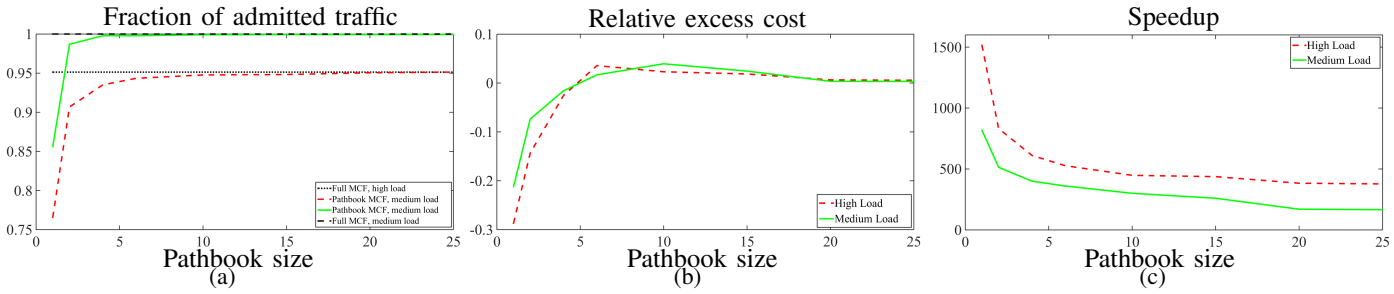

Fig. 4: Experiment results for the performance of pathbooks generated by the GSPD in min cost MCF with latency constraints on GEANT data: (a) Fractions of total admitted traffic (b) Relative excess cost paid with respect to the full min cost MCF (c) Speedup with respect to the full min cost MCF

Appendix A
Proof of Lemma 3

Let $\boldsymbol{P}(1)$ and $\boldsymbol{P}(2)$ be two pathbooks satisfying the constraint (9); and let $(\boldsymbol{f}^m(1))$ and $(\boldsymbol{f}^m(2))$ be optimal flow allocations attaining costs $(\boldsymbol{C}^m(\boldsymbol{P}(1)))$ and $(\boldsymbol{C}^m(\boldsymbol{P}(1)))$ respectively, while verifying the associated constraints (5)-(7). For any $\alpha \in [0,1]$, we have that $\alpha\boldsymbol{f}^m(1) + (1-\alpha)\boldsymbol{f}^m(2)$ also satisfy the linear constraints (5),(6), which do not depend on the path selection, and it also satisfies the coupling constraint (7) for the pathbook $\alpha\boldsymbol{P}(1) + (1-\alpha)\boldsymbol{P}(2)$. Hence, the flow allocation $\alpha\boldsymbol{f}^m(1) + (1-\alpha)\boldsymbol{f}^m(2)$ is feasible for the MCF problem associated to the traffic pattern m, and it attains the cost $\alpha C^m(\boldsymbol{P}(1)) + (1-\alpha)C^m(\boldsymbol{P}(2))$, which proves that $C^m(\alpha\boldsymbol{P}(1)+(1-\alpha)\boldsymbol{P}(2)) \leq \alpha C^m(\boldsymbol{P}(1))+(1-\alpha)C^m(\boldsymbol{P}(2))$. As C^m is a convex function of $\boldsymbol{P}$, so is the weighted sum $\mathbb{E}[C^m(\boldsymbol{P})]$.

Appendix B
Proof of Lemma 4

The Lagrangian function being multi-linear, we can consider the maximizations and minimizations in any order. We define

$$\widetilde{L}^m_{\boldsymbol{P}}(\boldsymbol{\gamma}) = \max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\in\mathbb{R}} \min_{\boldsymbol{f}\geq 0} L^m_{\boldsymbol{P}}(\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}).$$

We notice that the optimal values of $\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ in the definition of $\widetilde{L}^m_{\boldsymbol{P}}(\boldsymbol{\gamma})$ actually do not depend on $\boldsymbol{P}$, only the value of the function is affected by the term $-\sum_{p,k}\gamma_{pk}P_{pk}$. As a consequence, we have the following:

$$\begin{aligned}
C^m(\boldsymbol{P}) &= \max_{\boldsymbol{\gamma}\geq 0} \widetilde{L}^m_{\boldsymbol{P}}(\boldsymbol{\gamma}) \;=\; \widetilde{L}^m_{\boldsymbol{P}}\left(\boldsymbol{\gamma}^*(m,\boldsymbol{P})\right) \\
&= \widetilde{L}^m_{\boldsymbol{P}'}\left(\boldsymbol{\gamma}^*(m,\boldsymbol{P})\right) - \boldsymbol{\gamma}^*(m,\boldsymbol{P})^T(\boldsymbol{P}-\boldsymbol{P}') \\
&\leq \widetilde{L}^m_{\boldsymbol{P}'}\left(\boldsymbol{\gamma}^*(m,\boldsymbol{P}')\right) - \boldsymbol{\gamma}^*(m,\boldsymbol{P})^T(\boldsymbol{P}-\boldsymbol{P}') \\
&= C^m(\boldsymbol{P}') - \boldsymbol{\gamma}^*(m,\boldsymbol{P})^T(\boldsymbol{P}-\boldsymbol{P}').
\end{aligned}$$

Appendix C
Proof of Lemma 5

This result follows from the Karush-Kuhn-Tucker (KKT) conditions at $\boldsymbol{f}^*$, $\boldsymbol{\alpha}^*$, $\boldsymbol{\beta}^*$, $\boldsymbol{\gamma}^*$: for all p,k, we have

$$\begin{aligned}
&T_k(C_p + \alpha^*_p) - \beta^*_k + \gamma^*_{pk} = \xi_{pk} \geq 0, \\
&f^*_{pk} - P_{pk} \leq 0, \\
&f^*_{pk}\xi_{pk} = 0, \quad \gamma^*_{pk}\left(P_{pk} - f^*_{pk}\right) = 0,
\end{aligned}$$

where we omitted the KKT conditions which we will not use. From here there are three cases: if $P_{pk} = f^*_{pk} > 0$, then $\xi_{pk} = 0$ and the first line yields $\gamma^*_{pk} = \beta^*_k - T^m_k(C_p + \alpha^*_p)$; if $P_{pk} > f^*_{pk}$, then $\gamma^*_{pk} = 0$ immediately, and we note that $\beta^*_k - T^m_k(C_p + \alpha^*_p) = -\xi_{pk} \leq 0$. In the last case, $P_{pk} = f^*_{pk} = 0$, then any positive value can be chosen for γ^*_{pk}. Hence, the expression $\gamma^*_{pk} = \max\{0, \beta^*_k - T^m_k(C_p + \alpha^*_p)\}$ works for all the cases.