Computer Networks 138 (2018) 201–212

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

On time dependent routing algorithms for open marketplaces of path services with support for in-advance path reservation

Shireesh Bhat[a,*], George N. Rouskas[a,b], Iyad Katib[b]

[a] Department of Computer Science, North Carolina State Universtiy, Raleigh, NC, USA
[b] King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 2 June 2017
Revised 26 February 2018
Accepted 6 April 2018
Available online 7 April 2018

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 which are time sensitive. We define the problem of finding multi-criteria time-constrained paths in this context, and present algorithms to construct these paths and also provide support for in-advance path reservation.

1. 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.

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.,

* Corresponding author.
E-mail addresses: sbhat@ncsu.edu (S. Bhat), rouskas@ncsu.edu (G.N. Rouskas), iakatib@kau.edu.sa (I. Katib).

https://doi.org/10.1016/j.comnet.2018.04.006
1389-1286/

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 is an extension of our earlier work [11] and can 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 2. We present a model for the marketplace and the graph of path services that the planner maintains in Section 3. We define a suite of problems related to finding multi-criteria time constrained paths, and develop algorithmic solutions for them in Section 4. We present numerical results in Section 5, and we summarize and discuss future work in Section 6.

2. 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 any 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 [12, Problem ND30]. Consequently, a wide range of heuristics and approximation algorithms have been developed for a diverse set of constrained shortest path problem variants [13,14]. Also, while efficient algorithms exist for constructing k-shortest elementary (i.e., acyclic) [15] and non-elementary [16] paths, the k-constrained shortest path problem is significantly harder and has received little attention [17].

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 [18–20] 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 [21]. Similar time-constrained path problems have been studied in the context of vehicle routing [21,22] and travel planning [23]. The problem is NP-Complete regardless of whether the shortest path is required to be elementary or is allowed to contain cycles.

2.1. Problem classification

The shortest path problems and network reservation algorithms can be classified based on the objective function(s) as shown in Table 1 and Table 2 respectively. In this classification we use the notation $\alpha/\beta/\gamma/\pi/\phi$, $\alpha = \{G, T\}$ where G denotes problems which are not time sensitive and T denotes problems which are time sensitive, $\beta \in \mathbb{N}$ denotes the number of resource constrained objective functions, $\gamma = \{P, F\}$ where P denotes Pareto or non-dominated paths and F denotes all feasible paths, $\pi = \{1, K\}$ which indicates if the problem finds 1 optimal path or K paths in non-decreasing order of cost, $\phi = \{O, A\}$ where O denotes optimal paths and A denotes approximate or subset paths, to categorize problems and this classification differs from the one in [24] as we extend the classification to problems which are time driven, non-dominated and those which consider K non elementary shortest paths.

Dijkstra's algorithm [1] is designed to find a single source shortest path and runs in polynomial time. The rest of the algorithms shown in Table 1 either run in pseudopolynomial or exponential time. The concept of non-dominated or Pareto solution(s) was first defined on multiplicative lattices. It was then extended to network graph models [25–27] using the label setting/correction approach. Namorado Climaco and Queiros Vieira Martins [28], and Climaco et al. [29] use the K-shortest [30,31] paths approach to find the Pareto solution(s). Joksch [32] introduced the notion of shortest paths with constraints. The shortest path problems with resource

Table 1
Classification of shortest path problems.

Notation	Original algorithm	Variations
G/0/F/1/O	Dijkstra [1]	
G/0/P/1/O	Brown and Strauch [42]	Thuente [25], Hansen [26], Martins [27]
G/0/P/K/O	Namorado Climaco and Queiros Vieira Martins [28]	Climaco et al. [29]
G/1/F/1/O	Joksch [32]	
G/1/F/K/O	Handler and Zang [33], de Queirs Vieira Martins et al. [31]	Shi [43]
G/0/P/1/A	Henig [34], Warburton [35]	
G/1/F/1/A	Hassin [36]	
T/0/F/1/O	Desrochers and Soumis [37], Desaulniers and Villeneuve [44]	
T/0/P/1/O	Hamacher et al. [38]	
T/0/F/K/O	Problem 4 in this work	
T/1/P/1/O	Problem 1 in this work	

Table 2
Classification of network reservation algorithms.

Notation	Original algorithm	Variations
T/1/F/1/O	Guerin and Orda [45]	Balman et al. [46]
T/0/F/K/O	Problem 4 in this work	
T/1/P/1/O	Problem 1 in this work	

constraints was solved using Lagrangian relaxation by Handler and Zang [33] and using *K*-shortest paths algorithm by de Queirs Vieira Martins et al. [31]. Henig [34], Warburton [35] and Hassin [36] introduced the concept of approximate solutions to Pareto and constrained shortest path problems. The concept of shortest path problem with time windows was introduced by Desrochers and Soumis [37] and later it was defined for non-dominated shortest path problems with time windows by Hamacher et al. [38] for vehicle routing problems. The concept of time windows was extended to network graph models and the concept of non-dominated and constrained shortest path problems with time windows is defined in Problem 1 through Problem 4. This work connects the domain of shortest path problem with advance resource reservation in network graph models. Some of the notable network reservation algorithms without preemption are shown in Table 2

Aneja et al. [39], Beasley and Christofides [40] and Dumitrescu and Boland [41] employ preprocessing to improve the running time of several constrained shortest path algorithms.

2.2. Finding Pareto paths using k-shortest paths

There are two main challenges when finding Pareto paths: the running time for finding the Pareto solutions, and selecting one optimal/sub-optimal solution among these Pareto solutions.

There are primarily two ways of finding Pareto solutions.

1. Label setting/correction approach [38], and
2. *K*-shortest cost paths [28].

Since finding Pareto solution(s) is NP-Hard [28], both the approaches take exponential running time. For certain graph problems one of the approaches might be better suited than the other. In Fig. 1 we have three examples where the number and the gap between the Pareto solutions is different. Suppose every path from source to destination has a unique pair of (cost, time) solution and each corresponds to one of the of the *K*-shortest cost paths. In that case, for all the three examples we can quickly find all the Pareto solutions relatively quickly compared with the labeling approach.

The gap between the Pareto solutions and the number of Pareto solutions which can be found using the *K*-shortest paths plays an important role in determining which approach is the best. In Fig. 2 we have an example which is not suitable for the *K*-shortest cost/time paths as we have multiple paths which have the same cost or time and we are not guaranteed to find the shortest cost/time path which minimizes both cost and time for a small value of *K* when we are searching for *K*-shortest cost paths or the *K*-shortest time paths. So, we have to find *K*-shortest cost/time paths for a large value of *K* before we find Pareto solutions. This applies for the bi-criteria and multi-criteria Pareto solutions.

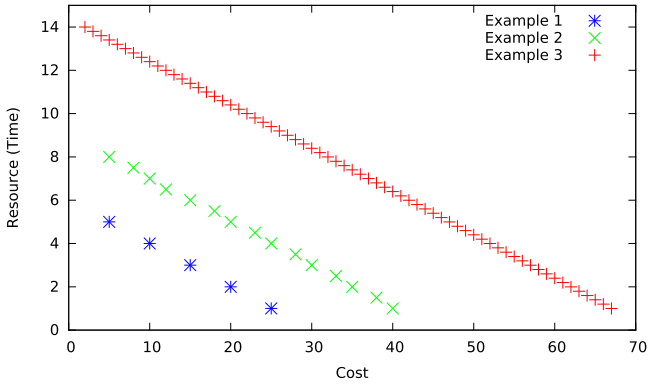

Fig. 1. The gap between Pareto solutions in the bi-criteria case.

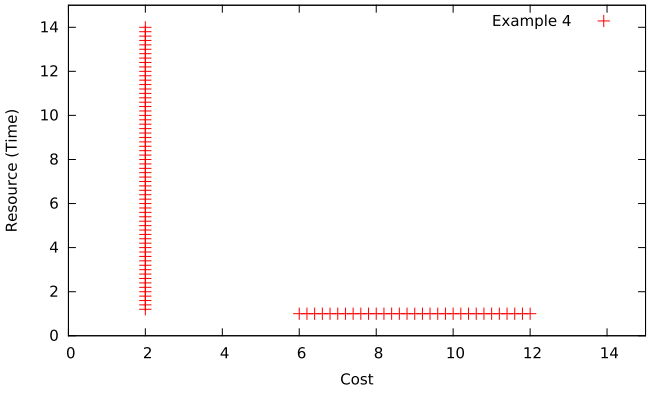

Fig. 2. Uniqueness of *K* shortest paths in the bi-criteria case.

Once we find the Pareto solutions, we still have the problem of choosing one or more among these. A utility function is defined as a mapping from the set of Pareto solutions to a combined solution. The utility function [47] can be designed in several ways and it is also possible that there might be several utility functions which can be applied to these Pareto solutions.

Finally regarding the tradeoff between running time and number of Pareto solutions, we state the following conjecture and provide counter examples to disprove it.

Conjecture 1. *Small number of Pareto paths ⇔ Lower running time.*

We provide two counter examples to disprove the conjecture from both directions of the equality. In Fig. 3 we have shown a

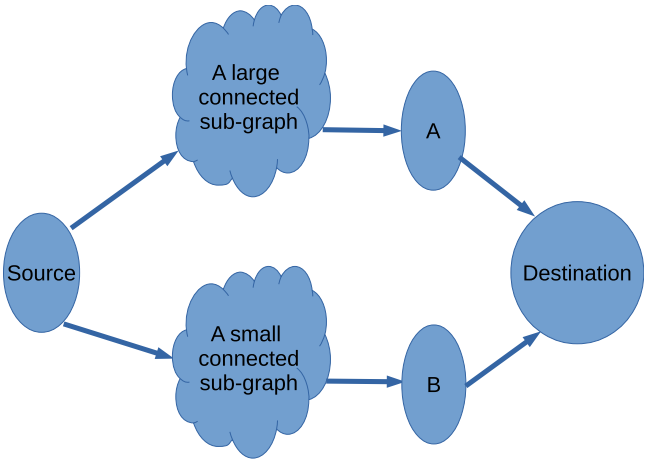

Fig. 3. Counter Example 1.

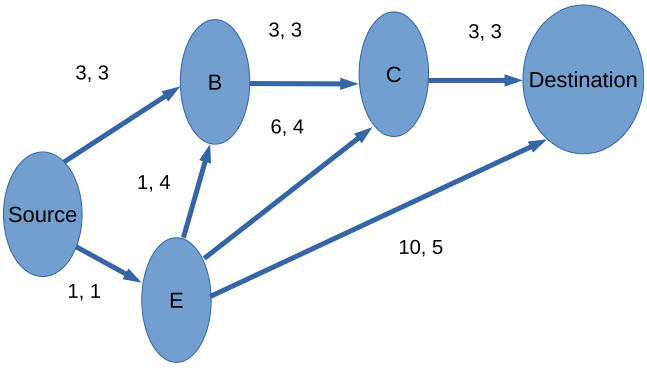

Fig. 4. Counter Example 2.

'Source' node and a 'Destination' node along with two intermediary nodes 'A' and 'B' and two sub-graphs connecting them. Suppose we have two Pareto paths from 'Source' to 'B' and hundred Pareto paths from 'Source' to 'A'. If the large connected sub-graph is dense we end up spending a lot of time traversing the edges and finding the Pareto paths. Suppose the Pareto paths from 'Source' to 'Destination' via 'B' dominates all the Pareto paths from 'Source' to 'Destination' via 'A', we end up having just two Pareto paths from 'Source' to 'Destination'. So a small number of Pareto paths does not imply a lower running time. In Fig. 4 we have shown a Directed Acyclic Graph with (cost, delay) attributes mentioned corresponding to every edge. Each path from 'Source' to 'Destination' forms a Pareto path, so we end up with four Pareto paths on a very small graph with a lower running time. So a small running time does not imply a lower number of Pareto paths. The number of Pareto paths and the running time depends not only on N, the size of the graph but also on the graph structure.

3. Markeplace and graph model

3.1. 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, cost of the link, energy spent for traversing the link, whereby the cost and energy attributes are expressed as price and joules per unit bandwidth; in other words,

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

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

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

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 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.,

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

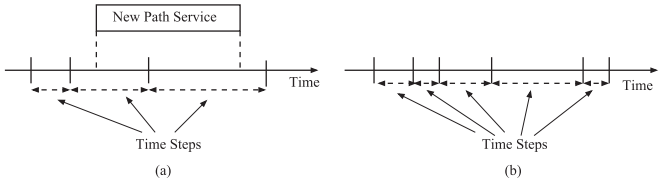

Fig. 5. The concept of time steps.

3.2. 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 is part of at least one service description, and E is the set of unique links defined by the service descriptions. In general G will 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 or when a provider withdraws an existing path service advertisement the edge is removed; (b) when a provider modifies an existing path service advertisement, the attributes of the corresponding edge are updated to reflect the new value(s); (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 to reflect the resources which have been assigned to the new reservation; (d) when an existing reservation terminates the attributes of the path service are updated to reflect the resources which have been reclaimed and are now available in the Marketplace for new reservation.

We define a time step [18] 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 established, terminated or expire. Consider Fig. 5(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 Fig. 5(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, where* $m > 0$.

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 communication 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 [18] to divide the search space; the latter method becomes inefficient even for networks of moderate size with a relatively small number of path services.

4. Multi-criteria time constrained paths

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

Problem 1 (Non-dominated k-TCSP with resource constraints (ND-k-TCSPRC)). Let $G = (V, E)$ be a graph with path services as edges such that each edge e is valid only during the time interval $[T^e_{start}, T^e_{end}]$. Let U be a utility function defined by the user which maps the set of Pareto solutions to the set $\mathbb{R}$. 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 the top k Pareto-optimal paths from R_s to R_d which provide the maximum utility, 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.

Problem 2 (Non-dominated k-TCSP (ND-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 $[T^e_{start}, T^e_{end}]$. Let U be a utility function defined by the user which maps the set of Pareto solutions to the set $\mathbb{R}$. 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 the top k Pareto-optimal paths from R_s to R_d which provide the maximum utility, such that each path:

1. is a concatenation of one or more path services (edges),
2. has bandwidth at least R_{bw},
3. 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 both NDTCSP and NDTCSPRC are in the class NPC [48] even for one time instance. To keep the notations uniform for solving the problem we set R_{delay} to ∞ for Problem 2.

Problem 3 (k-TCSP with resource constraints (k-TCSPRC)). 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 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.

This reduces to the k-CSP problem [17] which is known to be in NPC even for one time instance.

Problem 4 (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 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. 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 this is pseudo-polynomial algorithm [49] even for one time instance. To keep the notations uniform for solving the problem we set R_{delay} to ∞ for Problem 4.

4.1. Dynamic programming algorithm for Problems 1 and 2

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 set the utility function

$U \propto (1/\Sigma_{L \in \text{Pareto Path}} L_{cost})$

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:

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

$$F(j,t,D) = \min_{(i,j)\in E} \left\{ F\left(i, t, D - L^{(i,j)}_{delay}\right) + L^{(i,j)}_{cost} \right\},$$

$$\forall (i,j) \in E \ , \ D \leq R_{delay}, R_{bw} \leq L^{(i,j)}_{bw} \tag{2}$$

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:

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

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.

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:

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

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 algorithm by discarding labels that will not lead to Pareto-optimal solutions.

4.1.1. 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]

[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.

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. When we add a third criteria, energy, the dominating path is better than the dominated one for all three attributes i.e., cost, delay, and energy. 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.

4.2. K-shortest cost paths algorithm for Problems 3 and 4

In Problem 4, which is a variation of the TCSP problem, the user does not request non dominant paths but instead requests k least cost paths without resource constraints. Problem 4 may be solved in pseudopolynomial time [21] if the number of time instants is finite using the following steps at each of the n time instants t_i discussed in Section 4.1 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 [15] to construct the k shortest paths between R_s and R_d at time t_i. These steps will determine up to nk shortest paths, of which we present the k shortest to the user. Since Yen's algorithm is polynomial, assuming k and the number of time instants are bounded, this algorithm will produce the k shortest paths starting anywhere in $[\tau_e, \tau_l]$ in polynomial time.

In Problem 3, which is a variation of the TCSP problem, the user does not request non dominant paths but instead requests k least cost paths with resource constraints. This is identical to the problem defined in [43] but now in the context of time windows. The algorithmic approach to solving Problem 3 is similar to the one above but with an additional checking of the resource constraint done at every step when we relax the edge i.e. add a node to the list of explored nodes.

5. Numerical results

We now present simulation results to evaluate the algorithms for Problems 1–4.

We used BRITE [50] 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 (Autonomous System)-Level Barabasi models. The way BRITE works is by first placing the nodes in a plane whose dimension is specified in the configuration file or through the Graphical User Interface (GUI). The nodes of the generated topology are distributed in a plane divided into *HS* x *HS* squares, where *HS* denotes the size of one side of the outer plane. Each one of these high-level squares is further subdivided into smaller *LS* x *LS* low-level squares, where *LS* denotes the size of one side of the inner planes. Each low-level square can be assigned at most one node. Next, the nodes are interconnected to represent the connections of the various AS's. The placement of the nodes and the connections between the nodes can be done in several ways as mentioned in the BRITE user manual and the models generated are in sync with the Internet Power laws. This

Table 3
Mapping of problems to graph models.

Graph model	Link cost	Link delay	R_{delay}	Link energy	Problem
1	Fixed	$\propto$ Euclidean distance	Finite	NA	1 and 3
2	Fixed	$\propto$ Euclidean distance	∞	NA	2 and 4
3	$\propto$(1/Link Delay)	$\propto$ Euclidean distance	∞	NA	2 and 4
4	$\propto$(1/Link Delay)	$\propto$ Euclidean distance	Finite	NA	3
5	$\propto$(1/Link Delay)	$\propto$ Euclidean distance	∞	Uniformly distributed	2

is followed by assigning link attributes which is user configurable and finally the Network graph is output using a user specified format. 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. L_{delay}, the link delay was set proportional to the Euclidean distance between the two points in the plane representing the endpoints of the edge. L_{energy}, the energy consumption of an edge was set independent of both the link delay and link cost. L_{energy} is uniformly distributed between [1–1000].

We used two cost models. In the first model, the cost of using a link is a function of the product of bandwidth times the 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 [51]. Hence, the price that a user has to pay for a connection can be expressed as $\$0.06 \times R_{bw} \times R_{len}$. In the second model. the link cost is negatively correlated to the delay. 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 h, 3 h] (probability 0.6), and (3 h, 12 h] (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 for Problems 1 and 3 and is set to ∞ for Problems 2 and 4.

We further assume that user requests arrive as a Poisson process with mean equal to 1 min. We use five different graph models to evaluate the algorithms used for solving the problems defined earlier and the mapping of the graph models to the corresponding problems is captured in Table 3.

We have implemented the routing algorithms in Problem 3, 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 2-4GB of memory per core. In the figures we present in this section, each data point corresponds to the average of 30 randomly generated sequence of 100 user requests. We maintain the same network topology for running our experiments. We have observed that varying the network topology (graph) instance for the same N for running the sequence of user requests produces results which are hard to discern from each other. The figures also plot confidence intervals (95%) around the mean, estimated using the method of batch means.

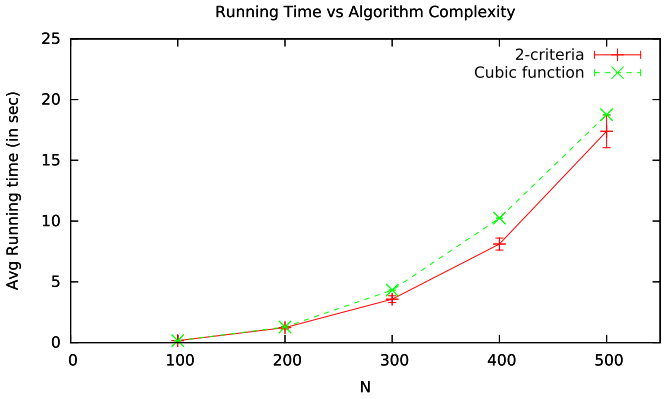

Fig. 6. Running time of the dynamic programming algorithm for Problem 1.

5.1. Model 1: Fixed cost and finite threshold Delay

Fig. 6 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 and 8 s, an amount of time comparable to what users experience in online travel sites. We also plot the running time of a $O(N^3)$ function to compare the running time of the dynamic programming algorithm with 2-criteria. We use the reference time for N = 100 to plot the extrapolated running times for N = 200 to N = 500. We expect a running time which is between $O(N^3)$ and an exponential running time since we use a variation of Label Correction Algorithm and also use an adjacency matrix. Since the running time of the algorithm is very close to the cubic function we can claim that the algorithm has been efficiently implemented in the context of Model 1.

5.2. Model 2: Fixed cost and no threshold delay

The second model compares the impact of having no resource constraints on Problems 2 and 4

Fig. 7 presents the average running time of the algorithm for solving Problem 4, as a function of the number k of shortest paths; for these experiments, we generated 1000 user requests and the average was taken over the 1000 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 re-

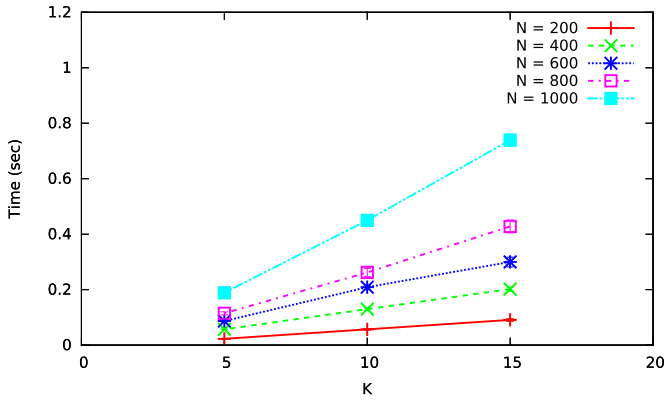

Fig. 7. Running time of the *k*-shortest cost paths algorithm for Problem 4 with no delay constraints.

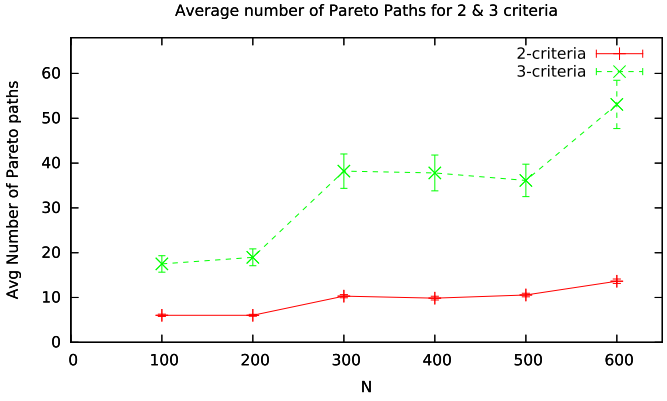

Fig. 8. Average number of Pareto paths found.

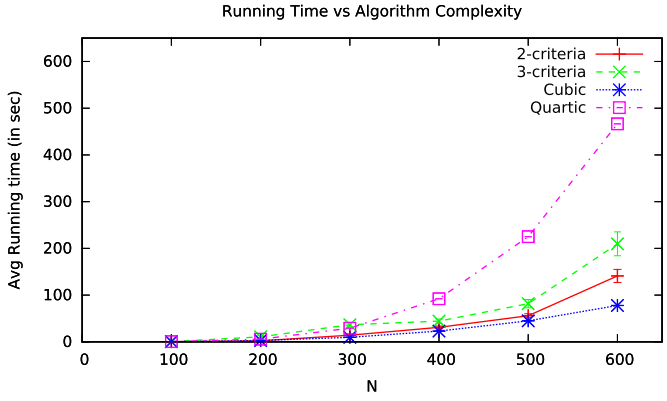

Fig. 9. Average running time for the dynamic programming algorithm.

laxing the delay and dominance constraints makes it possible to scale to very large networks.

5.3. Model 3: No Threshold delay and cost negatively correlated to delay

The third model is useful in two ways. From an experimental point of view, this is one of the ways of increasing the number of Pareto paths using the underlying graph with the same delay and average nodal degree. From an applications point of view, taking the analogy of ISPs, a link with the maximum delay would likely represent an edge which spans across continents/countries and the low cost would be amortized by the amount of traffic going through it; taking the analogy of airlines, a link with the maximum delay would likely represent a long distance flight and the low cost would be amortized by the full utilization, while the small delays would likely represent a short connecting flight and the high cost would likely compensate for any under utilization. In the network domain, the base cost represents per Mbps and per unit time. For the airline domain, the equivalent could be per person and per unit time spent in the flight.

5.4. Model 4: Threshold delay and cost negatively correlated to delay

In this model, the link costs in the graph are negatively correlated to link delay and the user requests are modeled with finite and fixed threshold delay. We evaluate Problem 3 and present the results in Section 5.7.

5.5. Model 5: Three-criteria Pareto paths

We extend the third model by adding another edge attribute "energy" consumption and evaluate Problem 2. In Figs. 8 and 9 we plot the average number of Pareto paths found and the average time it takes to find them and compare it with the two-criteria case presented in Model 3. In Fig. 9 we also plot the running time of a $O(N^3)$ function and a $O(N^4)$ function to compare the running time of the dynamic programming algorithm with 2-criteria and 3-criteria. We use the reference time for N = 100 to plot the extrapolated running times for N = 200 to N = 600. We expect a running time which is between $O(N^3)$ and an exponential running time since we use a variation of Label Correction Algorithm and also use an adjacency matrix. Since the running time of the algorithm is slightly higher than a cubic function but well below a Quartic function we can claim that the algorithm has been efficiently implemented for both 2 and 3 criteria.

5.6. Evaluation of Models 1, 2 and 3 for the non-dominated K path problem variations

We observe that the running time of the algorithm is significantly higher for the third model compared to the first and second models. This is mainly due to more non dominant paths being stored at the intermediate nodes in the graph. We observe that the number of Pareto paths has some influence on the running time of the algorithm, but the major factor influencing the running time of the dynamic programming algorithm which doesn't assume positive link attributes is N, the size of the graph. We also observe that the running time and the number of Pareto paths do not increase monotonically with N, this can be explained by going back to the Conjecture 1. The way AS-level topologies are represented in BRITE by taking into account the Internet Power laws, influences the Network (graph) structure and the simulation results (Figs. 10 and 11).

5.7. Evaluation of Models 1, 2, 3 and 4 for the K path problem variations

The average running time, the actual paths found when searching for K paths and the Pareto paths found among them, per user request with 95% confidence interval for 30 simulation runs with each simulation run consisting of 100 user requests are shown in Figs. 12, 13, and 14 respectively. The figures highlight the difference between Models 1 and 2.

Figs. 15–17 highlight the difference between Models 3 and 4.

The average number of Pareto paths among the K shortest paths for $K = 5$, and $K = 10$, is small for both types of problems, but marginally higher for the problem instance without delay constraints as seen in Figs. 12 and 15. The difference between the two models is small to draw any definite conclusions, but we can infer

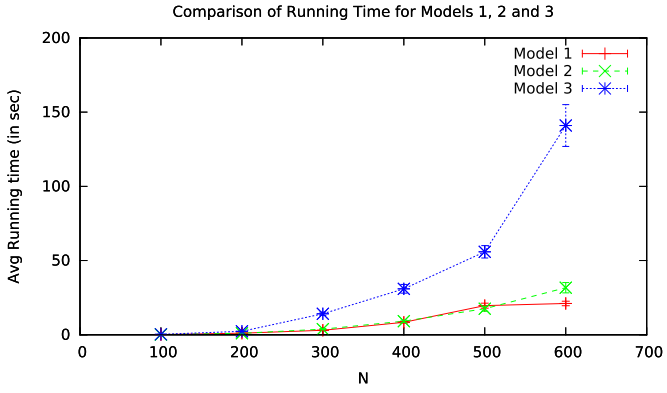

Fig. 10. Running time as a function of N.

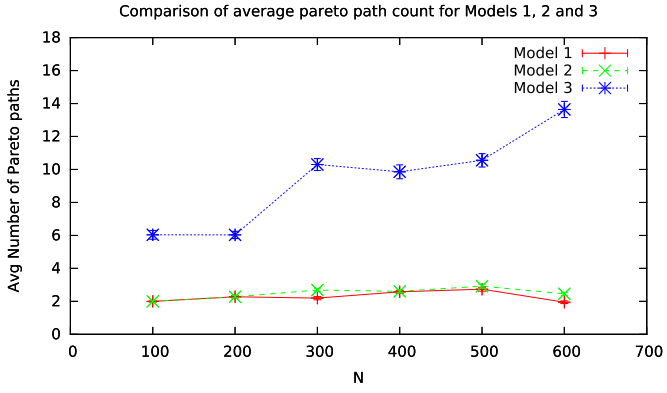

Fig. 11. Avg number of Pareto paths as a function of N.

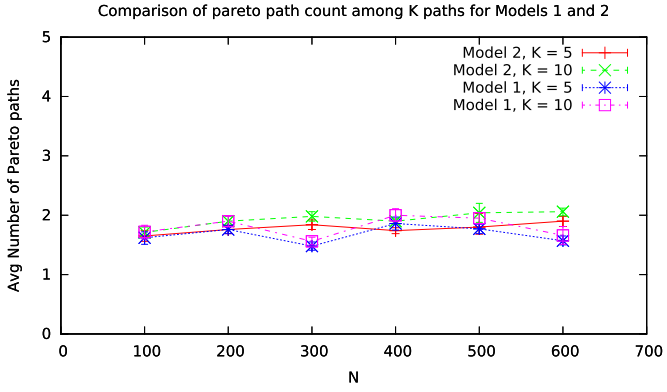

Fig. 12. Pareto paths among *K* paths for Models 1 and 2.

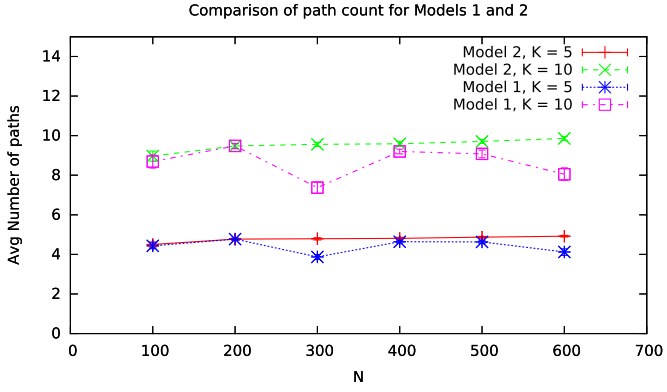

Fig. 13. Total paths for Models 1 and 2.

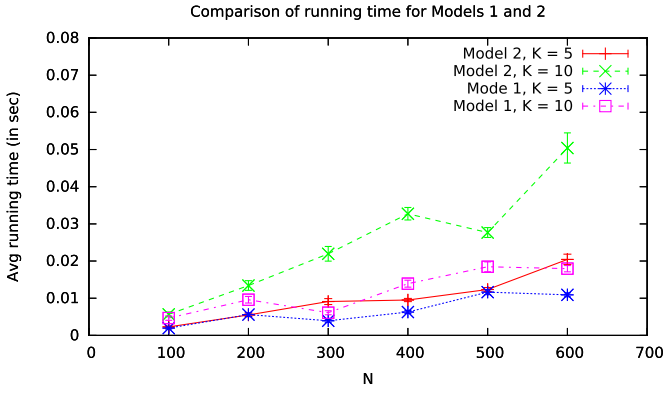

Fig. 14. Running time for Models 1 and 2.

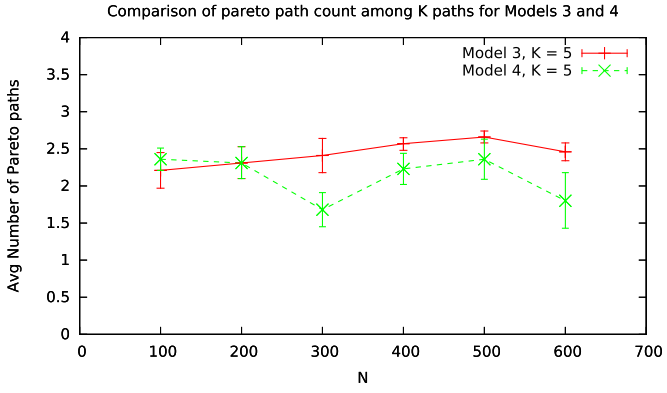

Fig. 15. Pareto paths among *K* paths for Models 3 and 4.

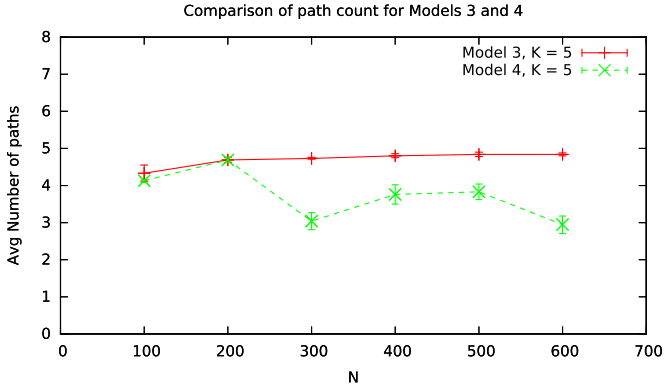

Fig. 16. Total paths for Models 3 and 4.

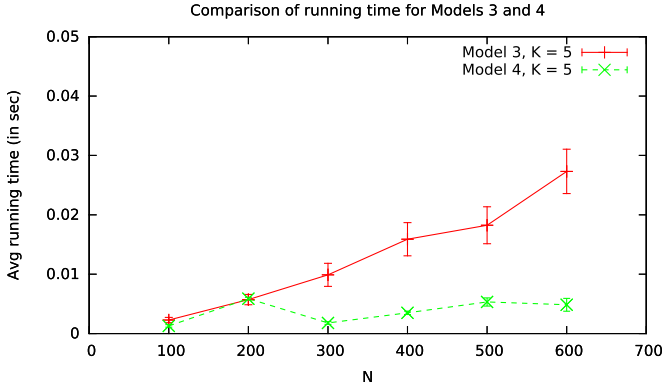

Fig. 17. Running time for Models 3 and 4.

that the finite threshold delay might lead to paths which are clustered around this threshold leading to paths which are very similar and also lesser in number giving a smaller number of Pareto paths.

The average number of total paths is also smaller for Model 1 compared to Model 2 as seen in Fig. 13 because of discarding some paths which exceed the threshold delay. This observation is consistent for Models 3 and 4 as seen in Fig. 16.

The average running time for Model 2 is higher compared to Model 1 as seen in Fig. 14, which is the result of relatively lesser paths being found which are below the threshold delay. The affect of a resource constraint i.e., threshold delay on the K shortest cost path problem doesn't lead to an increase in time compared to the K shortest cost problem without resource constraints for the graph instances considered in our experiments. This observation is also consistent with Models 3 and 4 as seen in Fig. 17

The observation made while comparing the results for both the problems is influenced by the graph/user model. These observations will change for a different graph/user model.

6. Summary and future work

We have introduced a new problem of finding time-constrained paths that presents the user with multiple flavors of composed services:

- Non dominated and resource constrained composed services
- Non dominated composed services
- Resource constrained k composed services
- k composed services

The composed services returned by the Planner is used to perform in-advance path reservation. The work presented in this paper is state of the art in the domain of Network path reservation where the reservation request can be specified using various criteria as mentioned above. This is also the most efficient and optimal way of finding a network path for a continuous interval of time when the start time is flexible.

Our current work focuses on providing the "Choice" of optimal vs. approximate solutions to the user. The trade-off is the running time to find the optimal solution vs. the optimality gap between the approximate and the optimal solution. For the resource constrained single source shortest path problem, Lagrangian Relaxation methods and preprocessing can lead to improvements in both time and space. Similarly, for the Pareto optimal paths problem the trade-off is the complete set of Pareto paths vs. the subset of the Pareto paths. There are broadly two ways for obtaining the subset of Pareto paths quickly. First, by defining one attribute which is a weighted average of all the attributes and second, by using ratio restricted lengths for the attributes.

Acknowledgments

This work was supported in part by the National Science Foundation under grant no. 1111088. The authors would like to thank the anonymous reviewers for their valuable feedback on earlier drafts of this paper.

References

[1] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[2] A.W. Brander, M. Sinclair, A comparative study of k-shortest path algorithms, in: Proceedings of the 11th UK Performance Engineering Workshop, 1995.
[3] S. Chen, K. Nahrstedt, An overview of quality of service routing for next-generation high speed networks: problems and solutions, IEEE Netw. 12 (6) (1998) 64–79.
[4] V.P. Kompella, J.C. Pasquale, G.C. Polyzos, Multicast routing for multimedia communication, IEEE/ACM Trans. Netw. 1 (3) (1993) 286–292.
[5] M.O. Ascigil, K. Calvert, Implications of source routing, in: Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop, in: CoNEXT Student '12, ACM, New York, NY, USA, 2012, pp. 11–12.
[6] P.B. Godfrey, I. Ganichev, S. Shenker, I. Stoica, Pathlet routing, in: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, in: SIGCOMM '09, ACM, New York, NY, USA, 2009, pp. 111–122.
[7] K. Lakshminarayanan, I. Stoica, S. 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, A. Nagurney, Choice as a principle in network architecture, in: Proceedings of ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM), Helsinki, Finland, 2012, pp. 105–106. (Poster)
[9] T. Wolf, J. Griffioen, K. Calvert, R. Dutta, G.N. Rouskas, I. Baldin, A. Nagurney, ChoiceNet: toward an economy plane for the Internet, ACM SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 58–65.
[10] X. Chen, T. Wolf, J. Griffioen, O. Ascigil, R. Dutta, G.N. Rouskas, S. Bhat, I. Baldin, K. Calvert, Design of a protocol to enable economic transactions for network services, in: Proceedings of IEEE ICC 2015, 2015.
[11] S. Bhat, G.N. Rouskas, On routing algorithms for open marketplaces of path services, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability, W. H. Freeman and Co., New York, 1979.
[13] M. Ziegelmann, Constrained Shortest Paths and Related Problems, Ph.D. thesis Universitaet des Saarlandes, 2001.
[14] Y. Xiao, K. Thulasiraman, G. Xue, A. Jttner, The constrained shortest path problem: algorithmic approaches and an algebraic study with generalization, AKCE Int. J. Graphs Comb. (2) (2005) 63–86.
[15] J.Y. Yen, Finding the k shortest loopless paths in a network, Manag. Sci. 17 (11) (1971) 712–716.
[16] D. Eppstein, Finding the k shortest paths, SIAM J. Comput. 28 (2) (1998) 652–673.
[17] N. Shi, k constrained shortest path problem, IEEE Trans. Autom. Sci.Eng. 7 (1) (2010) 15–23.
[18] M. Balman, E. Chaniotakisy, A. Shoshani, 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), 2010, pp. 1–11.
[19] E.-S. Jung, Y. Li, S. Ranka, 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), 2008, pp. 133–138.
[20] S. Sahni, N. Rao, S. Ranka, Y. Li, E.-S. Jung, N. Kamath, Bandwidth scheduling and path computation algorithms for connection-oriented networks, in: Proceedings of the Sixth International Conference on Networking (ICN 2007), 2007. pp:47–47
[21] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis, Chapter 2 time constrained routing and scheduling, in: C.M.M.O. Ball, T.L. Magnanti, G. Nemhauser (Eds.), Network Routing, Handbooks in Operations Research and Management Science, 8, Elsevier, 1995, pp. 35–139.
[22] W. Wu, Q. Ruan, A hierarchical approach for the shortest path problem with obligatory intermediate nodes, in: Signal Processing, 2006 8th International Conference on, 4, 2006. pp.–
[23] J.-F. Brub, J.-Y. Potvin, J. Vaucher, Time-dependent shortest paths through a fixed sequence of nodes: application to a travel planning problem, Comput. Oper. Res. 33 (6) (2006) 1838–1856.
[24] Z. Tarapata, Selected multicriteria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms, Int. J. Appl. Math. Comput. Sci. 17 (2) (2007) 269–287.
[25] D.J. Thuente, Two algorithms for shortest paths through multiple criteria networks, in: P.C. WANG, A.L. Schoenstadt, B.I. Russak, C. Comstock (Eds.), Information Linkage Between Applied Mathematics and Industry, Academic Press, 1979, pp. 567–573.
[26] P. Hansen, Bicriterion Path Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 109–127.
[27] E.Q.V. Martins, On a multicriteria shortest path problem, Eur. J. Oper. Res. 16 (2) (1984) 236–245.
[28] J.C. Namorado Climaco, E. Queiros Vieira Martins, A bicriterion shortest path algorithm, Eur. J. Oper. Res. 11 (4) (1982) 399–404.
[29] J.C.N. Climaco, J.M.F. Craveirinha, M.M.B. Pascoal, A bicriterion approach for routing problems in multimedia networks, Networks 41 (4) (2003) 206–220.
[30] E.L. Lawler, A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem, Manag. Sci. 18 (7) (1972) 401–405.
[31] E. de Queirs Vieira Martins, M.M.B. Pascoal, J.L.E.D. Santos, S. Olariu, Deviation algorithms for ranking shortest paths., Int. J. Found. Comput. Sci. 10 (3) (1999) 247.
[32] H. Joksch, The shortest route problem with constraints, J. Math. Anal. Appl. 14 (2) (1966) 191–197.
[33] G.Y. Handler, I. Zang, A dual algorithm for the constrained shortest path problem, Networks 10 (4) (1980) 293–309.
[34] M.I. Henig, The shortest path problem with two objective functions, Eur. J. Oper. Res. 25 (2) (1986) 281–291.
[35] A. Warburton, Approximation of pareto optima in multiple-objective, shortest–path problems, Oper. Res. 35 (1) (1987) 70–79.

[36] R. Hassin, Approximation schemes for the restricted shortest path problem, Math. Oper. Res. 17 (1) (1992) 36–42.
[37] M. Desrochers, F. Soumis, A generalized permanent labelling algorithm for the shortest path problem with time windows., INFOR 26 (3) (1988) 191–212.
[38] H.W. Hamacher, S. Ruzika, S.A. Tjandra, Algorithms for time-dependent bicriteria shortest path problems, Discret. Optim. 3 (3) (2006) 238–254.
[39] Y.P. Aneja, V. Aggarwal, K.P.K. Nair, Shortest chain subject to side constraints, Networks 13 (2) (1983) 295–302.
[40] J.E. Beasley, N. Christofides, An algorithm for the resource constrained shortest path problem, Networks 19 (4) (1989) 379–394.
[41] I. Dumitrescu, N. Boland, Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem, Networks 42 (3) (2003) 135–153.
[42] T. Brown, R. Strauch, Dynamic programming in multiplicative lattices, J. Math. Anal. Appl. 12 (2) (1965) 364–370.
[43] N. Shi, K constrained shortest path problem, IEEE Trans. Autom. Sci. Eng. 7 (1) (2010) 15–23.
[44] G. Desaulniers, D. Villeneuve, The shortest path problem with time windows and linear waiting costs, Transp. Sci. 34 (3) (2000) 312–319.
[45] R.A. Guerin, A. Orda, Networks with advance reservations: the routing perspective, in: Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), 1, 2000, pp. 118–127. vol.1
[46] M. Balman, E. Chaniotakisy, A. Shoshani, A. Sim, A flexible reservation algorithm for advance network provisioning, in: 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2010, pp. 1–11.
[47] P. Fishburn, Utility Theory for Decision Making, Wiley, New York, 1970.
[48] N. Shi, S. Zhou, F. Wang, Y. Tao, L. Liu, The multi-criteria constrained shortest path problem, Transp. Res. Part E 101 (2017) 13–29.
[49] J.Y. Yen, Finding the k shortest loopless paths in a network, Manag. Sci. 17 (11) (1971) 712–716.
[50] A. Medina, I. Matta, J. Byers, BRITE: A Flexible Generator of Internet Topologies, Technical Report, 2000. Boston, MA, USA
[51] W.B. Norton, The Internet Peering Playbook: Connecting to the Core of the Internet (2014), http://drpeering.net/core/bookOutline.html.

Shireesh Bhat received his Ph.D. degree in Computer Science at North Carolina State University in 2017 and the B.E. degree in Computer Science from RVCE, Bangalore, India in 2005. His research interests include Graph algorithms and Network Optimization.

George N. Rouskas I am an Alumni Distinguished Graduate Professor with the Department of Computer Science at North Carolina State University. Since January 2014, I serve as Director of Graduate Programs for the department. I received an undergraduate degree in Computer Engineering from the National Technical University of Athens, and MS and PhD degrees in Computer Science from the Georgia Institute of Technology. During the 2000-2001 academic year, I spent a sabbatical term at Vitesse Semiconductor, Morrisville, NC. I have been an invited Professor at King Abdulaziz University, Saudi Arabia, Paris VI University, France, and University of Evry, France, on several occasions.

My research interests are in the broad field of computer networking – specifically, optical networks, Internet architectures and protocols, network design and optimization, performance modeling, and scheduling. My research has received funding from federal and defense agencies as well as industry. As of 2018, 24 PhD students and 12 Masters thesis students have graduated under my supervision. I am author of the monograph "Internet Tiered Services: Theory, Economics and Quality of Service" (Springer, 2009), coeditor of the book "Next-Generation Internet: Architectures and Protocols" (Cambridge University Press, 2011), and co-editor of the book "Traffic Grooming for Optical Networks: Foundations, Techniques and Frontiers" (Springer, 2008).

I have served the research community in multiple roles. I currently serve as Editor-in-Chief of IEEE Networking Letters and as a Steering Committee Member of the OFC conference. I co-founded and served as co-Editor-in Chief of the Elsevier journal Optical Switching and Networking from 2004-2017. In 2016-17 I served as Chair of the IEEE Communications Society Distinguished Lecturer Selection Committee, as Chair of the Optical Networking Technical Committee (ONTC), and as Vice-Chair of the IEEE Communications Society Technical and Educational Activities Council. As an IEEE Distinguished Lecturer in 2010-2011, I completed four tours in the US, Caribbean, and Sri Lanka. I was the TPC or General chair for: NETWORKS 2018, IEEE MASS 2018, IEEE ICC ONS 2017, IEEE ICNP 2014, ICCCN 2013 and 2011, IEEE GLOBECOM ONS 2010, BROADNETS 2007, IEEE LANMAN 2005 and 2004, and IFIP NETWORKING 2004.

My awards include an IEEE GLOBECOM Outstanding Service Award (2010), an IBM Faculty Partnership Award (2007), an ALCOA Foundation Engineering Research Achievement Award (2004, NC State College of Engineering), an Alumni Outstanding Research Award (2003, NC State), an NSF CAREER Award (1997), an Outstanding New Teacher Award (1995, NC State Department of Computer Science). I was inducted in the NC State University Academy of Outstanding Teachers in 2004. I was elected Fellow of the IEEE in 2012 for "contributions in algorithms, protocols, and architectures of optical networks."

Iyad Katib Iyad Katib is an Associate Professor with the Computer Science Department and the current Vice Dean and the College Council Secretary of the Faculty of Computing and Information Technology (FCIT) in King Abdulaziz University (KAU). He is also the Director of KAU High Performance Computing Center.

Iyad received his Ph.D. and MS degrees in Computer Science from University of Missouri-Kansas City in 2011 and 2004, respectively. He received his BS degree in Statistics/Computer Science from King Abdul Aziz University in 1999. His current research interest is on Computer Networking and High Performance Computing.