On Routing Algorithms for Open Marketplaces of Path Services

Shireesh Bhat, George N. Rouskas
Department of Computer Science, North Carolina State Universtiy, Raleigh, NC, USA

Abstract—**Open marketplaces of path services are the next step towards realizing "routing-as-a-service." Such marketplaces will enable users to select from a set of path services offered by multiple competing network providers so as to construct customized end-to-end paths for their applications. This is analogous to online travel marketplaces that allow users to explore travel options and book their travel. We review the requirements for path planners to assist users in stitching together available path services. We define the problem of finding multi-criteria time-constrained paths in this context, and present a dynamic programming algorithm that constructs Pareto-optimal paths.**

I. Introduction

Routing algorithms are at the core of network design and operation, and their functionality has evolved over the last sixty years from finding single shortest paths [1] to encompassing a wide range of considerations, including multiple paths [2], quality-of-service (QoS) constraints [3], and various modes of communication beyond point-to-point [4]. Nevertheless, for the most part, these routing algorithms have been designed for use by network providers/operators who have complete control over all aspects of the network. Users of the network typically have no visibility into the network topology or access to the routing function, and their traffic usually follows paths assigned by the network provider – although, using service level agreements (SLAs) they may request paths that satisfy certain properties.

Due to the evolving nature of network applications, requirements of routing functionality are also likely to evolve over time. However, at a time when network customers demand more flexibility in path selection, changes in routing-level components in the Internet require broad consensus among a diverse set of stakeholders and, hence, are increasingly difficult to implement. Accordingly, there has been some work in providing users with options over the routing path [5]–[7] in a manner that separates the data plane (the paths that packets follow) from the control plane (routing decisions) and allows the two to evolve separately.

A natural next step in realizing "routing-as-a-service" (RaaS) is the creation of open marketplaces of path services that will enable customers to select from a set of path services offered by multiple competing network providers, and *stitch* them together to construct customized end-to-end paths for their applications. This is analogous to online travel marketplaces, including Travelocity, Orbitz, and Expedia, among others, that allow users to explore travel options, make plans, and book their travel.

This work was supported by the National Science Foundation under Grant CNS-1111088

At a high level, an open marketplace of path services will consist of the following components [8]–[10]:

1) *Service advertisements:* the marketplace provides mechanisms for service providers to advertize their services and modify existing advertisements.
2) *Service repository:* we assume that the repository of path services is updated in real time, and that users and third parties may query the repository to retrieve path services that meet certain criteria.
3) *Path planner:* the planner takes as input user preferences and applies them to select and combine existing path services into a set of end-to-end paths from which the user will make a final selection.
4) *Contracts:* the markeplace has mechanisms to establish and enforce contracts between customers and providers to ensure that economic exchanges (e.g., payments) are related to operations within the network (e.g., access to the path services).

The Choicenet marketplace [8], [9] supports all the above features. Nevertheless, the primary goal of Choicenet has been to facilitate the offering of multiple service options and to enable the establishment of economic relationships between marketplace entities (i.e., users and physical or virtual infrastructure and service providers), and hence early prototypes [10] only included a rudimentary planner.

In this paper, we focus on the planning aspects of an open marketplace of path services, and in particular, on the requirements on routing algorithms that can be used to construct end-to-end paths by stitching together path segments advertized by multiple, distinct providers. While our work is inspired by the Choicenet project, we note that the service advertisement, path planning, and contract components of a marketplace are orthogonal in terms of functionality. Therefore, the path planning algorithms we present in this paper may be deployed within any marketplace with a clear separation between the data and control planes.

Following the introduction, we discuss the challenges of the path planning process, along with related work, in Section II. We present a model for the marketplace and the graph of path services that the planner maintains. We define a suite of problems related to finding multi-criteria time constrained paths, and develop appropriate algorithmic solutions, in Section IV. We present numerical results in Section V, and we conclude the paper in Section VI.

II. Path Planning

The planner allows the users to explore end-to-end path options for their communication needs. For simplicity, we assume that the path planning tool is implemented by the marketplace, but it may also be implemented by the user or offered as a service by a third party. Similar to online travel markeplaces, during the planning phase, customers have the opportunity to review the available options in terms of cost, quality, or other criteria. No contracts are established during this phase and no resources are committed by the network. Note also that for planning to work, providers must first advertize their path services in the marketplace in a way that allows the planner to determine which services can be composed together in a meaningful manner. While service advertisements, contract establishment, and resource provisioning are outside the scope of this work, we note that mechanisms exist for all three functions and have been implemented in an earlier Choicenet prototype [10].

The main problem involved in planning is to combine available services into end-to-end paths that meet user requirements. From an algorithmic point of view, path planning shares a number of challenges with online travel planning:

- *Large network topologies with parallel edges.* Just as the planner of a travel site takes into consideration flights from multiple airlines, many of which offer competing flights between the same pairs of cities, a path planner must consider advertisements from multiple providers, including virtual operators who may lease capacity from the same physical infrastructure. Consequently, the path planner takes as input a topology that is a superset of the topologies representing the networks of individual providers, and that is likely to include parallel edges between nodes for which there exist competing path services. Such a topology is expected to be much larger than each of its constituent individual provider topologies.
- *Support for in-advance reservations and time constraints.* Planners must allow users to reserve end-to-end paths during specific continuous time intervals in the future; this feature is analogous to booking a hotel for a set of consecutive days long before travel takes place. On the other hand, support for time constraints allows users to explore additional options whenever their communication plans are flexible, in the same manner that travel planners allow users to provide a range of acceptable start and end dates for their travel.
- *Multiple alternatives selected using multiple criteria.* The planner must present the user with several options (i.e., viable alternative end-to-end paths) that meet multiple criteria, including price, bandwidth capacity, delay, the inclusion or exclusion of sub-paths from certain providers, etc. We envision that path planning services will differentiate from the competition by deploying sophisticated and specialized algorithms for selecting paths.

Each of the above considerations significantly complicate the path finding process. For instance, introducing one additional resource constraint (e.g., a delay constraint along with a cost constraint), makes the shortest path problem NP-Complete [11, Problem ND30]. Consequently, a wide range of heuristics and approximation algorithms have been developed for a diverse set of constrained shortest path problem variants [12], [13]. Also, while efficient algorithms exist for constructing k-shortest elementary (i.e., acyclic) [14] and non-elementary [15] paths, the k-constrained shortest path problem is significantly harder and has received little attention [16].

In-advance path reservations involve reserving resources along an end-to-end path for a continuous interval of time that has a specific duration and starts at a specific instant, either in the present or in the future. Algorithms for finding and reserving paths with sufficient bandwidth resources well in advance of the start of communication [17]–[19] have generally been designed for small, centrally controlled connection-oriented networks in which only a relatively small fraction of connections require such advance reservations. These algorithms may be extended to account for cost and delay constraints, but do not directly support time constraints.

The general shortest path problem with time constraints involves finding the least cost path from source to destination in a graph whose nodes can be visited within a specified time interval [20]. Similar time-constrained path problems have been studied in the context of vehicle routing [20], [21] and travel planning [22]. The problem is NP-Complete regardless of whether the shortest path is required to be elementary or is allowed to contain cycles.

III. Markeplace and Graph Model

A. The Marketplace

We consider a marketplace that includes a repository of *path services* as advertized by network services providers. Each path service is represented by the tuple:

$$(L_s, L_d, LID, L_{attr}, T_{start}, T_{end}), \; T_{start} < T_{end}$$

where L_s and L_d are the source and destination nodes, respectively, of a (physical or virtual) link with unique ID LID and attributes L_{attr}, and $[T_{start}, T_{end}]$ is the time interval during which this path service is valid. For this work, we assume that the attributes include the available bandwidth, delay, and cost of the link, whereby the latter is expressed as price per unit bandwidth or some other appropriate form; in other words,

$$L_{attr} \; = \; (L_{bw}, L_{delay}, L_{cost}).$$

This representation allows multiple distinct providers, including virtual providers who do not own any physical infrastructure, to advertize path services between the same (L_s, L_d) pairs, that can be distinguished using the unique link ID field.

Users submit to the path planner requests of the form

$$(R_s, R_d, R_{req}, \tau_e, \tau_l), \; \tau_e \leq \tau_l$$

where R_s and R_d are the source and destination node, respectively, of the requested communication service and R_{req} are user requirements that the service must meet, and $[\tau_e, \tau_l]$ is a time interval that specifies the earliest and latest start times for the service; if $\tau_e = \tau_l$, then the service must start at exactly

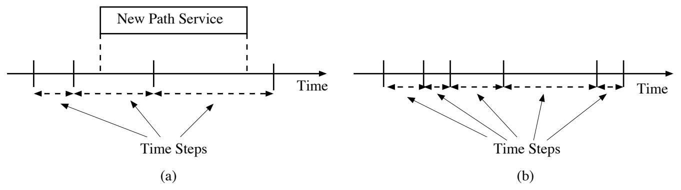

Fig. 1. The concept of time steps

time τ_e. We assume that user requirements include a minimum bandwidth along the path, an acceptable end-to-end delay, the time duration (length) of the communication, and a maximum cost that the user is willing to pay, i.e.,

$$R_{req} \ = \ (R_{bw}, R_{delay}, R_{len}, R_{cost}).$$

B. Graph of Path Services

The planner uses the path service descriptions stored in the marketplace repository to construct a graph $G = (V, E)$, where V is the set of nodes that are part of at least one service description, and E is the set of unique links defined by the service descriptions. As we mentioned earlier, the graph G will generally include parallel edges representing competing services or virtual links. Each edge includes all information associated with the corresponding link, i.e., LID, link attributes (bandwidth, delay, and cost), and the interval of time $[T_{start}, T_{end}]$ during which the edge is valid.

We assume that the planner updates the graph of path services in real time whenever each of these four events takes place: (a) when a new path service is advertized, a new edge is added to the graph; (b) when an advertisement updates an existing path service, the attributes of the corresponding edge are updated accordingly (or the edge is removed if the update cancels the service); (c) when a new reservation is established, the attributes (e.g., available bandwidth) of the path services in the end-to-end path are updated accordingly; and (d) when an existing reservation terminates, the attributes of its constituent path services are also updated.

We define a time step [17] as a continuous period of time during which the state of the network does not change; in other words, the graph of path services and their attributes remain the same throughout a time step. The planner updates the sequence of time steps whenever an advertisement creates a new path service or modifies an existing one, and when reservations are set up or terminate. Consider Figure 1(a), where three time steps are shown, representing the changes in network state before the arrival of the new path service. As seen in the figure, the time duration of the new path service overlaps with two of the time steps. Therefore the addition of this path service causes changes in the state of the network within each of the two time steps, resulting in the five time steps shown in Figure 1(b). Time steps must be similarly updated for new and departing reservations.

We have the following two results.

Lemma 1: For any set of path services that have m unique sets of $[T_{start}, T_{end}]$ time intervals, there can be at most $2m-1$ time steps.

Proof: In geometry, it is known that the number of non-overlapping segments formed by k distinct collinear points is $k-1$. Since m unique sets of $[T_{start}, T_{end}]$ time intervals include at most $2m$ distinct time instants at which a path service starts or ends, the number of non-overlapping time segments created by these instants is at most $2m-1$. Since a path service starts or ends at the boundary between two time segments, the state of the network (graph) does not change during any of the time segment. Therefore, there are at most $2m-1$ time steps. ■

Lemma 2: Consider a user request for a communciation service that may start anywhere in the interval $[\tau_e, \tau_l]$. If the time interval $[\tau_e, \tau_l]$ overlaps with n time steps, then, in order to satisfy this request, it is sufficient to run a path finding algorithm at most n times, each time with a start time equal to the beginning of one of the time steps.

Proof: Consider time step $x = [t_1, t_2]$ that overlaps with the interval $[\tau_e, \tau_l]$ of the user request. Let $\mathcal{P}$ be the set of paths that a specific path finding algorithm returns under the assumption that the communication service requested by the user starts at time t_1. Since the state of the network does not change for the duration of time step x, the same algorithm will not be able to find better paths than the ones in $\mathcal{P}$ for any start time t of the request such that $t_1 < t \leq t_2$. On the other hand, the algorithm may find worse paths when $t_1 < t \leq t_2$; this may occur if the later starting time causes the service to end within a later time step in which the network state may not be able to accommodate the quality of features of the paths in $\mathcal{P}$. ■

The above two results impose strict bounds on the search space that the planner has to explore to satisfy a user request. These bounds make path computations more efficient than the method used in [17] to divide the search space; the latter method becomes inefficient even for networks of moderate size with a relatively small number of path services.

IV. Multi-Criteria Time Constrained Paths

Our objective is to present each user requesting service with a set of Pareto-optimal time constrained paths that satisfy multiple user-specified constraints. More formally, the problem we address is the k-time constrained shortest path (k-TCSP) problem defined as follows.

Problem 1 (k-TCSP): Let $G = (V, E)$ be a graph with path services as edges such that each edge e is valid only during the time interval $[T^e_{start}, T^e_{end}]$. Consider the user request

$$(R_s, R_d, R_{req}, \tau_e, \tau_l), R_{req} \ = \ (R_{bw}, R_{delay}, R_{len}, R_{cost})$$

and an integer k. Find k least cost Pareto-optimal paths from R_s to R_d, such that each path:

1) is a concatenation of one or more path services (edges),
2) has bandwidth at least R_{bw},
3) has end-to-end delay at most R_{delay}, and
4) is valid throughout the interval $[t, t + R_{len}]$, for any $t \in [\tau_e, \tau_l]$,

where a path is considered valid in a given time interval if and only if all path services comprising the path are valid in the same interval.

We note that k-TCSP is NP-Complete since it reduces to the NP-Complete k-CSP problem [16] by letting $[T^e_{start}, T^e_{end}] = [0, \infty]$ for all edges e.

A. Dynamic Programming Algorithm for k-TCSP

Let $G = (V, E)$ be the graph of path services at the time the user request $(R_s, R_d, R_{req}, \tau_e, \tau_l), R_{req} = (R_{bw}, R_{delay}, R_{len}, R_{cost})$ arrives. We now present a dynamic programming algorithm that can be used to find Pareto-optimal paths from node R_s to node R_d that are valid in the interval $[\tau_e, \tau_l + R_{len}]$.

Define $F(i, t, R_{delay})$ as the minimum cost of any path from source R_s to the node i, $i \in V$, that starts at time t, has available bandwidth at least equal to R_{bw}, and its cumulative delay (i.e., the total delay along the path services from R_s to i) is at most R_{delay}. If no such path exists at time t, then $F(i, t, R_{delay}) = \infty$.

$F(i, t, R_{delay})$ can be calculated using the following recursion:

$$F(i,t,D) = \begin{cases} 0, & i = R_s \text{ and } D \geq 0 \\ \infty, & D < 0 \end{cases} \tag{1}$$

$$\begin{aligned} F(j,t,D) &= \min_{(i,j)\in E} \left\{ F\left(i,t,D - L^{(i,j)}_{delay}\right) + L^{(i,j)}_{cost} \right\}, \\ &\qquad \forall (i,j) \in E \;\;,\;\; D \leq R_{delay}, R_{bw} \leq L^{(i,j)}_{bw} \end{aligned} \tag{2}$$

The base case (1) simply states that (i) the cost of getting from the source node R_s to itself is zero, and (ii) the cost of going from R_s to any node i with a negative delay is infinity since no such path exists. The recursive expression (2) can be explained by noting that the minimum cost of getting from R_s to node j with a total delay of at most D, is equal to the minimum cost, over all path services $(i, j), i \neq j$, of getting from R_s to node i with a total delay of at most $D - L^{(i,j)}_{delay}$, plus the cost $L^{(i,j)}_{cost}$ of going from i to j. Note also that the minimum is taken only over edges (path services) (i, j) that have sufficient bandwidth for the user request.

The optimal solution at time t, i.e., the minimum cost of a path that starts at time t and can accommodate the user request, can be computed as:

$$F(R_d, t, R_{delay}). \tag{3}$$

Recall now that, according to Lemma 2, it is sufficient to run the path finding algorithm once for each time step that overlaps with the interval $[\tau_e, \tau_l]$ that represents the allowable start times for the user request. Let n be the number of such time steps and $t_1, \ldots, t_n$ be the time instants when the path finding algorithm must be run; according to Lemma 2, $t_1 = \tau_e$, while $t_2, \ldots, t_n$ coincide with the start of the following $n - 1$ time steps. Therefore, the overall optimal solution, i.e., the cost of the minimum-cost path for the user request starting anywhere in $[\tau_e, \tau_l]$, can be obtained as:

$$\min_{t_1,\ldots,t_n} F(R_d, t_i, R_{delay}). \tag{4}$$

We note that computing expression (3) may require the evaluation of an exponential number of paths. Furthermore, the recursion returns the cost of a minimum-cost, feasible path, if one exists, but it does not directly provide the path services (edges) comprising this path. Importantly, this expression does not compute multiple shortest paths, and hence it does not provide a solution to k-TCSP.

In the following subsection, we show that it is possible to maintain labels at the nodes of graph G during the execution of recursion (2), so as to (i) construct Pareto-optimal paths, and (ii) speed up the recursion by eliminating paths (i.e., terminating the recursion early) that will not lead to Pareto-optimal solutions.

B. Tracking Pareto-Optimal Paths

Consider an execution of the recursive algorithm (3) for a given start time t. At each node i visited by the recursion, we maintain labels to keep track of Pareto-optimal paths passing through that node. Specifically, for each path through node i, we maintain the tuple (C, D), where C (respectively, D) is the cost (respectively, delay) of the path from the source node R_s to node i[1].

Consider two paths through node i with labels (C_1, D_1) and (C_2, D_2), respectively. We say that the first path *dominates* the second, denoted by $(C_1, D_1) \prec (C_2, D_2)$, if $C_1 \leq C_2$ *and* $D_1 \leq D_2$. In other words, the dominating path is better than the dominated one in terms of both cost and delay. Note that, all paths entering node i have the exact same options as path services to continue towards the destination R_d. Therefore, it is certain that the dominated path will result in an end-to-end solution that cannot be superior to that resulting from the dominating path in terms of either cost and delay. Consequently, we eliminate the dominated path at node i by terminating the recursion at that point, which also speeds up the overall running time.

At the end of the recursion (3), we obtain Pareto-optimal paths that start at time t. We execute the recursion n times, once for each time step, as indicated in (4), and obtain Pareto-optimal paths that start in $[\tau_e, \tau_l]$. We then extract (up to) k least-cost Pareto-optimal paths from this list, and return them to the user, allowing the latter to make an informed selection.

C. k-TCSP with No Delay Constraints

Let us now consider the k-TCSP problem variant in which user requests do not impose any delay constraints. This variant may be solved in pseudopolynomial time [20] using the following steps at each of the n time instants t_i discussed in Section IV-A above: (1) remove from the graph all edges which, at time t_i, have available bandwidth less than R_{bw}; (2) run Yen's algorithm [14] to construct the k shortest paths between R_s and R_d at time t_i. These steps will determine up

[1]The label includes two additional parameters: the previous node j towards the source R_s and the unique link ID, LID, of the path service that leads from j to i. These parameters make it possible to reconstruct the path starting at the destination node, R_d, but are not essential for determining Pareto-optimal paths.

to nk shortest paths, of which we present the k shortest to the user. Since Yen's algorithm is polynomial, assuming that the number n of time instants is bounded, this algorithm will produce the k shortest paths starting anywhere in $[\tau_e, \tau_l]$ in polynomial time.

V. Numerical Results

We now present simulation results to evaluate the dynamic programming algorithm for the k-TCSP problem (Section IV-A) and the pseudopolynomial algorithm for the k-TCSP problem with no delay constraints (Section IV-C). We used BRITE [23] to generate graphs for running the simulation because it is a universal topology generator and offers more than just network connectivity at the AS level. We obtained undirected graphs by configuring BRITE to generate AS-Level Barabasi models. We set the size of the outer and inner planes to 1000 and 100 respectively, for placement of the nodes in a heavy tailed distribution. We set the growth type of the graph to be incremental in nature, we disabled the preferential connection property, and we set the average nodal degree to between 2 and 4. We used a uniform bandwidth distribution with a maximum and minimum bandwidth values of 2500 Mbps and 100 Mbps, respectively, with the additional restriction that bandwidth values be multiples of 100 Mbps. The delay L_{delay} of an edge was set proportional to the Euclidean distance between the two points in the plane representing the endpoints of the edge. We model the cost of using a link as a function of the product of bandwidth times duration of the connection. Specifically, we let the cost, L_{cost}, per unit bandwidth (i.e., 1 Mbps) to \$0.06, a value that is approximately one-tenth of the current market cost [24]. Hence, the price that a user has to pay for a connection can be expressed as \$0.06$\times R_{bw} \times R_{len}$. Finally, we let the start and end times of an edge (path service) to be in the range [0, 15 days].

We generate user requests using the following model:

- The bandwidth R_{bw} requested is uniformly distributed in the range [10, 100 Mbps] with probability 0.6, and in the range (100 Mbps, 500 Mbps] with probability 0.4.
- The duration R_{len} of the request is uniformly distributed in the ranges: [1, 30 min] (probability 0.1), [31 min, 60 min] (probability 0.1), (1 hr, 3 hr] (probability 0.6), and (3 hr, 12 hr] (probability 0.2).
- The earliest start time τ_e is between [0, 1 day] with probability 0.8, and between (1, 15 days] with probability 0.2.
- The latest start time τ_l is set to either equal to τ_e (with probability 0.5) or is uniformly distributed in the range (τ_e, τ_e+ 60 min] (with probability 0.5).
- The end-to-end delay R_{delay} is set to $\sqrt{2}$ times the delay along the Euclidean distance of the diameter in the outer plane of the topology graph.

We further assume that user requests arrive as a Poisson process with mean equal to 1 minute.

We have implemented the routing algorithms in C, and we run the simulation experiments on a Linux cluster, each node in the cluster consisting of two Xeon processors (representing a mix of 1, 2, 4, 6, or 8 cores) and 2-4 GB of memory per core. In the figures we present in this section, each data point corresponds to the average of 30 randomly generated problem instances. The figures also plot confidence intervals around the mean, estimated using the method of batch means.

Figure 2 plots the running time of the dynamic programming algorithm as a function of the number N of nodes in the graph; for these experiments, the average nodal degree was set to 2. For each problem instance, we generated 100 user requests and, hence, run the algorithm 100 times to find paths for each request. The running time shown in the figure is an average over these 100 executions. As we can see, the running time increases faster than linearly with the size of the network, but remains reasonable even for large topologies; for $N = 400$ nodes, it takes about 7-8 seconds, an amount of time comparable to what users experience in online travel sites. For comparison, Figure 3 presents the average running time of the pseudopolynomial algorithm for the k-TCSP problem with no delay constraints, as a function of the number k of shortest paths; for these experiments, we generated 1,000 user requests and the average was taken over the 1,000 executions of the algorithm. We can see that the running time increases linearly with k, and also with the network size, as expected. Overall, this algorithm runs more than one order of magnitude faster than the dynamic programming algorithm for the same network size, implying that relaxing the delay constraints makes it possible to scale to very large networks.

Finally, Figure 4 compares the number of paths returned by the two algorithms, as a function of the number k of shortest paths passed as a parameter to Yen's algorithm; for this experiment, we have used topologies with $N = 400$ nodes and average nodal degree of 2. As expected, the number of shortest paths returned by the pseudopolynomial algorithm is equal to k. However, as shown in the figure, the number of these shortest paths with a delay below the R_{delay} threshold is slightly fewer than k. Finally, the number of Pareto-optimal paths constructed by the dynamic programming algorithm is constant (i.e., independent of k), since this algorithm does not make use of Yen's k-shortest path algorithm. The number of Pareto-optimal paths is smaller than the number of shortest paths within the delay threshold due to the elimination of dominated paths during the execution of the dynamic programming algorithm.

VI. Concluding Remarks

We have introduced a new problem of finding time-constrained paths that also satisfy bandwidth and delay constraints. We have developed a dynamic programming algorithm that constructs Pareto-optimal paths. Our current work focuses on refining the algorithm to scale to topologies with thousands of nodes and high average nodal degrees.

References

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

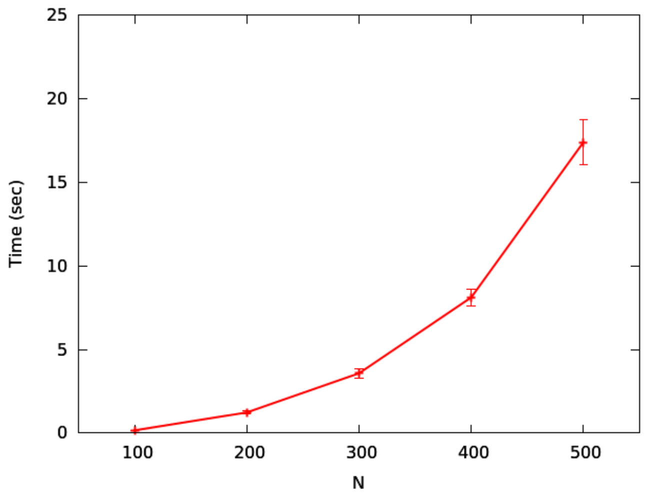

Fig. 2. Running time of the dynamic programming algorithm for k-TCSP

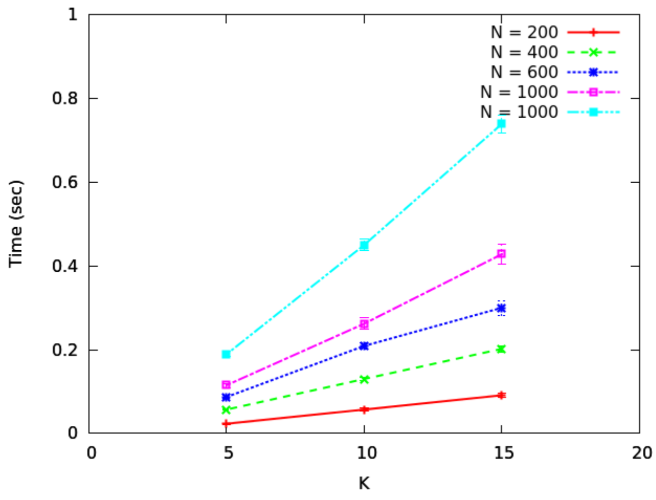

Fig. 3. Running time of the pseudopolynomial algorithm for k-TCSP with no delay constraints

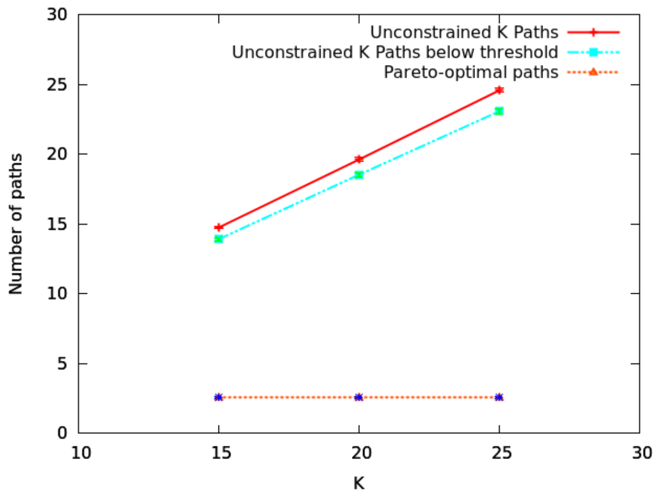

Fig. 4. Comparison of paths returned by the two algorithms

[2] A. W. Brander and M.C. Sinclair. A comparative study of k-shortest path algorithms. In *Proceedings of the 11th UK Performance Engineering Workshop*, September 1995.

[3] S. Chen and K. Nahrstedt. An overview of quality of service routing for next-generation high speed networks: Problems and solutions. *IEEE Network*, 12(6):64–79, November/December 1998.

[4] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.

[5] M. O. Ascigil and K. Calvert. Implications of source routing. In *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*, CoNEXT Student '12, pages 11–12, New York, NY, USA, 2012. ACM.

[6] P. Brighten Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 111–122, New York, NY, USA, 2009. ACM.

[7] Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. Routing as a service. Technical Report UCB/CSD-04-1327, EECS Department, University of California, Berkeley, 2004.

[8] T. Wolf, J. Griffioen, K. L. Calvert, R. Dutta, G. N. Rouskas, I. Baldine, and A. Nagurney. Choice as a principle in network architecture. In *Proceedings of ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, pages 105–106, Helsinki, Finland, August 2012. (Poster).

[9] T. Wolf, J. Griffioen, K. Calvert, R. Dutta, G. N. Rouskas, I. Baldin, and A. Nagurney. ChoiceNet: Toward an economy plane for the Internet. *ACM SIGCOMM Computer Communication Review*, 44(3):58–65, July 2014.

[10] X. Chen, T. Wolf, J. Griffioen, O. Ascigil, R. Dutta, G. N. Rouskas, S. Bhatt, I. Baldin, and K. Calvert. Design of a protocol to enable economic transactions for network services. In *Proceedings of IEEE ICC 2015*, June 2015.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., New York, 1979.

[12] M. Ziegelmann. *Constrained Shortest Paths and Related Problems*. PhD thesis, Universitaet des Saarlandes, 2001.

[13] Y. Xiao, K. Thulasiraman, G. Xue, and A. Jttner. The constrained shortest path problem: algorithmic approaches and an algebraic study with generalization. *AKCE International Journal of Graphs and Combinatorics*, (2):63–86, December 2005.

[14] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

[15] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[16] N. Shi. k constrained shortest path problem. *IEEE Transactions on Automation Science and Engineering*, 7(1):15–23, January 2010.

[17] M. Balman, E. Chaniotakisy, A. Shoshani, and A. Sim. A flexible reservation algorithm for advance network provisioning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2010)*, pages 1–11, Nov 2010.

[18] E.-S. Jung, Y. Li, S. Ranka, and S. Sahni. An evaluation of in-advance bandwidth scheduling algorithms for connection-oriented networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2008)*, pages 133–138, May 2008.

[19] S. Sahni, N. Rao, S. Ranka, Y. Li, E.-S. Jung, and N. Kamath. Bandwidth scheduling and path computation algorithms for connection-oriented networks. In *Proceedings of the Sixth International Conference on Networking (ICN 2007)*, pages 47–47, April 2007.

[20] Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Franois Soumis. Chapter 2 time constrained routing and scheduling. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35 – 139. Elsevier, 1995.

[21] Wei Wu and Qiuqi Ruan. A hierarchical approach for the shortest path problem with obligatory intermediate nodes. In *Signal Processing, 2006 8th International Conference on*, volume 4, pages –, Nov 2006.

[22] Jean-Franois Brub, Jean-Yves Potvin, and Jean Vaucher. Time-dependent shortest paths through a fixed sequence of nodes: application to a travel planning problem. *Computers and Operations Research*, 33(6):1838 – 1856, 2006.

[23] A. Medina, I. Matta, and J. Byers. Brite: A flexible generator of internet topologies. Technical report, Boston, MA, USA, 2000.

[24] William B. Norton. The internet peering playbook : Connecting to the core of the internet. 2014.