Polynomial Time Approximations for Multi-Path Routing with Bandwidth and Delay Constraints

Satyajayant Misra, Guoliang Xue and Dejun Yang

Abstract— **In this paper, we study the problem of multi-path routing with bandwidth and delay constraints, which arises in applications for video delivery over bandwidth-limited networks. Assume that each link in the network has a bandwidth and a delay. For a given source-destination pair and a bandwidth requirement, we want to find a set of source to destination paths such that the delay of the longest path is minimized while the aggregated bandwidth of the set of paths meets the bandwidth requirement. This problem is NP-hard, and the state of the art is a maximum flow based heuristic. We first construct a class of examples showing that the maximum flow based heuristic could have very bad performance. We then present a fully polynomial time approximation scheme that can compute a $(1+\epsilon)$-approximation with running time bounded by a polynomial in $1/\epsilon$ and the input size of the instance. Given the NP-hardness of the problem, our approximation scheme is the best possible. We also present numerical results confirming the advantage of our scheme over the current state of the art.**

Keywords: **Quality of service routing, multi-path routing, bandwidth and delay constraints, polynomial time approximation.**

1. Introduction

In recent years, we have witnessed an increasing number of applications on the Internet. Popular applications include IPTV, VoIP, and Video-on-Demand [9], [12], [17], [29], [36]. Driven by these applications, the Internet has also migrated from the best-effort service model to an integrated service model for data, voice, and video applications. As high-bandwidth applications become increasingly popular, supporting guaranteed quality-of-service (QoS) requirements becomes an important challenging research problem.

In traditional data networks, routing is commonly along the shortest path [1], [2]. These protocols are inadequate to support multimedia applications such as on-demand video delivery, which require a certain minimum end-to-end bandwidth together with a bounded start-up delay to offer high user satisfaction [5]. For example, in an MPEG-I video-on-demand system, a streaming bandwidth of about 1.5Mb/s has to be guaranteed [24], [29]. Very often, a single path in a bandwidth limited network such as the Internet or the wireless network, may not be able to support such a high bandwidth requirement. As a result, researchers have proposed to use multi-path routing to support high bandwidth multimedia applications in bandwidth limited networks [6], [7], [25].

We use Fig. 1 to illustrate the need for multi-constrained multi-path routing. The figure shows a network with 6 nodes and 9 links, where the labels on the links indicate the bandwidth-delay pair of each link. For example, the link (s, x) has a bandwidth of 3 and a delay of 1. We want to transmit a video from source node s to destination node t, which requires an end-to-end bandwidth of at least 4 and a start-up delay of at most 2. If we use the *widest shortest path* [2], we will find the path s-t, which has a bandwidth of 1 and a delay of 1. This path cannot support the required end-to-end bandwidth. If we use the *shortest widest path* [27], we will find the path s-v-x-y-t, which has a bandwidth of 5 and a delay of 4. This path can support the required end-to-end bandwidth, but violates the start-up delay requirement. If we use the path s-u-t and the path s-x-t simultaneously, we will have an aggregated bandwidth of 6 and a delay of 2, therefore satisfying both the bandwidth requirement and the delay requirement.

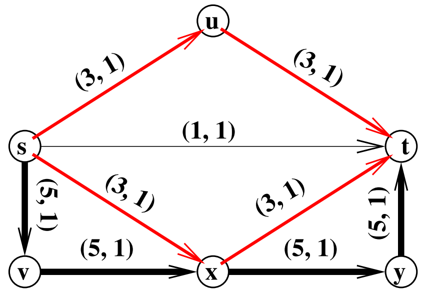


Fig. 1. A graph showing the (b, d) value for each link.

Motivated by observations outlined in the above, Chen *et al.* [6], [7] studied the following optimization problem (formally defined in Section 2, and denoted by OMPBD). Given a source node s, a destination node t, and a bandwidth requirement $\mathbf{B} > 0$, the goal is to find a set $\mathcal{P}$ of s–t paths such that the aggregated bandwidth of the paths is at least $\mathbf{B}$ and the delay of the longest path in the set is minimized. Suppose the delay of the longest path in the set is $\mathbf{D}$, then the users at node t can start viewing the video with the desired quality with a waiting time no more than $\mathbf{D}$. Chen *et al.* proposed a maximum flow based heuristic for OMPBD, which first computes a maximum flow from s to t, and if the flow value is greater than or equal to $\mathbf{B}$, a set of paths is computed by sequentially extracting the shortest path in the flow graph.

In this paper, we first construct a class of examples for which the flow based heuristic computes solutions with delays significantly bigger than those of the optimal solutions. We then present a fully polynomial time approximation scheme for the OMPBD problem, which can compute a $(1+\epsilon)$-approximation with a running time bounded by a polynomial in $1/\epsilon$ and the input size of the instance.

Misra, Xue and Yang are all with the CSE Dept. at Arizona State University, Tempe, AZ 85287. Email: {misra, xue, dejun.yang}@asu.edu. This research was supported in part by ARO grant W911NF-04-1-0385 and NSF grants CNS-0721803 and CCF-0830739. The information reported here does not reflect the position or the policy of the federal government.

The OMPBD problem is closely related to the *quickest flow problem* [4], [16], [23], [31], [32]. Xue *et al.* in [31] first studied the quickest flow problem, where we are given σ units of data, and are asked to find a set of paths that minimizes the total transmission time of the given data. The quickest flow problem has been shown to be polynomial time solvable in [16] and [32], making use of the *maximal dynamic flow* problem studied by Ford and Fulkerson [11]. Note that the quickest flow problem is different from the OMPBD problem. In the former, we want to minimize the total time required to transmit the whole data (viewing the video after the whole video is transmitted to the destination node). In the latter, we want to minimize the waiting time, that is the time required to guarantee that the aggregated end-to-end bandwidth is at least $\mathbf{B}$ (viewing the video as soon as the aggregated end-to-end bandwidth meets the requirement). Therefore the solutions to the quickest flow problem do not solve the OMPBD problem.

The OMPBD problem is also closely related to multi-constrained QoS routing, where the goal is to find a *single path* from the source to the destination which satisfies *multiple QoS requirements* [8], [14], [15], [18], [19], [20], [21], [22], [26], [30], [33], [35]. Actually, the techniques used in the design of our approximation scheme can be viewed as generalizations of the techniques used in QoS routing [20], [33].

In Section 2, we present the system model and define the problems. In Section 3, we show that the flow based heuristic could have very bad performance. In Section 4, we present a pseudo-polynomial time algorithm for an integer version of the problem, which is used as a subroutine in our polynomial time approximation scheme. In Section 5, we present our fully polynomial time approximation scheme for the OMPBD problem. Numerical results are presented in Section 6. We conclude the paper in Section 7.

2. System Model and Problem Definitions

We model the network using a weighted directed graph $G(V, E, b, d)$, where V is the set of n nodes, and E is the set of m links. Each link $e = (u, v) \in E$ is associated with a *bandwidth* $b(e) > 0$ (also denoted by $b(u, v)$) and a *delay* $d(e) \geq 0$ (also denoted by $d(u, v)$). Let s be a *source* node and t a *destination* node. An *s–t path* is a sequence of nodes x_0-x_1-x_2 $\cdots$ x_{l-1}-x_l in V such that $x_0 = s$, $x_l = t$, and (x_{i-1}, x_i) is a link in E for $i = 1, 2, \ldots, l$. We are only interested in *simple paths*–for which the sequence of nodes are distinct. Let p be an s–t path, the *bottleneck bandwidth of path* p is

$$b(p) \triangleq \min_{e \in p} b(p), \tag{2.1}$$

and the *delay of path* p is

$$d(p) \triangleq \sum_{e \in p} d(p). \tag{2.2}$$

Let $\mathcal{P}$ be a set of s–t paths, where each path $p \in \mathcal{P}$ is associated with a *bandwidth allocation* $\beta(p) \leq b(p)$. We say β is a *feasible bandwidth allocation* of $\mathcal{P}$ if for each link $(u, v) \in E$, $\sum_{p \in \mathcal{P}, (u,v) \in p} \beta(p) \leq b(u, v)$. The delay of $\mathcal{P}$, denoted by $d(\mathcal{P})$, is defined to be the delay of the longest path in $\mathcal{P}$, i.e., $d(\mathcal{P}) \triangleq \max_{p \in \mathcal{P}} d(p)$. The *aggregated bandwidth* of $\mathcal{P}$, denoted by $b(\mathcal{P})$, is the sum of all the bandwidth allocations of the paths in $\mathcal{P}$: $b(\mathcal{P}) \triangleq \sum_{p \in \mathcal{P}} \beta(p)$.

The bandwidth and delay constrained multi-path routing problem that we are studying is defined in the following.

Definition 2.1 (MPBD)**:** Let $G = (V, E, b, d)$ be a weighted directed graph with node set V and link set E, where each link $e \in E$ is associated with a bandwidth $b(e) > 0$ and a delay $d(e) \geq 0$. Let s be a source node and t a destination node. Let $\mathbf{B} > 0$ be a bandwidth requirement and $\mathbf{D} > 0$ be a delay tolerance. The **M**ulti**P**ath routing with **B**andwidth and **D**elay constraints problem (denoted as MPBD) seeks a set $\mathcal{P}$ of s–t paths together with a feasible bandwidth allocation β such that

1) $b(\mathcal{P}) \geq \mathbf{B}$, 2) $d(\mathcal{P}) \leq \mathbf{D}$. □

We will use MPBD$(G, b, d, \mathbf{B}, \mathbf{D}, s, t)$ to denote an instance given by graph $G(V, E, b, d)$, bandwidth requirement $\mathbf{B}$, delay tolerance $\mathbf{D}$, source node s, and destination node t. The instance is *feasible* if it has a feasible solution, and *infeasible* otherwise. When we talk about a feasible solution $\mathcal{P}$ for MPBD$(G, b, d, \mathbf{B}, \mathbf{D}, s, t)$, we implicitly assume that there is a corresponding feasible bandwidth allocation β for $\mathcal{P}$.

The MPBD problem defined in the above is a *decision* problem [10]. Its corresponding *optimization* problem (denoted as OMPBD) is defined in the following.

Definition 2.2 (OMPBD)**:** Let $G = (V, E, b, d)$ be a weighted directed graph with node set V and link set E, where each link $e \in E$ is associated with a bandwidth $b(e) > 0$ and a delay $d(e) \geq 0$. Let s be a source node and t be a destination node. Let $\mathbf{B} > 0$ be a bandwidth requirement. The *optimization version* of the MPBD problem seeks for a set $\mathcal{P}$ of s–t paths together with a feasible bandwidth allocation β such that

1) $b(\mathcal{P}) \geq \mathbf{B}$, 2) $d(\mathcal{P})$ is minimized. □

We will use OMPBD$(G, b, d, \mathbf{B}, s, t)$ to denote an instance given by graph $G(V, E, b, d)$, bandwidth requirement $\mathbf{B}$, source node s, and destination node t. We will also use ν^{OMPBD} to denote the delay value of an optimal solution to the given instance.

The goal here is to find a set of s–t paths with an aggregated bandwidth no less than the given bandwidth requirement $\mathbf{B}$, and in the meantime, having the minimum possible delay. The problem arises from applications in on-demand video delivery [6], [7]. It is known to be NP-hard, and the state of the art in solving this problem is a maximum flow based heuristic algorithm [7]. In the next section, we will show that the solutions obtained by the maximum flow based heuristic could be very far from optimal solutions.

3. Flow Based Heuristic and the Complexity of MPBD

The authors of [7] presented a *maximum flow based heuristic* for solving OMPBD. The heuristic first computes a maximum flow from s to t. If the value of the flow is smaller than $\mathbf{B}$, the heuristic correctly claims that the given instance of OMPBD does not have a solution. In the case where the flow value is greater than or equal to $\mathbf{B}$, the heuristic finds a set of s–t paths by repeatedly extracting a shortest path in the flow graph (induced by the links in the maximum flow) until the aggregated bandwidth of the set of paths is greater than or equal to $\mathbf{B}$. The flow based heuristic is described in the following.

Algorithm 1 Flow Based Heuristic for OMPBD

```
1:  Compute a maximum flow $\mathbf{f}$ from $s$ to $t$. Let the value of $\mathbf{f}$ be $M$.
2:  if $M < \mathbf{B}$ then
3:      STOP–The instance has no solution.
4:  end if
5:  Set $G_{\mathcal{P}} = (V_{\mathcal{P}}, E_{\mathcal{P}})$ to be the subgraph of $G$ induced by the links in the maximum flow $\mathbf{f}$. Set $\mathcal{P} := \emptyset$.
6:  while $b(\mathcal{P}) < \mathbf{B}$ do
7:      Compute a shortest path $p$ in $G_{\mathcal{P}}$ and add $p$ to $\mathcal{P}$, with bandwidth allocation $\beta(p)$ set to the bottleneck bandwidth of $p$ in $G_{\mathcal{P}}$.
8:      Subtract $\beta(p)$ from the available bandwidth of each link along $p$ in $G_{\mathcal{P}}$, i.e., $b(e') := b(e') - \beta(p)$, $\forall\ e' \in p$. Remove from $G_{\mathcal{P}}$ all links with zero available bandwidth.
9:  end while
10: Output $\mathcal{P}$.
```

While the flow based heuristic is simple to implement, it is a best effort approach, which does not have any known performance guarantee. As a first step in designing efficient algorithms for OMPBD with provably good performance, we will analyze the performance of the flow based heuristic.

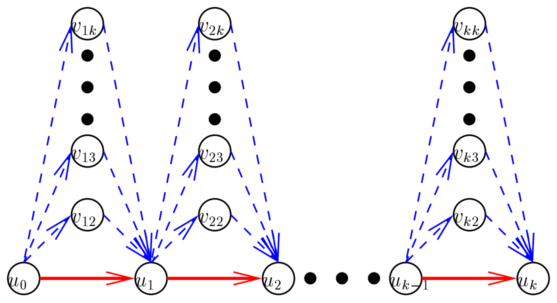


Fig. 2. A graph with $k^2 + 1$ nodes and $2k^2 - k$ links. All links have bandwidths equal to 1. All blue (dashed) links have delays equal to 1. All red (solid) links have delays equal to $\mathbf{L}$.

We use an example to demonstrate that the flow based heuristic can have arbitrarily bad performance. To be specific, we will show that for any given integer $k \geq 4$, there is an instance for which the solution computed by the flow based heuristic has a delay that is at least $\frac{k}{2}$ times the delay of the optimal solution.

Fig. 2 shows a directed graph G with $k^2 + 1$ nodes and $2k^2 - k$ links. All links have bandwidths equal to 1. All blue (dashed) links have delays equal to 1. All red (solid) links have delays equal to $\mathbf{L}$, where $\mathbf{L}$ is a very large number such that $\mathbf{L} \geq 2k$. Let $\mathbf{B} = k$, $s = u_0$, and $t = u_k$. Then the flow based heuristic will compute a set of k paths, where each of the first $k-1$ paths has a delay equal to $2k$, and the last path v_0-v_1-$\cdots$-v_k has a delay equal to $k\mathbf{L}$. One can easily show that the optimal solution for this instance consists of k paths, where each path uses exactly one red (solid) link and $2k-2$ blue (dashed) links. Therefore the delay for the optimal solution is $\mathbf{L}+2k-2$. Since $\mathbf{L} \geq 2k$ (by assumption), we have $\frac{k\mathbf{L}}{\mathbf{L}+2k-2} \geq \frac{k}{2}$. Note that $\frac{k\mathbf{L}}{\mathbf{L}+2k-2}$ is lower bounded by $\frac{k}{2}$ and upper bounded by k, but can be made arbitrarily close to k (by choosing $\mathbf{L}$ large enough). Therefore our example shows that *the solutions computed by the flow based heuristic can have delay values much larger than the delay values of the optimal solutions*. Therefore we should seek for better algorithms.

Before attempting to design better algorithms for this problem, it is helpful to understand the computational complexity of the problem. To this end, we have the following.

Lemma 3.1: Both MPBD and OMPBD are NP-hard. □

This hardness result is stated in [7] without proof. To make the current paper self-contained, without deviating from the main contributions, we present a rigorous hardness proof in Appendix A, which is based on a polynomial time reduction from **Partition** [13] to MPBD.

Given the hardness of the problems, there does not exist any polynomial time optimal algorithm for either MPBD or OMPBD, unless $P = NP$ [13]. Therefore the best one can hope for are polynomial time approximation algorithms or polynomial time approximation schemes. We will present a fully polynomial time approximation scheme (FPTAS) for the OMPBD problem. In other words, for any given constant $\epsilon > 0$, we can compute a $(1+\epsilon)$-approximation in time bounded by a polynomial in $1/\epsilon$ and the input size of the instance. Given the hardness of the problem, our result is the best possible for this problem, unless $P = NP$.

4. Solving the Integer MPBD Problem in Pseudo-Polynomial Time

In the MPBD problem, the link delays may take nonnegative *real* values. In this section, we study a special case of the MPBD problem, denoted by IMPBD, where the link delays and the delay tolerance $\mathbf{D}$ are restricted to *positive integer values*. We present a novel pseudo-polynomial time algorithm for solving IMPBD. Our algorithm is based on a graph transformation technique that has been used by Xue *et al.* in designing approximation algorithms for multi-constrained QoS routing [30].

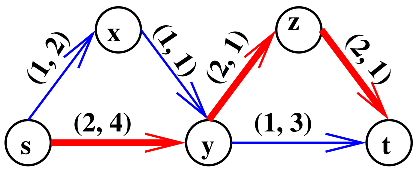


Fig. 3. Graph showing $(b(e), d(e))$ on each link e.

Let an instance of IMPBD be given by graph $G(V, E)$, bandwidth constraint $\mathbf{B} > 0$, delay constraint $\mathbf{D} > 0$, and source-destination node pair (s, t). We construct a *layered graph* $G^{\mathbf{D}} = (V^{\mathbf{D}}, E^{\mathbf{D}})$ from G in the following

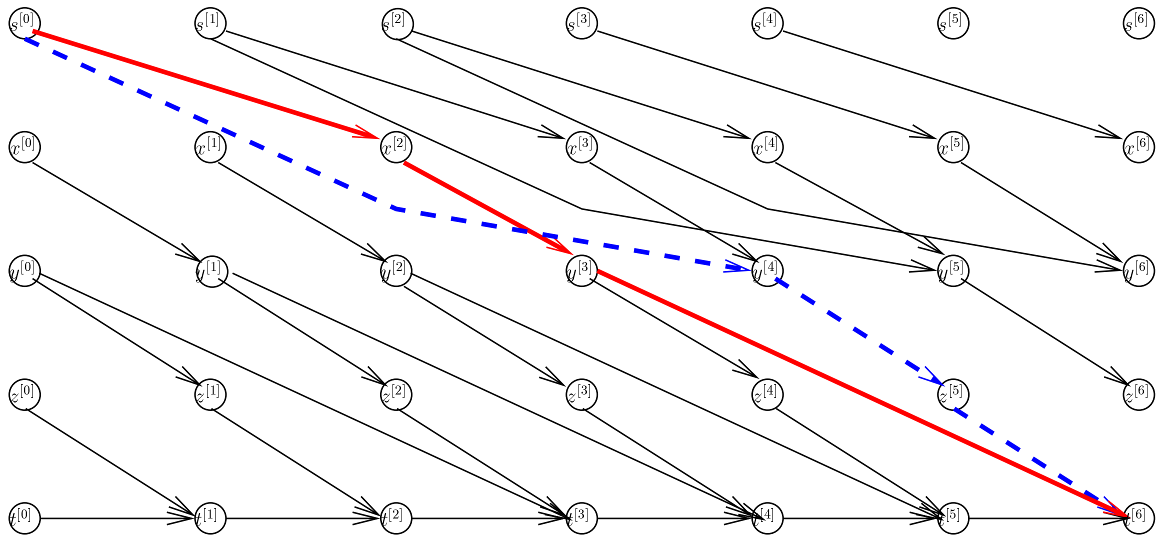


Fig. 4. Graph G^6 corresponding to the graph G in Fig. 3: All links in the form $(s^{[i]}, x^{[i+2]})$ have an aggregated bandwidth of 1. All links in the form $(s^{[i]}, y^{[i+4]})$ have an aggregated bandwidth of 2. All links in the form $(x^{[i]}, y^{[i+1]})$ have an aggregated bandwidth of 1. All links in the form $(y^{[i]}, z^{[i+1]})$ have an aggregated bandwidth of 2. All links in the form $(y^{[i]}, t^{[i+3]})$ have an aggregated bandwidth of 1. All links in the form $(z^{[i]}, t^{[i+1]})$ have an aggregated bandwidth of 2. Each link in the form $(t^{[i]}, t^{[i+1]})$ has a bandwidth of 3.

way. Corresponding to each node $u \in V$, $V^{\mathbf{D}}$ contains $\mathbf{D}+1$ nodes $u^{[0]}, u^{[1]}, \ldots, u^{[\mathbf{D}]}$. Corresponding to each link $(u,v) \in E$, $E^{\mathbf{D}}$ contains $\mathbf{D} - d(u,v) + 1$ links in the form $(u^{[i]}, v^{[i+d(u,v)]}), i = 0, 1, \ldots, \mathbf{D} - d(u,v)$, with an aggregated bandwidth of $b(u,v)$. $E^{\mathbf{D}}$ also contains $\mathbf{D}$ links in the form $(t^{[i]}, t^{[i+1]}), i = 0, 1, \ldots, \mathbf{D}-1$, each with a bandwidth of $\mathbf{B}$. Fig. 3 shows an instance of IMPBD with $\mathbf{B} = 3$ and $\mathbf{D} = 6$. The corresponding layered graph $G^{\mathbf{D}}$ is shown in Fig. 4.

Our pseudo-polynomial time algorithm for solving the IMPBD problem hinges on the layered graph $G^{\mathbf{D}}$. The following observations are helpful in understanding our algorithm.

1) While the original graph G may contain cycles, the layered graph $G^{\mathbf{D}}$ is acyclic: for any link $(u^{[i]}, v^{[j]}) \in G^{\mathbf{D}}$, we always have $i < j$.
2) For any $s^{[0]}$–$t^{[\mathbf{D}]}$ path p in $G^{\mathbf{D}}$, there is a corresponding path p' in G, whose delay is no more than $\mathbf{D}$. For example, p=$(s^{[0]}, x^{[2]}, y^{[3]}, z^{[4]}, t^{[5]}, t^{[6]})$ is a path in the layered graph in Fig. 4. Its corresponding path in G is p'=(s, x, y, z, t), with a delay of 5.
3) Let $\mathbf{f}$ be an $s^{[0]}$–$t^{[\mathbf{D}]}$ flow in $G^{\mathbf{D}}$ with a flow value of $\mathbf{B}$ such that for each link $(u,v) \in E$, the aggregated flow on links $(u^{[i]}, v^{[i+d(u,v)]})$ is no more than $b(u,v)$. Then there is a corresponding s–t flow $\mathbf{f}'$ in G with flow value $\mathbf{B}$.

These observations lead us to study the following *constrained network flow* problem in the layered graph, which is a special case of a Linear Programming (LP) problem:

$$\mathbf{LP}(\mathbf{B},\mathbf{D})\text{: max} \sum_{i=0}^{\mathbf{D}-1} f(t^{[i]}, t^{[i+1]}), \tag{4.1}$$

$$\text{s.t.} \sum_{(s^{[0]},v)\in E^{\mathbf{D}}} f(s^{[0]}, v) = \mathbf{B}, \tag{4.2}$$

$$\sum_{(u,t^{[\mathbf{D}]})\in E^{\mathbf{D}}} f(u, t^{[\mathbf{D}]}) = \mathbf{B}, \tag{4.3}$$

$$\sum_{(x,y)\in E^{\mathbf{D}}} f(x,y) = \sum_{(y,z)\in E^{\mathbf{D}}} f(y,z), \forall y \neq s^{[0]}, t^{[\mathbf{D}]}, \tag{4.4}$$

$$\sum_{i=0}^{\mathbf{D}-d(u,v)} f(u^{[i]}, v^{[i+d(u,v)]}) \le b(u,v), \forall (u,v) \in E, \tag{4.5}$$

$$f(x,y) \ge 0, \forall (x,y) \in E^{\mathbf{D}}. \tag{4.6}$$

The following is a brief description of the formulation of $\mathbf{LP}(\mathbf{B},\mathbf{D})$. The objective function (4.1) forces the flows to use links $(t^{[i]}, t^{[i+1]})$ as much as possible, which ensures that the paths in $G^{\mathbf{D}}$ generated from the computed flow correspond to simple paths in G. The variables are the (real valued) flows defined on the links in $G^{\mathbf{D}}$. They have to be nonnegative, which is ensured by Constraints (4.6). Constraint (4.2) ensures that the net flow out of node $s^{[0]}$ is $\mathbf{B}$. Constraint (4.3) ensures that the net flow into node $t^{[\mathbf{D}]}$ is $\mathbf{B}$. Constraints (4.4) ensure flow conservation at all other nodes of $G^{\mathbf{D}}$. Constraints (4.5) ensure that for each link (u,v) in G, the aggregated flow on all links $(u^{[i]}, v^{[i+d(u,v)]}) \in E^{\mathbf{D}}$ is no more than $b(u,v)$. These constraints make $\mathbf{LP}(\mathbf{B},\mathbf{D})$ different from conventional network flow problems. They are necessary, as all the flows on links $(u^{[i]}, v^{[i+d(u,v)]})$ in $G^{\mathbf{D}}$ correspond to an aggregated flow on link (u,v) in G. Nevertheless, $\mathbf{LP}(\mathbf{B},\mathbf{D})$ is a linear programming problem *whose input size is of the same order of* $\mathbf{D}$ *times the input size of* IMPBD$(G, b, d, \mathbf{B}, \mathbf{D}, s, t)$. We have the following.

Theorem 4.1: An instance of IMPBD given by graph $G(V,E)$, source node s, destination node t, bandwidth constraint $\mathbf{B}$, and delay constraint $\mathbf{D}$ is feasible if and only if there is an $s^{[0]}$–$t^{[\mathbf{D}]}$ flow in $G^{\mathbf{D}} = (V^{\mathbf{D}}, E^{\mathbf{D}})$ with value $\mathbf{B}$ such that for each link $(u,v) \in E$, the sum of all flows on links in the form $(u^{[i]}, v^{[i+d(u,v)]})$ is no more than $b(u,v)$. In addition, a feasible solution to IMPBD can be obtained by the arc-chain decomposition of the computed flow in $G^{\mathbf{D}}$. □

Proof. Assume that the instance of the IMPBD problem is feasible. Then there exists a set of paths $\mathcal{P}$ in G such that $d(\mathcal{P}) \leq \mathbf{D}$ and $\mathbf{B} = b(\mathcal{P}) = \sum_{p\in\mathcal{P}} \beta(p)$, where $\beta(p)$ is the bandwidth allocation for path p such that for each link $e \in E$, we have $\sum_{p\in\mathcal{P}, e\in p} \beta(p) \leq b(e)$. We can define a flow $\mathbf{f}$ in $G^{\mathbf{D}}$ in the following way: for each path $p \in \mathcal{P}$, and each link $(u,v) \in p$, set $f(u^{[i]}, v^{[i+d(u,v)]}) = \sum_{p\in\mathcal{P},(u,v)\in p, d_p(u)=i} \beta(p)$, where $d_p(u)$ denotes the delay from s to u on path p. Then $\mathbf{f}$ is a feasible solution to $\mathbf{LP}(\mathbf{B}, \mathbf{D})$.

Next, we will show that a feasible solution for $\mathbf{LP}(\mathbf{B}, \mathbf{D})$ leads to a feasible solution for IMPBD. Let $\mathbf{f}$ be a feasible solution for $\mathbf{LP}(\mathbf{B}, \mathbf{D})$. We can decompose the flow $\mathbf{f}$ into a finite number of $s^{[0]}$–$t^{[\mathbf{D}]}$ paths using the following arc-chain decomposition. Set $B := \mathbf{B}$. Set $\mathcal{P} := \emptyset$. Let $G^{\mathbf{D}}_B$ be the subgraph of $G^{\mathbf{D}}$ composed of all the links of $G^{\mathbf{D}}$ such that the flow of $\mathbf{f}$ is positive on that link. Compute an $s^{[0]}$–$t^{[\mathbf{D}]}$ path $p^{\mathbf{D}}_B$ in $G^{\mathbf{D}}_B$. Let $b(p^{\mathbf{D}}_B)$ be the bottleneck bandwidth of $p^{\mathbf{D}}_B$ in $G^{\mathbf{D}}_B$. Let p_B be the path in G corresponding to $p^{\mathbf{D}}_B$ (delete cycles if any). Add p_B to $\mathcal{P}$, with bandwidth allocation $\beta(p_B) = b(p^{\mathbf{D}}_B)$. For all the links of $G^{\mathbf{D}}_B$ on $p^{\mathbf{D}}_B$, reduce its residual bandwidth by $b(p^{\mathbf{D}}_B)$, remove the link if its updated residual bandwidth is zero. Replace B by $B - b(p^{\mathbf{D}}_B)$. If $B = 0$, we have obtained a feasible solution $\mathcal{P}$ for IMPBD. If $B > 0$, there must be an $s^{[0]}$–$t^{[\mathbf{D}]}$ path in $G^{\mathbf{D}}_B$, and we repeat the above process. It is clear that $b(\mathcal{P}) = \mathbf{B}$. Since each path in $\mathcal{P}$ corresponds to an $s^{[0]}$–$t^{[\mathbf{D}]}$ path in $G^{\mathbf{D}}$, it must have a delay value bounded by $\mathbf{D}$. Also, the cardinality of $\mathcal{P}$ is bounded by the number of links in $G^{\mathbf{D}}$, as each time we extract a path, at least one link will be deleted from $G^{\mathbf{D}}_B$. ■

For the instance given in Fig. 3, the constrained flow is illustrated in Fig. 4 with thick links, where the path along the red (thick solid) links has a flow value of 1, and the path along the blue (thick dashed) links has a flow value of 2. The red path $s^{[0]}$-$x^{[2]}$-$y^{[3]}$-$t^{[6]}$ in $G^{\mathbf{D}}$ corresponds to path s-x-y-t in G. The blue path $s^{[0]}$-$y^{[4]}$-$z^{[5]}$-$t^{[6]}$ in $G^{\mathbf{D}}$ corresponds to path s-y-z-t in G. It is clear that all these paths generated from the arc-chain decomposition [11] have path delays upper bounded by $\mathbf{D}$. Note that, in general, some of the paths may have path delays smaller than $\mathbf{D}$, due to the links $(t^{[i]}, t^{[i+1]})$ in $G^{\mathbf{D}}$. We summarize our algorithm for solving IMPBD in the following.

Algorithm 2 PSEUDO-IMPBD$(G, b, d, \mathbf{B}, \mathbf{D}, s, t)$

```
1: Formulate the instance $\mathbf{LP}(\mathbf{B}, \mathbf{D})$ and solve it.
2: if $\mathbf{LP}(\mathbf{B}, \mathbf{D})$ is infeasible then
3:     The instance of IMPBD is infeasible. STOP.
4: else
5:     Let $\mathcal{P}$ be the set of $s$–$t$ paths obtained from the
       solution to $\mathbf{LP}(\mathbf{B}, \mathbf{D})$. Output $\mathcal{P}$, together with its
       corresponding bandwidth allocation.
6: end if
```

We have the following theorem.

Theorem 4.2: Algorithm 2 correctly solves the IMPBD problem in $(m^3 \cdot \mathcal{L} \cdot \mathbf{D}^4)$ time, where $\mathcal{L}$ is the input size of the instance of IMPBD. □

Proof. The correctness of the theorem follows from Theorem 4.1. Since $\mathbf{LP}(\mathbf{B}, \mathbf{D})$ has $O(m\mathbf{D})$ variables, and has input size $O(\mathbf{D}\mathcal{L})$ (where $\mathcal{L}$ is the input size of the instance of IMPBD), solving the LP takes $O((m\mathbf{D})^3(\mathbf{D}\mathcal{L}))$ time [34]. This proves the polynomial time complexity. ■

5. A Fully Polynomial Time Approximation Scheme

For a given positive real number θ and an instance OMPBD$(G, b, d, \mathbf{B}, s, t)$ of OMPBD, we construct an auxiliary graph $G^\theta = (V, E, b, d^\theta)$ which is the same as $G = (V, E, b, d)$ except that the link delay function d is changed to d^θ such that for each $e \in E$, $d^\theta(e) = \lfloor d(e) \cdot \theta \rfloor + 1$. This is the scaling and rounding technique used by Lorenz and Raz [20] and by Xue *et al.* [33]. We need the following four lemmas.

Lemma 5.1: Let p be any path in G, we have

$$\theta \cdot d(p) \leq d^\theta(p) \leq \theta \cdot d(p) + n - 1. \qquad (5.1)$$

Proof. From the definition of $d^\theta(e)$, we have $d^\theta(p) = \sum_{e\in p}(\lfloor \theta \cdot d(e) \rfloor + 1)$. Therefore we have (5.1), noting that p has at most $n - 1$ links. ■

Lemma 5.2: Let ν^{OMPBD} be the optimal value of OMPBD$(G, b, d, \mathbf{B}, s, t)$. Let LB and UB be given positive constants such that $\mathsf{LB} \leq \nu^{\mathsf{OMPBD}} \leq \mathsf{UB}$. Let $\epsilon > 0$ be any given constant and let $\theta = \frac{n-1}{\mathsf{LB}\cdot\epsilon}$. Then

1) IMPBD$(G^\theta, b, d^\theta, \mathbf{B}, \lfloor \frac{(n-1)\mathsf{UB}}{\mathsf{LB}\cdot\epsilon} \rfloor + n - 1, s, t)$ is guaranteed to have a feasible solution.
2) $\frac{\mathbf{D}_\theta}{\theta} \leq (1+\epsilon)\cdot\nu^{\mathsf{OMPBD}}$, where $\mathbf{D}_\theta$ is the smallest integer such that IMPBD$(G^\theta, b, d^\theta, \mathbf{B}, \mathbf{D}_\theta, s, t)$ is feasible.
3) A $(1 + \epsilon)$-approximation to OMPBD$(G, b, d, \mathbf{B}, s, t)$ can be found in $O(m^3(\frac{n\cdot\mathsf{UB}}{\epsilon\cdot\mathsf{LB}})^4 \log(\frac{n\cdot\mathsf{UB}}{\epsilon\cdot\mathsf{LB}})\mathcal{L})$ time, by first computing $\mathbf{D}_\theta$, and then solving IMPBD$(G^\theta, b, d^\theta, \mathbf{B}, \mathbf{D}_\theta, s, t)$, where $\mathcal{L}$ is the input size of OMPBD. □

Proof. Let $\mathcal{P}$ be an optimal solution for OMPBD. We will show that $\mathcal{P}$ is also a feasible solution to IMPBD$(G^\theta, b, d^\theta, \mathbf{B}, \lfloor \frac{\mathsf{UB}\cdot(n-1)}{\mathsf{LB}\cdot\epsilon} \rfloor + n - 1, s, t)$. Let p be any path in $\mathcal{P}$. It suffices to show that

$$d^\theta(p) \leq \lfloor \frac{(n-1)\nu^{\mathsf{OMPBD}}}{\mathsf{LB}\cdot\epsilon} \rfloor + n - 1. \qquad (5.2)$$

It follows from Lemma 5.1 that

$$\begin{aligned} d^\theta(p) &\leq \theta \cdot d(p) + n - 1 \leq \theta \cdot \nu^{\mathsf{OMPBD}} + n - 1 \\ &= \frac{(n-1)\nu^{\mathsf{OMPBD}}}{\mathsf{LB}\cdot\epsilon} + n - 1 \end{aligned} \qquad (5.3)$$

Since $d^\theta(p)$ only takes integer values, (5.3) implies (5.2). Part 1) is proved.

Following from (5.2) and the definition of $\mathbf{D}_\theta$, we have

$$\mathbf{D}_\theta \leq \frac{(n-1)\nu^{\mathsf{OMPBD}}}{\mathsf{LB}\cdot\epsilon} + n - 1. \qquad (5.4)$$

This implies

$$\begin{aligned} \frac{\mathbf{D}_\theta}{\theta} &\leq \frac{\mathsf{LB}\cdot\epsilon}{n-1} \cdot \left(\frac{(n-1)\nu^{\mathsf{OMPBD}}}{\mathsf{LB}\cdot\epsilon} + n - 1 \right) \\ &= \nu^{\mathsf{OMPBD}} + \mathsf{LB}\cdot\epsilon \leq \nu^{\mathsf{OMPBD}}(1+\epsilon). \end{aligned} \qquad (5.5)$$

Now we turn to prove Part 3). Following Part 1), $\mathbf{D}_\theta$ can be computed by solving $O(\log(\frac{n\cdot\mathsf{UB}}{\epsilon\cdot\mathsf{LB}}))$ instances of IMPBD (using bisection), each requiring $O(m^3(\frac{n\cdot\mathsf{UB}}{\epsilon\cdot\mathsf{LB}})^4\mathcal{L})$ time (refer to Theorem 4.2). It follows from Part 2), a feasible solution to $\mathsf{IMPBD}(G^\theta, b, d^\theta, \mathbf{B}, \mathbf{D}_\theta, s, t)$ is a $(1+\epsilon)$-approximation to OMPBD. This proves Part 3) of the lemma. ■

It follows from Lemma 5.2 that if we know a pair of lower and upper bounds of ν^{OMPBD} such that the upper bound is within a constant factor of the lower bound, then we can compute a $(1+\epsilon)$-approximation to $\mathsf{OMPBD}(G, b, d, \mathbf{B}, s, t)$ in polynomial time. In the following, we will show how to efficiently compute a lower bound LB and an upper bound UB such that

$$\mathsf{LB} \leq \nu^{\mathsf{OMPBD}} \leq \mathsf{UB} \leq 4 \times \mathsf{LB}. \tag{5.6}$$

To achieve the above result, we use a generalization of the *approximate testing* technique used by Lorenz and Raz [20] and Xue *et al.* [33]. For given real numbers $\Delta > 0$ and $\zeta \in (0, 1]$, we define $\mathsf{TEST}(\Delta, \zeta) = \mathsf{YES}$ if $\mathsf{IMPBD}(G^\theta, b, d^\theta, \mathbf{B}, \lfloor\frac{n-1}{\zeta}\rfloor + n - 1, s, t)$ is feasible (where $\theta = \frac{n-1}{\Delta\cdot\zeta}$) and define $\mathsf{TEST}(\Delta, \zeta) = \mathsf{NO}$ otherwise. We have the following.

Lemma 5.3: Let ν^{OMPBD} be the optimal value of $\mathsf{OMPBD}(G, b, d, \mathbf{B}, s, t)$. Let Δ and ζ be two fixed positive numbers such that $0 < \zeta \leq 1$. Then

- $\mathsf{TEST}(\Delta, \zeta) = \mathsf{YES}$ implies $\nu^{\mathsf{OMPBD}} \leq \Delta \cdot (1 + \zeta)$;
- $\mathsf{TEST}(\Delta, \zeta) = \mathsf{NO}$ implies $\nu^{\mathsf{OMPBD}} > \Delta$. □

Proof. First, we assume that $\mathsf{TEST}(\Delta, \zeta) = \mathsf{YES}$. Let $\mathcal{P}$ be a feasible solution to $\mathsf{IMPBD}(G^\theta, b, d^\theta, \mathbf{B}, \lfloor\frac{n-1}{\zeta}\rfloor + n - 1, s, t)$ and let p be any path in $\mathcal{P}$. Therefore we have

$$\lfloor\frac{n-1}{\zeta}\rfloor + n - 1 \geq d^\theta(p) \geq \theta \cdot d(p), \tag{5.7}$$

where the first inequality follows from feasibility and the second inequality follows from (5.1). This implies

$$d(p) \leq \frac{1}{\theta}(n-1)(\frac{1}{\zeta} + 1) = \Delta(1 + \zeta). \tag{5.8}$$

Since this is true for *any* $p \in \mathcal{P}$, we have proved $\nu^{\mathsf{OMPBD}} \leq \Delta(1 + \zeta)$.

Next, assume $\nu^{\mathsf{OMPBD}} \leq \Delta$. We will prove that $\mathsf{TEST}(\Delta, \zeta)$ must return YES. Let $\mathcal{P}$ be an optimal solution for OMPBD, and p be any path in $\mathcal{P}$. We have (first inequality follows from Lemma 5.1)

$$d^\theta(p) \leq \theta d(p) + n - 1 \leq \theta\nu^{\mathsf{OMPBD}} + n - 1 \tag{5.9}$$

$$= \frac{(n-1)\nu^{\mathsf{OMPBD}}}{\Delta\cdot\zeta} + n - 1 \tag{5.10}$$

$$\leq \frac{n-1}{\zeta} + n - 1. \tag{5.11}$$

Since $d^\theta(p)$ only takes integer values, we have

$$d^\theta(p) \leq \lfloor\frac{n-1}{\zeta}\rfloor + n - 1. \tag{5.12}$$

This shows that $\mathcal{P}$ is a feasible solution to $\mathsf{IMPBD}(G^\theta, b, d^\theta, \mathbf{B}, \lfloor\frac{n-1}{\zeta}\rfloor + n - 1, s, t)$. Therefore $\mathsf{TEST}(\Delta, \zeta)$ must return YES. ■

Assume that we have a pair of lower and upper bounds LB and UB such that $\mathsf{UB} \geq 2(1 + \zeta)\mathsf{LB}$. Set $\Delta = \sqrt{\frac{\mathsf{LB}\times\mathsf{UB}}{1+\zeta}}$. We can run $\mathsf{TEST}(\Delta, \zeta)$. If $\mathsf{TEST}(\Delta, \zeta) = \mathsf{YES}$, it follows from Lemma 5.3 that $\nu^{\mathsf{OMPBD}} \leq \Delta \times (1 + \zeta)$. Therefore we can decrease UB to $\Delta \times (1 + \zeta)$. If $\mathsf{TEST}(\Delta, \zeta) = \mathsf{NO}$, it follows from Lemma 5.3 that $\nu^{\mathsf{OMPBD}} > \Delta$. Therefore we can increase LB to Δ. In either of the two cases, $\frac{\mathsf{UB}}{\mathsf{LB}}$ is reduced to $\sqrt{(1 + \zeta)\cdot\frac{\mathsf{UB}}{\mathsf{LB}}}$. Assume that for the initial pair of lower and upper bounds, we have $r = \frac{\mathsf{UB}}{\mathsf{LB}}$. Then we will have $\mathsf{UB} \leq 2(1 + \zeta)\mathsf{LB}$ after $\log\log r$ iterations. If we choose $\zeta = 1$, this would lead to a pair of lower and upper bounds satisfying inequality (5.6).

The remaining question is how to compute the initial pair of lower and upper bounds such that $\frac{\mathsf{UB}}{\mathsf{LB}}$ is not very big. We will show that this can be computed efficiently as well.

Since there are m links, the number of distinct link delay values is no more than m. Let these distinct link delay values be $d_1 < d_2 < \cdots < d_k$. We will use these values to define a set of subgraphs of G, known as *delay bottleneck subgraphs*. For each $i = 1, 2, \ldots, k$, we will use $G(d_i)$ to denote the subgraph of G which contains all nodes of G, but only those links of G whose delay values are no more than d_i. As a technical convention, we assume $d_0 = 0$ and use $G(d_0)$ to denote the subgraph of G which contains no link at all. We can prove the following fact.

Lemma 5.4: If there is no s–t flow of value $\mathbf{B}$ in $G(d_k)$, then the instance $\mathsf{OMPBD}(G, b, d, \mathbf{B}, s, t)$ has no solution. Let i' be the integer such that there is no s–t flow of value $\mathbf{B}$ in $G(d_{i'-1})$, but there is an s–t flow of value $\mathbf{B}$ in $G(d_{i'})$. Then $d_{i'} \leq \nu^{\mathsf{OMPBD}} \leq (n-1)d_{i'}$, where ν^{OMPBD} is the delay of an optimal solution for the instance $\mathsf{OMPBD}(G, b, d, \mathbf{B}, s, t)$. Therefore $\mathsf{LB} = d_{i'}$ and $\mathsf{UB} = (n-1)d_{i'}$ form a pair of lower and upper bounds of ν^{OMPBD}. □

Proof. Assume that there is no s–t flow of value $\mathbf{B}$ in $G(d_k)$. Then clearly the instance $\mathsf{OMPBD}(G, b, d, \mathbf{B}, s, t)$ has no feasible solution, as $G(d_k) = G$. Next we will concentrate on the case where the instance has a feasible solution. By the choice of i', the integer i' is unique such that $1 \leq i' \leq k$. Furthermore, every s–t flow (in G) of value $\mathbf{B}$ must contain at least one link with a delay greater than or equal to $d_{i'}$. Therefore $d_{i'}$ is a lower bound of ν^{OMPBD}.

Also from the choice of i', we know that there is an s–t flow of value $\mathbf{B}$, which uses only links of delay $d_{i'}$ or smaller. From this flow, we can get a set $\mathcal{P}$ of s–t paths such that $b(\mathcal{P}) = \mathbf{B}$. Since each path in $\mathcal{P}$ has at most $n-1$ links, and each link has a delay at most $d_{i'}$, therefore $d(\mathcal{P}) \leq (n-1) \times d_{i'}$. ■

We have now presented three key building blocks for designing our FPTAS for OMPBD. Lemma 5.4 shows how to compute an initial pair of lower and upper bounds efficiently. Lemma 5.3 shows how to refine a pair of lower and upper bounds efficiently. Lemma 5.2 shows how to compute a $(1+\epsilon)$-approximation once a good pair of lower and upper bounds are computed. Our FPTAS is presented in the following.

Algorithm 3 FPTAS-OMPBD$(G, b, d, \mathbf{B}, s, t)$

```
1: As in Lemma 5.4, find the smallest integer $i'$ such that there is no $s$–$t$ flow of value $\mathbf{B}$ in $G(d_{i'-1})$, but there is an $s$–$t$ flow of value $\mathbf{B}$ in $G(d_{i'})$. Set $\mathsf{LB}^{[0]} := d_{i'}$, $\mathsf{UB}^{[0]} := (n-1) \cdot d_{i'}$, Set $i := 0$.
2: while $\mathsf{UB}^{[i]} \geq 4 \times \mathsf{LB}^{[i]}$ do
3:   Let $\Delta := \sqrt{\frac{\mathsf{LB}^{[i]} \times \mathsf{UB}^{[i]}}{2}}$.
4:   if TEST$(\Delta, 1)$ = YES then
5:     $\mathsf{UB}^{[i+1]} := 2 \cdot \Delta$,  $\mathsf{LB}^{[i+1]} := \mathsf{LB}^{[i]}$.
6:   else
7:     $\mathsf{UB}^{[i+1]} := \mathsf{UB}^{[i]}$,  $\mathsf{LB}^{[i+1]} := \Delta$.
8:   end if
9:   $i := i + 1$.
10: end while
11: Set LB := $\mathsf{LB}^{[i]}$, UB := $\mathsf{UB}^{[i]}$.
12: Set $\theta := \frac{n-1}{\mathsf{LB} \cdot \epsilon}$. Compute $\mathbf{D}_\theta$ as in Lemma 5.2. Apply Algorithm 2 to compute a feasible solution $\mathcal{P}$ to IMPBD$(G^\theta, b, d^\theta, \mathbf{B}, \mathbf{D}_\theta, s, t)$. OUTPUT $\mathcal{P}$ as a $(1+\epsilon)$-approximation of OMPBD.
```

The basic idea of FPTAS-OMPBD is as follows. First (in Line 1), as in Lemma 5.4, we compute an initial lower bound LB and upper bound UB of ν^{OMPBD} such that

$$\mathsf{LB} \leq \nu^{\mathsf{OMPBD}} \leq \mathsf{UB} = (n-1) \cdot \mathsf{LB}. \qquad (5.13)$$

Then (in Lines 2-10), using approximate testing (with ζ set to 1), we refine the pair of lower and upper bounds (in no more than $\log\log n$ iterations) so that inequality (5.11) is satisfied. Finally (in Lines 11-12), for the given constant $\epsilon > 0$, we use the approach outlined in Lemma 5.2 to compute a $(1+\epsilon)$-approximation for OMPBD$(G, b, d, \mathbf{B}, s, t)$.

Theorem 5.1: For any given $\epsilon > 0$, Algorithm 3 computes a set $\mathcal{P}$ of s–t paths that is a $(1+\epsilon)$-approximation of OMPBD$(G, b, d, \mathbf{B})$ in time bounded by $O(m^3 n^4 \mathcal{L} \left(\log\log n + (\frac{1}{\epsilon})^4 \log \frac{n}{\epsilon}\right))$, where $\mathcal{L}$ is the input size of the OMPBD instance. □

Proof. In Line 1 of the algorithm, we find the initial pair of lower and upper bounds as in Lemma 5.4. This can be accomplished by solving $O(\log m) = O(\log n)$ maximum flow problems on bottleneck graphs of G, after the links are sorted according to their delay values in $O(m \log n)$ time. Since the maximum flow problem requires $O(n^3)$ time, the time required by Line 1 is $O(n^3 \log n)$. It follows from Lemma 5.4, at the end of Line 1, we have

$$\mathsf{LB}^{[0]} \leq \nu^{\mathsf{OMPBD}} \leq \mathsf{UB}^{[0]} = (n-1) \cdot \mathsf{LB}^{[0]}. \qquad (5.14)$$

In Lines 2 through 10, we use approximate testing to refine the pair of lower and upper bounds, with ζ set to 1. By the choice of Δ in Line 3 of the algorithm, we have $\frac{\Delta}{\mathsf{LB}^{[i]}} = \frac{\mathsf{UB}^{[i]}}{\Delta} = \sqrt{2\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}}}$ throughout the while-loop in Lines 2-10. It follows from Lemma 5.3 that TEST$(\Delta, 1)$ = YES implies that $2 \cdot \Delta$ is an upper bound of ν^{OMPBD} and TEST$(\Delta, 1)$ = NO implies that Δ is a lower bound of ν^{OMPBD}. Therefore we have

$$\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}} = 2^{\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^i}} \cdot \left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}} \leq 2 \cdot \left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}}, \ \forall i \geq 1. \quad (5.15)$$

Therefore the while-loop in Lines 2-10 is executed at most $O(\log\log \frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}) = O(\log\log n)$ times. Since each execution of TEST$(\Delta, 1)$ solves an LP problem with $O(mn)$ variables, it requires $O((mn)^3 (n\mathcal{L}))$ time in the worst case. Therefore the time complexity of all executions of this while-loop is bounded by $O(m^3 n^4 \mathcal{L} \log\log n)$. It follows from Lemma 5.2 that our algorithm finds a $(1+\epsilon)$-approximation in Lines 11-12, in time bounded by $O(m^3 (\frac{1}{\epsilon})^4 \mathcal{L} \log \frac{n}{\epsilon})$. This proves the theorem. ■

6. Numerical Results

In this section, we present the numerical results to confirm the effectiveness of our FPTAS. We implemented FPTAS-OMPBD which is denoted as OMPBD in the figures, and the flow based heuristic, which is denoted as FBH. We compare the results of OMPBD with those obtained from FBH. All our simulation runs were performed on a 2.8 GHz Linux PC with 1G bytes of memory.

We ran both OMPBD and FBH on well-known Internet topologies, as well as randomly generated topologies. The well-known Internet topologies used are ItalianNet (which has 33 nodes and 67 links), ArpaNet (which has 20 nodes and 32 links), and NSFNET (which has 14 nodes and 21 links) [33]. Each link in the graph has two weights, the first represents the bandwidth, and the second represents the delay. Similar to [33], the bandwidth and the delay values were generated uniformly in the interval $(0, 20]$.

For the random graphs, we used the widely used BRITE graph generator [3]. We used the default values of $\alpha = 0.15$ and $\beta = 0.2$ for the Waxman model [28] in BRITE. For the setup, the nodes were randomly generated in a square field of size 1000×1000 square meters. According to the Waxman model, if d_{uv} denotes the Euclidean distance between two nodes u and v, the probability of having a bidirectional link (u, v) connecting u and v is given by $\beta \times e^{\frac{d_{uv}}{\alpha \cdot L}}$, where e is the base for natural logarithms, L is the maximum distance between any two nodes. We have used 8 different network sizes, where the number of nodes are $40, 60, 80, 100, 120, 140, 160$, and 180, respectively.

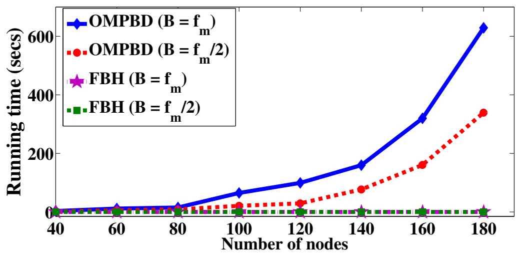
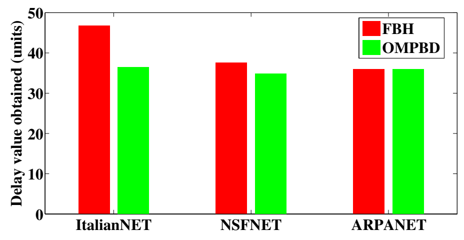
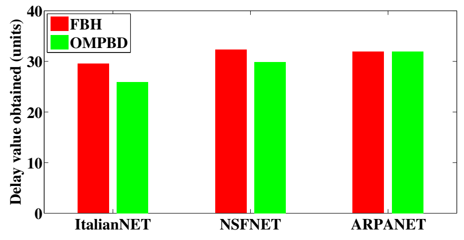


Fig. 5. Running time of OMPBD and the heuristic for two different $\mathbf{B}$ values

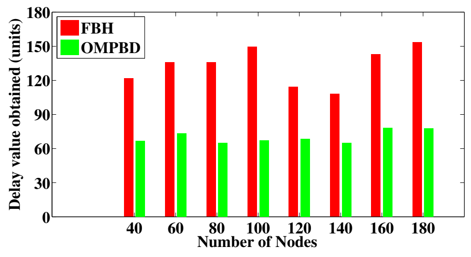
For this simulation study, we set $\epsilon = 0.5$ in the FPTAS. For each network configuration, we averaged the results of our



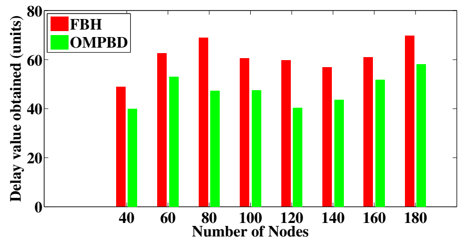
(a) Delay values for Internet topologies, $\mathcal{B} = f_m$



(b) Delay values for Internet topologies, $\mathcal{B} = f_m/2$



(c) Delay values for random topologies, $\mathcal{B} = f_m$



(d) Delay values for random topologies, $\mathcal{B} = f_m/2$

Fig. 6. Comparison of the delay values obtained by OMPBD and FBH for $\mathcal{B} = \{f_m, f_m/2\}$

simulation runs over 10 graph instances. For each instance, the source and destination pairs were chosen randomly, while the value for $\mathbf{B}$ was either set to the maximum flow (f_m) between the source and the destination or half of the maximum flow. The reason for choosing $\mathbf{B} = \{f_m, f_m/2\}$ is to test the algorithms with high bandwidth requirements and to see how the delay value changes with a decrease in the bandwidth requirement. Fig. 5 illustrates the running time of OMPBD and FBH for the random graph topologies for two different values of $\mathbf{B}$. As expected, OMPBD required longer running time than FBH. However, the growth of the running time of OMPBD is polynomial in the network size. We do not show the running time for the Internet based graphs as they are much smaller than the random graphs and both OMPBD and FBH took very little time to obtain the solutions.

Fig. 6(a) and Fig. 6(b) show the delay values obtained by OMPBD and FBH for the Internet topologies for $\mathbf{B} = f_m$ and $\mathbf{B} = f_m/2$, respectively. The delay values of the solutions found by OMPBD are always less than or equal to the delay values of the solutions found by FBH. However, the difference is not large. This is due to the small size of the networks, leading to few number of paths.

Fig. 6(c) and Fig. 6(d) show the delay values obtained by OMPBD and FBH for random topologies generated by BRITE, for $\mathbf{B} = f_m$ and $\mathbf{B} = f_m/2$, respectively. We observe that OMPBD outperforms FBH in all cases. For the case of higher bandwidth requirement (Fig. 6(c)), solutions obtained by OMPBD have delay values that are about half of those obtained by FBH. For the case of lower bandwidth requirement (Fig. 6(d)), the difference is less significant. This is as expected, as OMPBD is designed for *bandwidth-limited networks*. When the network is not bandwidth-limited, a single shortest path could satisfy the desired bandwidth requirement. In such cases, the flow based heuristic may find solutions close to optimal.

To summarize, our numerical results confirm our theoretical analysis, showing the advantage of OMPBD over FBH: (1) The running time of OMPBD scales polynomially with the network size. (2) In almost all cases, the solutions obtained by OMPBD are better than the solutions obtained by FBH. Given the theoretical analysis of the performance and time complexity of our FPTAS, we believe the presented numerical results are representative.

7. Conclusions

In this paper, we have made several important contributions to multi-path routing with bandwidth and delay constraints. First, we have constructed a class of examples to show that the popular flow based heuristic may have very bad performance. Then, we have presented a fully polynomial time approximation scheme for this problem, using a novel linear programming based pseudo-polynomial time algorithm for a restricted version of the problem. Since the problem is NP-hard, our approximation scheme is the best possible. Numerical results confirm the advantage of our FPTAS over the flow based heuristic.

Acknowledgment

We thank David Curtiss and Samuel Rasumssen for their help with the implementation of the algorithms.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi; Quality of service based routing: A performance perspective; *ACM SIGCOMM'1998*, pp. 17-28.

[2] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda; QoS routing mechanisms and OSPF extensions; *IETF RFC 2676*, 1999.

[3] BRITE; http://www.cs.bu.edu/brite.

[4] R.E. Burkard, K. Dlaska and B. Klinz; The quickest flow problem; *Math. Methods of Operations Research*, Vol. 37 (1993), pp. 31-58.

[5] J. Chen, R. Sundaram, M. Marathe and R. Rajaraman; The confluent capacity of the Internet: congestion vs. dilation; *IEEE ICDCS'06*, pp. 5.

[6] J.-C. Chen and S.-H. Chan; Multipath routing for video unicast over bandwidth-limited networks; *IEEE Globecom'2001*, pp. 1963-1997.

[7] J.-C. Chen, S.-H. Chan and V. Li; Multipath routing for video delivery over bandwidth-limited networks; *IEEE Journal on Selected Areas in Communications*, Vol. 22(2004), pp. 1920-1932.

[8] S. Chen and K. Nahrstedt; On finding multi-constrained paths; *IEEE ICC'1998*, pp. 874-879.

[9] I. Cheng, A. Basu, Y. Zhang, S. Tripathi; QoS specification and adaptive bandwidth monitoring for multimedia delivery; *EUROCON'2001*, pp. 483-486.

[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein; *Introduction to Algorithms*, second edition, MIT Press and McGraw-Hill, 2001.

[11] L.R. Ford and D.R. Fulkerson; *Flows in Networks*, Princeton University Press, 1962.

[12] L. Gao and D. Towsley; Supplying instantaneous video-on-demand services using controlledmulticast; *IEEE ICMCS'1999*, pp. 117-121.

[13] M.R. Garey and D.S. Johnson; *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[14] A. Goel, K.G. Ramakrishnan, D. Kataria and D. Logothetis; Efficient computation of delay-sensitive routes from one source to all destinations; *IEEE INFOCOM'2001*, pp. 854-858.

[15] R. Hassin; Approximation schemes for the restricted shortest path problem; *Mathematics of Operations Research*, Vol. 17(1992), pp. 36-42.

[16] B. Hoppe E. Tardos; The quickest transshipment problem; *SIAM/ACM SODA'1995*, pp. 512-521.

[17] D. Jurca and P. Frossard; Media flow rate allocation in multipath networks; *IEEE Transactions on Multimedia*, Vol. 9(2006), pp. 1227-1240.

[18] T. Korkmaz and M. Krunz; Bandwidth-delay constrained path selection under inaccurate state information; *IEEE/ACM Transactions on Networking*, Vol. 11(2003), pp. 384-398.

[19] F.A. Kuipers, A. Orda, D. Raz, and P. Van Mieghem; A comparison of exact and ϵ-approximation algorithms for constrained routing; *IFIP Networking'2006*, pp. 197-208.

[20] D.H. Lorenz and D. Raz; A simple efficient approximation scheme for the restricted shortest path problem; *Operations Research Letters*, Vol. 28(2001), pp. 213-219.

[21] Q. Ma, P. Steenkiste; On path selection for traffic with bandwidth guarantees; *IEEE ICNP'1997*, pp. 191–202.

[22] A. Orda and A. Sprintson; Efficient algorithms for computing disjoint QoS paths; *IEEE INFOCOM'2004*, pp. 727-738.

[23] N.S.V. Rao, S. G. Batsell; QoS routing via multiple paths using bandwidth reservation; *IEEE INFOCOM'1998*, pp. 11-18.

[24] D. Saparilla and K.W. Ross; Optimal streaming of layered video; *IEEE INFOCOM'2000*, pp. 737-746.

[25] H. Suzuki and F.A. Tobagi; Fast bandwidth reservation scheme with multilink and multipath routing in ATM networks; *IEEE INFOCOM'1992*, pp. 2233-2240.

[26] P. Van Mieghem and F.A. Kuipers, Concepts of exact QoS routing algorithms; *IEEE/ACM Transactions on Networking*, Vol. 12(2004), pp. 851-864.

[27] Z. Wang and J. Crowcroft; Quality-of-service routing for supporting multimedia applications; *IEEE Journal on Selected Areas in Communications*, Vol. 14(1996), pp. 1228-1234.

[28] B.M. Waxman; Routing of multipoint connections; *IEEE Journal on Selected Areas in Communications*, Vol. 6(1988), pp. 1617-1622.

[29] D. Wu, Y.T. Hou, W. Zhu, Y.Q. Zhang, and J.M. Peha; Streaming video over the Internet: approaches and directions; *IEEE Transactions on CAS for Video Technology*, Vol. 11(2001), pp. 282-300.

[30] G. Xue, A. Sen, W. Zhang, J. Tang and K. Thulasiraman; Finding a path subject to many additive QoS constraints; *IEEE/ACM Transactions on Networking*, Vol. 15(2007), pp. 201-211.

[31] G. Xue, S.Z. Sun and J.B. Rosen; Minimum time message transmission in networks; *IEEE ICCI'1992*, pp. 22-25.

[32] G. Xue, S.Z. Sun and and J.B. Rosen; Fast data transmission and maximal dynamic flow; *Information Processing Letters*, Vol. 66(1998), pp. 127-132.

[33] G. Xue, W. Zhang, J. Tang and K. Thulasiraman; Approximation algorithms for multi-constrained QoS routing; *IEEE/ACM Transactions on Networking*, Vol. 16(2008), pp. 656-669.

[34] Y. Ye; An $O(n^3\mathcal{L})$ potential reduction algorithm for linear programming; *Mathematica Programming*, Vol. 50(1991), pp. 239-258.

[35] X. Yuan; Heuristic algorithms for multiconstrained quality-of-service routing; *IEEE/ACM Transactions on Networking*, Vol. 10(2002), pp. 244-256.

[36] Q. Zhang, W. Zhu, and Y.Q. Zhang; Resource allocation for multimedia streaming over the Internet; *IEEE Transactions on Multimedia*, Vol. 3(2001), pp. 339-355.

Appendix

A. Hardness of MPBD and OMPBD

In [7], the OMPBD problem is stated to be NP-hard. However, no proof was provided there. In the following, we will show that MPBD is NP-hard [13], by outlining a reduction from **Partition** to MPBD. Note that the hardness of MPBD implies the hardness of OMPBD.

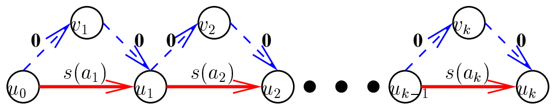


Fig. 7. Reduction from **Partition** to MPBD.

An instance of **Partition** is given by a finite set A, where each $a \in A$ is associated with a positive integer $s(a)$, known as the size of a. It asks for the existence of a subset A' of A such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a)$. This problem is known to be NP-hard [13].

We outline a reduction from **Partition** to MPBD in the following, with the help of Fig. 7. Let an instance $\mathcal{I}_1$ of **Partition** be given by $A = \{a_1, a_2, \ldots, a_k\}$ and size function s. We construct an instance $\mathcal{I}_2$ of MPBD in the following way. The set of nodes of graph $G(V, E)$ is given by $V = \{u_0, v_1, u_1, v_2, u_2, \ldots, u_{k-1}, v_k, u_k\}$. The set of directed links are (u_{i-1}, u_i) (solid red in Fig. 7) with delays equal to $s(a_i)$, $i = 1, 2, \ldots, k$ and (u_{i-1}, v_i) and (v_i, u_i) (dashed blue in Fig. 7) with a delay equal to 0, $i = 1, 2, \ldots, k$. All links have bandwidth equal to 1. The graph is shown in Fig. 7. Set $\mathbf{B} = 2$ and $\mathbf{D} = \sum_{1\le i\le k} s(a_i)/2$.

Clearly, $\mathcal{I}_2$ can be constructed from $\mathcal{I}_1$ in polynomial time. Suppose $\mathcal{I}_1$ has a feasible partition $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a)$. Then $\mathcal{I}_2$ has a feasible solution where one path uses the solid links corresponding to elements $a \in A'$, and the other path uses the solid links corresponding to elements $a \in A \setminus A'$. Conversely, suppose $\mathcal{I}_2$ has a feasible solution. It must have two paths p_1 and p_2, as each link has a bandwidth of 1. Let A' be the set of elements whose corresponding solid links are used by p_1. Then A' form a feasible partition for $\mathcal{I}_1$. This proves that MPBD is NP-hard, which also implies that OMPBD is NP-hard.