Bandwidth Scheduling with Multiple Fixed Node-Disjoint Paths in High-Performance Networks

Aiqin Hou[1], Chase Q. Wu[1,2(✉)], Dingyi Fang[1], Yongqiang Wang[1], and Meng Wang[1]

[1] School of Information Science and Technology, Northwest University, Xi'an 710127, Shaanxi, China
{houaiqin,dyf,yqwang}@nwu.edu.cn, xidawm@stumail.nwu.edu.cn
[2] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
chase.wu@njit.edu

Abstract. Many large-scale applications generate large volumes of data that must be transferred over high-performance networks for various storage or analysis purposes. Such requirements call for a fast bandwidth scheduling solution to discover feasible and efficient reservation options in a time-varying network environment. We investigate a bandwidth scheduling problem with two node-disjoint paths, referred to as BS-2NDP, to support big data transfer. In BS-2NDP, we further consider two different types of paths: (i) two fixed paths of fixed bandwidth (2FPFB), and (ii) two fixed paths of variable bandwidth (2FPVB). We show that both 2FPFB and 2FPVB are NP-complete, and then design heuristic approach-based solutions, which are implemented and tested in both simulated and real-life networks. Extensive results illustrate that the proposed heuristics achieve a close-to-optimal performance in small-scale networks, and significantly outperform other heuristic approaches in large-scale networks.

Keywords: High-performance networks · Bandwidth scheduling · Node-disjoint paths

1 Introduction

Many large-scale applications in various domains require fast and reliable transfer of big data for remote operations, which has gone beyond the capability of traditional shared IP networks. In recent years, high-performance networks (HPNs) with the capability of bandwidth reservation have emerged as an effective solution and their significance has been well recognized in broad science and network research communities.

As the central function unit of a generalized control plane for provisioning dedicated channels in HPNs, the bandwidth scheduler computes appropriate

J.-H. Lee and S. Pack (Eds.): QShine 2016, LNICST 199, pp. 86–96, 2017.
DOI: 10.1007/978-3-319-60717-7_9

network paths and allocates link bandwidths to meet specific user requests based on network topology and bandwidth availability. To meet the unprecedented requirement of big data movement, it is a natural extension from single-path to multi-path transfer, which is generally more effective in terms of throughput, robustness, load balance, and congestion reduction. However, multi-path routing also brings additional complexity and overhead to the network's control and data planes [1].

The complexity of multi-path routing varies depending on the type and number of constraints, and many of these routing problems are NP-complete. Several studies have shown that the Multiple Constrained Path (MCP) problems are generally NP-complete and hence are not solvable in polynomial time [2,3]. Furthermore, finding disjoint paths with a single constraint is also an NP-hard problem [4–6]. Multiple paths usually have an additional constraint to be link-disjoint or node-disjoint. Node-disjoint paths are usually harder to find but provide more robustness in case of node failures. Traditional one-path routing with path constraint/objective is NP-complete, while the problem of two-path routing that considers reliability is strongly NP-hard [7].

In this paper, we investigate a problem of Bandwidth Scheduling with Two Node-Disjoint Paths (BS-2NDP) to support big data transfer. In BS-2NDP, we further consider two different types of paths: (i) two fixed paths of fixed bandwidth (2FPFB), and (ii) two fixed paths of variable bandwidth (2FPVB). We prove that both 2FPFB and 2FPVB are NP-complete, and design heuristic approach-based solutions, which are implemented and tested in both simulated and real-life networks. Extensive results illustrate that the proposed heuristics achieve a close-to-optimal performance in small-scale networks, and significantly outperform other heuristic approaches in large-scale networks.

2 Related Work

Several studies addressed the problem of finding maximum combined bandwidth of disjoint paths. In [5], Shen et al. considered two problems where the notions of multi-path, disjoint path, and widest path are combined. They proved that both problems are NP-complete and provided each an exact solution using ILP and a heuristic solutions. In [6], Dahshan proved the Maximum-Bandwidth Node-Disjoint Paths (MBNDP) problem to be NP-complete, which is essentially an MCP problem with two constraints: the first constraint is for the two paths to be node-disjoint, and the second constraint is to maximize the sum of the bandwidths of the two paths. This problem is also shown to be NP-complete in [5]. A Max-Limit Bandwidth Disjoint Path (MLBDP) algorithm was proposed in [6], which labels two concurrent paths as R (red path) and B (blue path). The algorithm finds the R path with the maximum bandwidth, together with a node-disjoint path B with bandwidth no less than a specified limit. In [8], the problem is to obtain the λ-edge-disjoint-path-set (λDP/B) that has a maximum total bandwidth for $\lambda > 1$, and a polynomial-time heuristic, Maximum Bandwidth Algorithm (MBA), is designed to compute a path with the maximum bandwidth.

In [9], a distributed distance-vector algorithm is used to find multiple node-disjoint paths in a computer network. In [10], a two-disjoint multi-path routing strategy using colored trees is proposed.

Different from most of the aforementioned work that considers static networks, we consider the problem of finding multiple node-disjoint paths with maximum bandwidth in HPNs with time-varying link bandwidth availability.

3 Problem Formulation

3.1 Network Model

The HPN is typically modeled as a network graph $G(V, E)$ with n nodes and m links. Each link $l \in E$ in the network is associated with a time-varying residual bandwidth $b_l[i]$, which is denoted by a 3-tuple of time-bandwidth (TB) $(t_l[i], t_l[i+1], b_l[i])$ for the time interval $(t_l[i], t_l[i+1])$, $i = 0, 1, 2, \ldots, T_l - 1$, where T_l denotes the number of time-slots on link l.

We build an Aggregated TB (ATB) list by combining and storing the TB lists of all individual links in their intersected time-slots. We create a set of new time slots by combining the time slots of all links, and then map the residual bandwidths of each link to the ATB list in each new time slot. We denote the ATB list as $(t[0], t[1], b_0[0], b_1[0], \ldots, b_{m-1}[0]), \ldots, (t[T-1], t[T], b_0[T-1], b_1[T-1], \ldots, b_{m-1}[T-1])$, where T is the number of intersected time slots after aggregating all TB lists. Note that time slot i corresponds to time interval $(t[i], t[i+1])$.

3.2 Problem Formulation

Definition 1. *BS-2NDP: Given an HPN graph $G(V, E)$ with an ATB list and a user request specifying source node v_s, destination node v_d, and data size δ, we wish to find two node-disjoint paths to move data of size δ from node v_s to node v_d such that the data transfer end time t_{end} is minimized.*

Without loss of generality, we suppose that the data transfer may start at time point $t[0] = 0$. In view of the simplicity and popularity of fixed-path routing in practice, we consider two commonly used service models based on a fixed path in BS-2NDP, i.e. 2FPFB and 2FPVB. On the other hand, the service models based on a variable path would require some additional support from the control plane and the network infrastructure for path switching, and hence are not used as commonly in real-life HPNs.

Definition 2. ***2FPFB*** *(Two Fixed Paths with Fixed Bandwidth): Given the network model and user request in BS-2NDP, the goal is to find two fixed node-disjoint paths from v_s to v_d, each of which has a fixed bandwidth (i.e. the bandwidth is invariable during the entire period of data transfer for the given request), such that the data transfer end time is minimized.*

Definition 3. ***2FPVB*** *(Two Fixed Paths with Variable Bandwidth): Given the network model and user request in BS-2NDP, the goal is to find two fixed node-disjoint paths from v_s to v_d, each of which may use varying bandwidths across different time slots during the data transfer, such that the data transfer end time is minimized.*

3.3 Complexity Analysis

Compared with the work in [4–7], which considers static networks (with constant link bandwidths), the BS-2NDP problem is more general as the bandwidth availability of each link in HPNs is time-varying. In fact, both 2FPFB and 2FPVB are NP-complete. We prove the NP-completeness of 2FPFB by showing that the WPDPC [5] problem is a special case of 2FPFB, and prove the NP-completeness of 2FPVB by generalizing from the single-path FPVB problem.

Theorem 1: 2FPFB is NP-complete
Proof: We restrict 2FPFB to WPDPC by only considering those instances in which, the available bandwidth of each link does not change, i.e. the bandwidth of each link is constant across all time-slots. In other words, WPDPC is a special case of 2FPFB when the network is static with constant link bandwidths. Since WPDPC is NP-complete [5], so is 2FPFB.

Theorem 2: 2FPFB is NP-complete
Proof: The NP-completeness of FPVB has been established in [11] by reducing from the 0–1 Total Bandwidth (0–1 TB) problem. Obviously, 2FPVB is a more general version of FPVB that computes multiple concurrent FPVB paths.

4 Algorithm Design

The NP-completeness of both 2FPFB and 2FPVB indicates that there does not exist any polynomial-time optimal algorithm, unless $P = NP$. In this section, firstly, we design naive greedy algorithms to solve 2FPFB/2FPVB. Secondly, we design optimal algorithms based on exhaustive search for small-scale 2FPFB/2FPVB problem instances. Lastly, we propose efficient heuristic algorithms for large-scale problem instances.

4.1 Greedy Algorithms for 2FPFB and 2FPVB

Greedy2FPFB Algorithm. For 2FPFB, based on the single-path OptFPFB algorithm in [12], we design a polynomial-time greedy algorithm, referred to as Greedy2FPFB, whose pseudocode is provided in Algorithm 1. In Line 1, it first employs OptFPFB to compute the first path p_1 with the maximum fixed bandwidth BW_1, assuming the use of one single path to transfer data of size δ. In Line 2, it removes all nodes on path p_1 from the original graph, resulting in a new graph $G'(V, E)$ comprised of only the residual nodes and links. In Line 3, it employs OptFPFB to compute the second path p_2 in $G'(V, E)$ with the maximum

fixed bandwidth BW_2, again assuming the use of one single path to transfer data of size δ. In Line 4, it computes the maximum combined bandwidth β of the two paths p_1 and p_2. In Lines 5–7, it computes the end time for the transfer of data size δ using these two node-disjoint paths concurrently. In order to ensure that the two data transfers concurrently taking place on these two paths finish at the same time, it allocates the data size δ to p_1 and p_2 in proportion to their fixed bandwidths. In Line 8, it computes the data size δ_1 to be transferred by p_1, and the rest $(\delta - \delta_1)$ is assigned to p_2. The time complexity of Greedy2FPFB is the same as that of OptFPFB, which is $O(T^2 \cdot m \cdot \log n + T^3 \cdot m)$ [12], where n is the number of nodes, m is the number of links, and T is the total number of new time slots in the ATB list.

Algorithm 1. Greedy2FPFB

Input: an HPN graph $G(V, E)$ with an ATB list of T time slots, source v_s, destination v_d, and data size δ

Output: the earliest transfer end time t_{end}, data partition δ_1

1: $p_1(BW_1) = OptFPFB(G, T, v_s, v_d, \delta)$;
2: Remove the nodes and links of p_1 from G to create a new G';
3: $p_2(BW_2) = OptFPFB(G', T, v_s, v_d, \delta)$;
4: $\beta = BW_1 + BW_2$;
5: **if** $(\beta \cdot (t[T-1] - t[0]) \geq \delta$ and $\beta \neq 0)$ **then**
6: $\quad \tau = \delta / \beta$;
7: $\quad t_{end} = t[0] + \tau$;
8: $\delta_1 = BW_1 \times \tau$;
9: **return** t_{end}, δ_1;

Greedy2FPVB Algorithm. For 2FPVB, based on the single-path MinFPVB algorithm in [12], we design a greedy algorithm, referred to as Greedy2FPVB, whose pseudocode is provided in Algorithm 2. In Lines 1-3, it initializes the bandwidth in each time-slot for each of two paths $BW_1[i], BW_2[i], i = 0, 1, \ldots, T-1$, the remaining data size rd and the transfer time $time$. In Line 4, it employs MinFPVB to compute the first path p_1 with the maximum bandwidth $BW_1[i]$ in each time slot, assuming the use of one single path. In Line 5, it removes the nodes and links of path p_1 from the original graph G, resulting in a new graph G' comprised of only the residual nodes and links. In Line 6, it employs MinFPVB to compute the second path p_2 with the maximum bandwidth $BW_2[i]$ in each time slot, again assuming the use of one single path. In Lines 7-20, it computes the maximum combined bandwidth of the two paths p_1 and p_2 in the order of sequential time slots, and then computes the transfer end time. In each time slot, it allocates the data size to p_1 and p_2 in proportion to their bandwidths to ensure that these two concurrent data transfers finish at the same time. Similarly, the time complexity of Greedy2FPVB is also the same as that of MinFPVB [12], which is $O(m(T + logn))$.

Algorithm 2. Greedy2FPVB

Input: an HPN graph $G(V, E)$ with an ATB list of T time slots, source v_s, destination v_d, and data size δ

Output: the earliest transfer end time t_{end}, data partition δ_1

```
1: for (i = 0; i ≤ T − 1; i++) do
2:    BW1[i] = 0, BW2[i] = 0;
3: rd = δ; time = 0;
4: p1(BW1[]) = MinFPVB(G, T, vs, vd, δ);
5: Remove the nodes and links of path p1 from G to create G′;
6: p2(BW2[]) = MinFPVB(G′, T, vs, vd, δ);
7: i = 0; δ1 = 0;
8: while (rd ≥ 0) do
9:    BWSum[i] = BW1[i] + BW2[i];
10:   δ1 = δ1 + BW1[i] · (t[i + 1] − t[i]);
11:   if (rd ≥ BWSum[i]) then
12:      time = time + 1;
13:      rd = rd − BWSum[i] · (t[i + 1] − t[i]);
14:   else
15:      time = time + rd/BWSum[i];
16:      rd = 0;
17:   tend = t[0] + time;
18:   i = i + 1;
19: return tend, δ1.
```

4.2 Optimal Algorithms for 2FPFB and 2FPVB

We design two exhaustive search-based optimal algorithms for 2FPFB and 2FPVB, referred to as Opt2FPFB and Opt2FPVB, to compute the path-pair that minimizes the data transfer end time in a brute-force manner: (i) find all possible paths between the source and the destination in all possible time ranges across contiguous time slots, (ii) enumerate all possible node-disjoint path-pairs in each time range, and choose the path-pair that minimizes the data transfer end time. Obviously, Opt2FPFB and Opt2FPVB are of exponential time complexity and are only meant for being used as a comparison base in small-scale problem instances. We will design more efficient heuristics below for large-scale problems.

4.3 Improved Algorithms for 2FPFB and 2FPVB

In Greedy2FPFB and Greedy2FPVB, although the transfer time of each path for data size δ is minimized at each respective step, there is no guarantee that the concurrent transfer time τ by the path-pair for data size δ is minimized from a global perspective, especially when the bandwidths of p_1 or p_2 before time slots τ are smaller than the later time slots. We propose two improved algorithms, i.e. Imp2FPFB and Imp2FPVB, to overcome their defects.

Imp2FPFB Algorithm. The pseudocode of Imp2FPFB is provided in Algorithm 3. It first calls the previous Greedy2FPFB algorithm to compute the concurrent transfer time τ by the path-pair (p_1, p_2) for data size δ, and the data partition δ_1 to be assigned to path p_1. It then uses OptFPFB twice to find a pair of paths (p'_1, p'_2) with larger bandwidths than the path-pair (p_1, p_2) during $[0, \tau]$ time slots, and computes the transfer time by (p'_1, p'_2) for δ. The data partitioning method is the same as in Greedy2FPFB, i.e. the data size to be transferred by each path is proportional to its bandwidth. Since the time complexity of OptFPFB is $O(T^2 \cdot m \cdot \log n + T^3 \cdot m)$ [12], the time complexity of Imp2FPFB is also $O(T^2 \cdot m \cdot \log n + T^3 \cdot m)$, which is the same as that of Greedy2FPFB.

Algorithm 3. Imp2FPFB

Input: an HPN graph $G(V, E)$ with an ATB list of T time slots, source v_s, destination v_d, and data size δ

Output: the earliest transfer end time t_{end}

1: $(\tau, \delta_1) = Greedy2FPFB(G, T, v_s, v_d, \delta)$;
2: $\delta_2 = \delta - \delta_1$;
3: $p'_1(BW'_1) = OptFPFB(G, \tau, v_s, v_d, \delta_1)$;
4: Remove the nodes and links of path p'_1 to create G';
5: $p'_2(BW'_2) = OptFPFB(G', \tau, v_s, v_d, \delta_2)$;
6: $\beta' = BW'_1 + BW'_2$;
7: **if** $(\beta' \cdot (t[\tau - 1] - t[0]) \geq \delta$ and $\beta \neq 0)$ **then**
8: $\quad t_{end} = t[0] + \delta/\beta'$;
9: **return** t_{end}.

Imp2FPVB Algorithm. The pseudocode of Imp2FPVB is provided in Algorithm 4. In Line 1, it calls the previous Greedy2FPVB algorithm to compute the concurrent transfer time τ by a path-pair (p_1, p_2) for data size δ and the variable bandwidths of each path $BW_1[i], BW_2[i], i = 0, 1, \ldots \tau - 1$, and the data size δ_1 to be transferred by p_1. In Line 2, it computes the data size δ_2 to be transferred by p_2. In the remaining lines of code, it uses MinFPVB to find a path-pair (p'_1, p'_2) whose combined bandwidth is larger than that of the path-pair (p_1, p_2) during $[0, \tau)$ time slots, computes the time-varying bandwidths $BW'_1[i], BW'_2[i], i = 0, 1, \ldots \tau - 1$, of each path, and computes the transfer end time for δ by (p'_1, p'_2). The data partitioning is the same as in Greedy2FPVB. In each time slot, it allocates the data size to these two paths in proportion to their bandwidths to ensure that these two concurrent transfers finish at the same time. The time complexity of Imp2FPVB is the same as that of MinFPVB [12], i.e. $O(m(T + logn))$.

Algorithm 4. Imp2FPVB

Input: an HPN graph $G(V, E)$ with an ATB list of T time slots, source v_s, destination v_d, and data size δ

Output: the earliest transfer end time t_{end}

1: $p_1, p_2(\tau, \delta_1) = Greedy2FPVB(G, T, v_s, v_d, \delta)$;
2: $\delta_2 = \delta - \delta_1$;
3: $p'_1(BW'_1[]) = MinFPVB(G, \tau, v_s, v_d, \delta_1)$;
4: Remove the nodes and links of path p_1 to create G';
5: $p'_2(BW'_2[]) = MinFPVB(G', \tau, v_s, v_d, \delta_2)$;
6: $rd = \delta$; $time = 0$;
7: **while** $(rd \geq 0)$ **do**
8: $\quad BWSum[i] = BW'_1[i] + BW'_2[i]$;
9: $\quad$ **if** $(rd \geq BWSum[i])$ **then**
10: $\quad\quad time = time + 1$;
11: $\quad\quad rd = rd - BWSum[i] \cdot (t[i+1] - t[i])$;
12: $\quad$ **else**
13: $\quad\quad time = time + rd/BWSum[i]$;
14: $\quad\quad rd = 0$;
15: $\quad t_{end} = t[0] + time$;
16: **return** t_{end}.

5 Performance Evaluation

We evaluate the performance of the proposed algorithms in small- and large-scale simulated networks as well as a network based on the topology of the real-life ESnet of the U.S. Department of Energy.

5.1 Simulation Setup

For performance evaluation, we create a set of networks of randomly generated topology with a different number of nodes and links of random bandwidths, which follow a normal distribution: $b = b_{max} \cdot e^{-\frac{1}{2}(x)^2} Mb/s$, where b_{max} is set to be 100 Gbps, x is a random variable within the range of $[0, 1]$. The aggregated time-bandwidth (ATB) list contains 100 time slots starting from the time $t[0] = 0$. For each user request, we randomly select a source node v_s and a destination node v_d, and set the data size for transfer to be 500 GBytes.

5.2 Algorithm Comparison for 2FPFB/2FPVB in Small Networks

We first generate 10 random small-scale networks, indexed from 1 to 10, as shown in Table 1.

For 2FPFB, in each of these 10 small-scale networks, we run Imp2FPFB, Greedy2FPFB, and Opt2FPFB for 10 times with different random seeds, and plot the mean and standard deviation of the data transfer end time in Fig. 1 for comparison. We observe that Opt2FPFB always achieves the best performance as expected and Imp2FPFB consistently outperforms Greedy2FPFB. It is worth

Table 1. Index of 10 small-scale networks.

Index of network size	1	2	3	4	5	6	7	8	9	10
Number of nodes	7	10	12	15	17	20	23	26	28	30
Number of links	10	15	18	20	23	26	29	32	35	37

pointing out that Imp2FPFB achieves a close-to-optimal performance in these small-scale problem instances.

We also evaluate the performance of Imp2FPVB, Greedy2FPVB and Opt2FPVB for 2FPVB in the same 10 small-scale networks, and plot their performance measurements in Fig. 2. Similarly, we observe that Opt2FPVB always achieves the best performance as expected, and Imp2FPVB outperforms Greedy2FPVB in all the cases we studied. It is also worth pointing out that Imp2FPVB achieves a close-to-optimal performance in these small-scale problem instances.

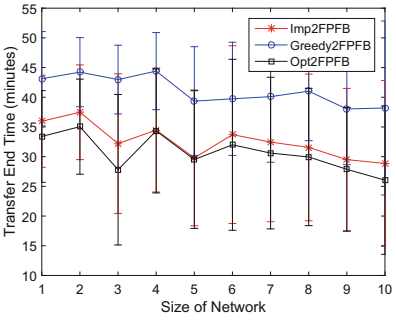

Fig. 1. Performance comparison of the algorithms for 2FPFB in small networks.

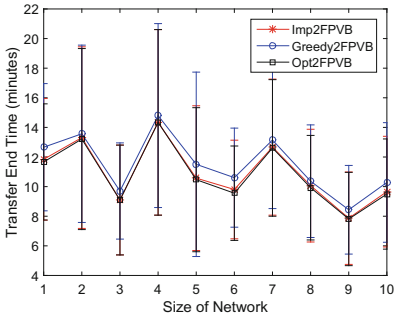

Fig. 2. Performance comparison of the algorithms for 2FPVB in small networks.

5.3 Algorithm Comparison for 2FPFB/2FPVB in Large Networks

We generate 10 different large-scale networks with a random topology, indexed from 1 to 10, as shown in Table 2. We run Greedy2FPFB/GreedyFPVB and Imp2FPFB/Imp2FPVB in each of these network instances for 10 times with different random seeds. Note that Opt2FPFB/Opt2FPVB is of exponential time complexity and hence is not tested in these large-scale networks.

The performance measurements for 2FPFB and 2FPVB are plotted in Figs. 3 and 4, respectively. For 2FPFB, we observe that Imp2FPFB consistently outperforms Greedy2FPFB, and for 2FPVB, Imp2FPVB consistently outperforms Greedy2FPVB.

Table 2. Index of 10 large-scale networks.

Index of network size	1	2	3	4	5	6	7	8	9	10
Number of nodes	40	50	60	70	80	90	100	120	150	200
Number of links	80	100	120	140	160	180	200	240	300	400

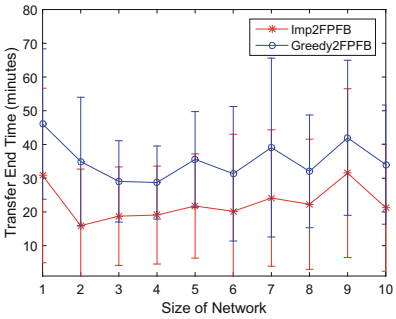

Fig. 3. Performance comparison of the algorithms for 2FPFB in large networks.

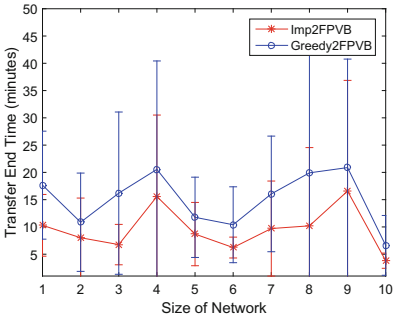

Fig. 4. Performance comparison of the algorithms for 2FPVB in large networks.

5.4 Algorithm Comparison for 2FPFB/2FPVB in ESnet5

To evaluate the performance in a practical setting, we run the proposed algorithms on the topology of a real-life HPN, ESnet5 [13], with 57 nodes and 65 links, each of which has a bandwidth between 30 Gbps–100 Gbps. The user requests have a variable data transfer size. We plot the performance measurements of 2FPFB and 2FPVB in Figs. 5 and 6, respectively, which show that Imp2FPFB and Imp2FPVB achieve better performance than Greedy2FPFB and Greedy2FPVB, respectively.

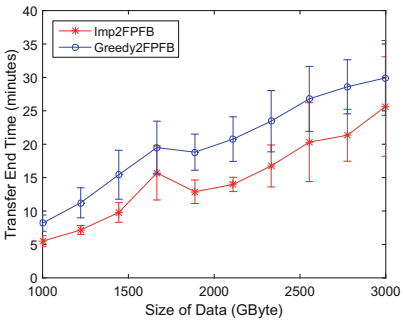

Fig. 5. Performance comparison of the algorithms for 2FPFB in ESnet5.

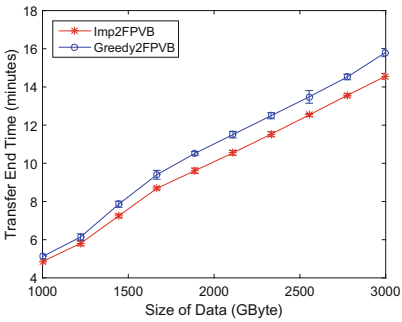

Fig. 6. Performance comparison of the algorithms for 2FPVB in ESnet5.

6 Conclusion

We investigated a problem of bandwidth scheduling with two different types of node-disjoint paths in dedicated networks. We showed that these two problems are NP-complete, and designed heuristic approaches. Extensive results based on both simulated and real-life networks illustrated that the proposed heuristics achieve a superior performance over other algorithms in comparison. It is of our future interest to incorporate and test these scheduling algorithms in the control plane of existing HPNs.

Acknowledgment. This research is sponsored by U.S. NSF under Grant No. CNS-1560698 with New Jersey Institute of Technology, and National Nature Science Foundation of China under Grant No. 61472320 and Beilin Science and Technology Plan under Grant No. GX1403 with Northwest University, P.R. China.

References

1. Domzal, J., Dulinski, Z., Kantor, M.: A survey on methods to provide multipath transmission in wired packet networks. COMNET **77**, 18–41 (2015)
2. Mieghem, P.V., Kuipers, F.A.: On the complexity of QoS routing. Comput. Commun. **26**, 376–387 (2003)
3. Kuipers, F.A., VanMieghem, P.F.: Conditions that impact the complexity of QoS routing. IEEE/ACM Trans. Netw. **13**, 717–730 (2005)
4. Liang, W.: Robust routing in wide-area WDM networks. In: Proceedings of IPDPS, San Francisco, CA (2001)
5. Shen, B.H., Hao, B., Sen, A.: On multipath routing using widest pair of disjoint paths. In: Workshop on High Performance Switching and Routing, pp. 134–140 (2004)
6. Dahshan, M.H.: Maximum-bandwidth node-disjoint paths. Int. J. Adv. Comput. Sci. Appl. **3**, 48–56 (2012)
7. Andreas, A.K. Smith, J.C.: Exact algorithms for robust k-path routing problems. In: Proceedings of GO, pp. 1–6 (2005)
8. Loh, R.C., Soh, S., Lazarescu, M.: Maximizing bandwidth using disjoint paths. In: 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 304–311 (2010)
9. Sidhu, D., Nair, R., Abdallah, S.: Finding disjoint paths in networks. In: Proceedings of ACM SIGCOMM, pp. 43–51 (1991)
10. Ramasubramanian, S., Krishnamoorthy, H., Krunz, M.: Disjoint multipath routing using colored trees. COMNET **51**(8), 2163–2180 (2007)
11. Guerin, R., Orda, A.: Networks with advance reservations: the routing perspective. In: Proceedings of the 19th IEEE INFOCOM (2000)
12. Lin, Y., Wu, Q.: Complexity analysis and algorithm design for advance bandwidth scheduling in dedicated networks. IEEE/ACM Trans. Netw. **21**(1), 14–27 (2013)
13. ESnet. https://www.es.net