Maximizing Bandwidth Using Disjoint Paths

Ruen Chze Loh, Sieteng Soh, and Mihai Lazarescu
Department of Computing
Curtin University of Technology, Perth, Australia
{ruen-chze.loh@postgrad.curtin, S.Soh@curtin, M.Lazarescu@curtin}.edu.au

Abstract – **Recently, multi-paths solutions have been proposed to improve the quality-of-service (QoS) in communication networks (CNs). This paper addresses the problem to obtain the λ-edge-disjoint-path-set (λDP/B) with maximum bandwidth (λDP^B), for λ≥1. λDP/B is useful for applications that require maximum bandwidth for data transmission, such as video conferencing, video-on-demand, large file downloads and FTP. We propose a polynomial time heuristic algorithm, Maximum Bandwidth Algorithm (MBA), to solve the problem. We have implemented MBA and evaluated its performance against an optimal, but exponential time, brute force algorithm (BF) and three existing heuristic algorithms: Algorithm-1, CBA-G', DPSP'. Simulations on seventy CNs show that MBA is able to produce the optimal λDP^B for about 99% of the time while using only 0.005% CPU time of BF. Our simulations also show that MBA is significantly more effective than these existing algorithms while using competitive CPU time.**

Keywords – **edge-disjoint paths; maximum bandwidth; path bandwidth; network bandwidth; network QoS.**

I. INTRODUCTION

Multi-paths solutions [1-10] have been proposed to improve the source to destination (s,t) quality-of-service (QoS) such as bandwidth, cost, delay and reliability in communication networks (CNs). The paths are used to provide load-balancing [1, 2], reduce delay transmission [3, 4], increase bandwidth [4, 5] or reliability [6, 7], help in path recovery [8, 9] and provide fault-tolerance [8, 10]. Typically, there are more edge-disjoint paths (DP) than node-disjoint paths in a network, thus DP is more commonly used [11]. The problem of optimizing the QoS of (s,t) disjoint paths had been shown to be NP-complete [5, 12]. Thus, heuristics [1, 5] and approximate [13-15] solutions have been used to solve this problem.

The problem to generate (s,t) paths in CN with bandwidth QoS has been addressed in [16-19]. Reference [16] attempts to find one path that satisfies both the bandwidth and delay constraints. Reference [17] proposed a routing solution to find one path with minimum bandwidth consumption and minimizing delay while [18] proposed a partial protection concept where backup paths are created for a selected set of domains in a network so as to meet bandwidth and reliability requirements with the minimum cost. However, the solutions in [16-18] cannot be used to generate a DP with maximum bandwidth. Reference [19] proposed a routing algorithm that select paths to maximize bandwidth, minimize delay and packet loss rate. Since [19] considers all the three QoS, the paths selected may not have the maximum bandwidth.

Maximizing the bandwidth in a CN is of practical importance [5, 19] for bandwidth-intensive applications like video conferencing, video-on-demand, file downloads from the Internet and file transfer using FTP. Since the bandwidth of an (s,t) path is limited by the minimum bandwidth of the edge in the path, the most viable solution to increase the total (s,t) bandwidth in the network is to use multiple paths.

This paper addresses the problem by generating an (s,t) λ-edge-disjoint-path-set (called λDP^B) that in parallel has a maximum total bandwidth, for λ≥1. Note that, λDP^B, while maximizing bandwidth, increases system fault tolerance using disjoint paths. Reference [5] proposed Algorithm-1 to obtain λDP^B. However, the heuristic solution [5] requires to generate all possible (s,t) paths in the network which increases exponentially with respect to the number of edges in the CN. Therefore, the solution in [5] is not feasible for CN with many (s,t) paths.

In this paper, we propose a polynomial-time heuristics algorithm, Maximum Bandwidth Algorithm (MBA), to generate λDP^B. Our simulations in Section V show that MBA significantly outperforms Algorithm-1. To evaluate the performance of MBA, we compare the results of MBA with an exponential time brute force approach (BF), and two other heuristic algorithms, CBA-G' and DPSP'. Note that CBA-G' and DPSP' algorithms are the modified versions of the CBA-G [6] and DPSP [12] algorithms respectively, which have been shown to produce almost optimal DP with maximum reliability.

This paper is organized as follows. Section II discusses the network model and notations. Section III formulates the λDP/B problem and discusses the related works as the basis of our approach. Section IV describes our MBA. Section V presents the simulation results and Section VI concludes our paper.

II. NETWORK MODEL AND NOTATIONS

A CN is modeled by an undirected edge-weighted graph G=(V,E,b) without multiple edges and self-loops. Each edge $e_j \in$ E is characterized by its bandwidth, $b_j \in$ b, where $b_j \geq 1$ is an integer representing the bandwidth capacity of e_j. The vertices and edges in G may represent computers and communication links, respectively.

An (s,t) simple path P$_i$ between vertices s and t is formed by the set of edges such that no vertex is traversed more than once. Any proper subset of a simple path does not result in a path between the vertex pair. The pathset P_{st} is a set whose elements are (s,t) simple paths. Fig. 1 shows an example CN for s=1 and t=11; the value inside each bracket on each edge indicates the edge bandwidth.

The P_{st} of Fig. 1 are: $P_1=\{a, b, c, d, e\}$, $P_2=\{a, b, c, r\}$, $P_3=\{a, b, c, h, k, m\}$, $P_4=\{f, g, k, m\}$, $P_5=\{f, g, h, d, e\}$, $P_6=\{f, g, h, r\}$, $P_7=\{n, p, q, m\}$, $P_8=\{n, p, q, k, h, d, e\}$ and $P_9=\{n, p, q, k, h, r\}$.

Paths P_i and P_j are edge-disjoint paths (DP) if $e_\alpha \neq e_\beta$ for each $e_\alpha \in P_i$ and $e_\beta \in P_j$. In other words, there is no edge in P_i that is in P_j. Let $\lambda DP_\sigma \subseteq P_{st}$ be a DP, where $\lambda \geq 1$ is the total number of paths in the DP, and path index σ is any integer. For a given P_{st} there can be more than one λDP_σ, and none of them is a subset of any other. For example, the CN in Fig. 1 has seven λDP_σ: $3DP_1=\{P_1,P_6,P_7\}$, $2DP_2=\{P_1,P_9\}$, $2DP_3=\{P_2,P_4\}$, $2DP_4=\{P_2,P_8\}$, $3DP_5= \{P_2,P_5,P_7\}$, $1DP_6=\{P_3\}$ and $2DP_7=\{P_1,P_4\}$.

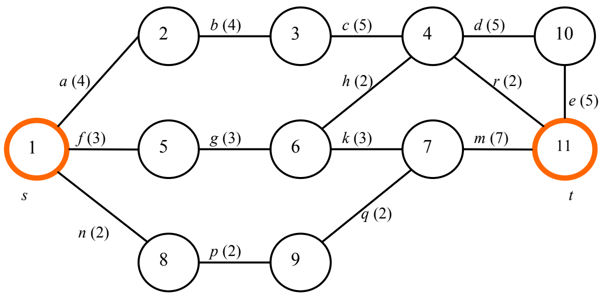


Figure 1. A CN with 11 vertices and 14 edges

The bandwidth of path P_i, $\beta(P_i)$, is the minimum bandwidth among all the edges in P_i;

$$\beta(P_i) = \min_{e_j \in P_i}(b_j) \tag{1}$$

For example, $\beta(P_1)$=min(4,4,5,5,5)=4. The bandwidth of λDP_σ, $\beta(\lambda DP_\sigma)$, is the sum of $\beta(P_i)$, for all $P_i \in \lambda DP_\sigma$, *i.e.*,

$$\beta(\lambda DP_\sigma) = \sum_{P_i \in \lambda DP_\sigma} \beta(P_i) \tag{2}$$

For example, $\beta(3DP_1)=\beta(P_1)+\beta(P_6)+\beta(P_7)= 4+2+2=8$.

III. λDP/B Problem and Related Work

A. Problem Formulation

For a given G(V,E,b), let λDP_σ be any DP with λ-edge-disjoint paths. The λ-edge-disjoint paths with maximum bandwidth problem, λDP/B, is to find the DP with maximum bandwidth (called λDP^B), *i.e.*,

$$\beta(\lambda DP^B) = \max(\beta(\lambda DP_\sigma)) \tag{3}$$

To illustrate the λDP/B problem, consider the CN in Fig. 1. Using Eq. (1), we obtained, $\beta(P_1)=4$, $\beta(P_2)=2$, $\beta(P_3)=2$, $\beta(P_4)=3$, $\beta(P_5)=2$, $\beta(P_6)=2$, $\beta(P_7)=2$, $\beta(P_8)=2$ and $\beta(P_9)=2$. Using Eq. (2), we obtained, $\beta(3DP_1)=8$, $\beta(2DP_2)=6$, $\beta(2DP_3)=5$, $\beta(2DP_4)=4$, $\beta(3DP_5)=6$, $\beta(1DP_6)=2$ and $\beta(2DP_7)=7$. Thus, $\lambda DP^B=3DP_1$ since $\max(\beta(\lambda DP_\sigma))= \beta(3DP_1)=8$.

One may obtain the $\max(\beta(\lambda DP_\sigma))$ by exhaustively generating all possible λDP_σ, then using Eq. (3) to select one with the maximum bandwidth. Note that G(V,E,b), in general, contains an exponential number (in terms of |E|) of (s,t) paths ($|P_{st}|$), and therefore this brute force (BF) approach may generate an exponential number (in terms of $|P_{st}|$) of λDP_σ, and thus this solution has double exponential (in terms of |E|) time complexity. Note that we use this BF approach to evaluate the performance of MBA in Section V.

B. Related Work

Flow networks using multiple paths have been studied [20-22] since the work done by Ford and Fulkerson [23]. However, the problem in [20-23] is different from λDP/B. In flow networks, an edge can be shared by multiple paths, *i.e.*, the paths are not edge-disjoint. In contrast, the bandwidth of an edge in λDP/B cannot be shared by a different (s,t) path. In practical networks, links can fail. Thus, the solution to λDP/B is better than the flow-based methods for use in some critical systems since the edge-disjoint paths generated for λDP/B would improve (s,t) communication reliability [12]. However, the maximum bandwidth for λDP/B is upper bounded by the maximum flow in the network.

Reference [5] proposed Algorithm-1 to solve the λDP/B problem. Algorithm-1 first finds all possible (s,t) paths from G(V,E,b). Then, it transforms the G(V,E,b) into a path intersection graph, G'(N,L), in which each node $N_i \in N$ represents an (s,t) simple path P_i in G(V,E,b), and each link $L_{i,j} \in L$ that connects N_i and N_j indicates that paths P_i and P_j are non-edge-disjoint, *i.e.*, share some edges. Each N_i has a weight which is the bandwidth of P_i. Finally, Algorithm-1 proceeds to find the maximum weight independent set in G'. Reference [5] proposes to use an approximation algorithm for this final step since finding a maximum independent set in a graph is a well-known NP-complete problem. Nevertheless, Algorithm-1 runs in exponential time since it requires to generate all possible (s,t) paths in the network which increases exponentially with respect to the number of edges, |E|. Further, our evaluations in Section V show that Algorithm-1 produced optimal λDP^B in less than 45% of the time.

Reference [6] proposes CBA algorithm to optimize the (s,t) communication reliability in a CN. Each edge in CN has a weight (between 0.0 and 1.0) that represents the edge's operational probability. Conceptually, CBA is similar to Algorithm-1. The CBA algorithm transforms the G(V,E,b) into a weighted *path graph* PG(N,L) in which each node $N_i \in N$ represents an (s,t) simple path P_i in G(V,E,b), and each link $L_{i,j} \in L$ that connects N_i and N_j indicates that paths P_i and P_j are edge-disjoint. Each N_i has a weight, which is the reliability of P_i. Note that the PG(N,L) is the inverse of G'(N,L) used by Algorithm-1 [5] if the weight of each N_i in PG(N,L) is the path bandwidth of P_i. Thus, the problem of finding the DP with maximum reliability/bandwidth is transformed into finding the clique [24] in PG(N,L) with the maximum weight. Unfortunately, like the maximum weight independent set solution in [5], the maximum weight clique in CBA is also an NP-complete problem. Therefore the authors [6] proposed a greedy heuristic polynomial time algorithm, CBA-G, that generates the maximal cliques directly from G(V,E,b).

CBA-G produced almost optimal results [6]. However, it cannot be used as such to solve the λDP/B since the path reliability used in [6] is computed from the summation of edge weight (*i.e.*, reliability) while the path bandwidth in λDP/B is computed from the minimum among the weight (*i.e.*, bandwidth) in the path. Further, the shortest path function (*i.e.*, Yen's algorithm [25]) used in [6] aims to *minimize* the path weight while our problem's goal is to *maximize* the path weight (*i.e.*, bandwidth). In Section V, we show how to modify CBA-G [6] to solve λDP/B problem. Our simulations in Section V show that the modified CBA-G (called CBA-G') produced λDP as optimal as those using Algorithm-1, while using less than 0.01% CPU time of Algorithm-1.

Reference [12] proposes the DPSP algorithm to optimize the (*s*,*t*) reliability in a CN. Like CBA-G, the heuristic algorithm does not require to generate exponential number of (*s*,*t*) paths (in terms of |E|) and thus is a polynomial time algorithm. However, similar to CBA-G, DPSP cannot be used as such to solve the λDP/B problem. In Section V, we used the max_bandwidth_path() function (presented in Section IV.A) for the shortest path function used in DPSP so that the algorithm can be used to solve λDP/B. Nevertheless, as shown in our simulations (Section V), the modified DPSP (called DPSP') is only as optimal as Algorithm-1 and CBA-G'. Therefore, a better approach is needed to solve λDP/B more optimally.

IV. Proposed Algorithm

A. Finding Path with the Maximum Bandwidth

In this section, we propose a max_bandwidth_path() function as shown in Fig. 2 to heuristically generate an (*s*,*t*) path with the maximum bandwidth from G(V,E,b). We use this function in our MBA (described in Section IV.B).

As shown in Fig. 2, max_bandwidth_path() function uses a shortest_path() function (*e.g.*, Dijkstra's [25]) to obtain the (*s*,*t*) path. However, the shortest_path() function cannot be used on G(V,E,b) since it aims to *minimize* the path weight while our goal is to *maximize* the path weight, *i.e.*, bandwidth. Therefore, Line 1 of max_bandwidth_path() converts the weighted bandwidth graph G=(V,E,b) into a weighted graph G'(V,E,b') such that each $b_i' \in b'$ has a weight computed as,

$$b_i' = \zeta - b_i \quad (4)$$

where ζ is at least one unit higher than the maximum edge bandwidth in G(V,E,b). Notice that an edge b_i that has larger bandwidth will have a smaller weight b_i'. Therefore, the shortest_path() function (Line 2) on G'(V,E,b') will produce an (*s*,*t*) path P_i with minimum weight, which heuristically obtains a path in G(V,E,b) with the maximum bandwidth. If shortest_path() fails to find an (*s*,*t*) path in G'(V,E,b'), function max_bandwidth_path() returns FALSE (Line 4); otherwise, it returns TRUE (Line 6).

```
max_bandwidth_path (G, $P_i$)
Input  : G(V,E,b)
Output: $P_i$ with maximum bandwidth

 1.   Generate G' from G;
 2.   $P_i$ = shortest_path(G');
 3.   if $P_i$ = NULL then
 4.     return FALSE;
 5.   else
 6.   return TRUE;
```

Figure 2. Function max_bandwidth_path()

The max_bandwidth_path() function does not always generate (*s*,*t*) path P_i with maximum bandwidth. Notice that the bandwidth of a path is computed from the minimum bandwidth of the edges in the path (see Eq. (1)). In contrast, the shortest_path() function generates path P_i from G'(V,E,b'), where its weight, wt(P_i), is calculated by taking the summation of the weight of each edge within P_i, *i.e.*,

$$\text{wt}(P_i) = \sum_{e_j \in P_i} (b_j) \quad (5)$$

For example, consider G(V,E,b) in Fig. 1. We obtained the G'(V,E,b') as shown in Fig. 3 by substituting ζ=10 into Eq. (4); note that the maximum edge bandwidth in G is 5, thus any value≥5+1 can be used as ζ.

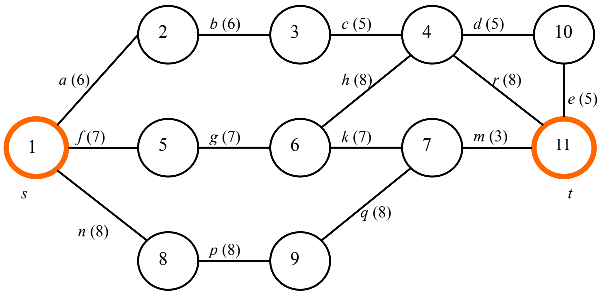


Figure 3. G'(V,E,b') of Fig. 1

Using Eq. (5), we obtain wt(P_1)=27, wt(P_2)=25, wt(P_3)=35, wt(P_4)=24, wt(P_5)=32, wt(P_6)=30, wt(P_7)=27, wt(P_8)=49 and wt(P_9)=47. Therefore, shortest_path() algorithm will return P_4 as the shortest path. However, the path which has bandwidth β(P_4)=3 is not optimal, since path P_1 with β(P_1)=4 has a higher bandwidth. Notice that the shortest_path() does not generate P_1 since wt(P_1)=27 is larger than wt(P_4)=24.

The time complexity of the max_bandwidth_path() function is as follows. Generating G' from G depends on |E|, thus its time complexity is O(|E|). The shortest path algorithm (*e.g.*, Dijkstra [25]) has a time complexity of O(|E|*log|V|), thus, the time complexity of the max_bandwidth_path() function is O(|E|*log|V|).

B. Maximum Bandwidth Algorithm (MBA)

Let ES be a sequence of outgoing edges from source, *s* sorted in the decreasing order of their bandwidths. Similarly, let EB be a sequence of the remaining edges in E sorted in the decreasing order of their bandwidths. Note, |ES| + |EB| = |E|, and ES[1] (EB[1]) is the edge with the largest bandwidth in ES (EB). Our proposed greedy heuristics algorithm, MBA as shown in Fig. 4, generates a λDP^B from G(V,E,b) starting from a path with the largest bandwidth.

Among the edges in ES, MBA selects the edge with the highest bandwidth, b_i (Line 2) and forms a subgraph G_i(V,E',b) from G(V,E,b) by including only edges in E that have bandwidth at least b_i (Line 4). Line 5 uses the max_bandwidth_path() function as shown in Fig. 2 to find a path P_i with maximum bandwidth in G_i(V,E',b). If there is no (*s*,*t*) path found and all the edges in EB have already been evaluated, it outputs λDP^B (Lines 6-8). Otherwise, it sets b_i with the next highest bandwidth in EB (Line 9). Each time a path P_i is found in Line 5, the algorithm inserts the path into λDP^B (Line 12), updates G(V,E,b), ES and EB by removing all edges in P_i from each of them (Line 13) and sets b_i to highest bandwidth in the updated ES (Line 2). The algorithm produces λDP^B (Line 15) when all the edges in ES have been considered.

The time complexity of MBA is as follows. The worst case of |ES| is when |ES|=|E|, thus, the first while-loop is O(|E|). Line 4 has O(|E|) and the max_bandwidth_path() function in Line 5 has O(|E|*log|V|). Thus, the time complexity of MBA is O($|E|^2$*log|V|).

```
Input   : G(V,E,b)
Output: λDP^B

1.  while (|ES| > 0) do
2.    b_i = β(ES[1]);  found=FALSE; j=1;
3.    while (found=FALSE) do
4.      form G_i(V,E',b) from G(V,E,b);
5.      found=max_bandwidth_path(G_i, P_i);
6.      if (found=FALSE) then
7.        if j>|EB|
8.          exit the while loop in Line 1;
9.        b_i=β(EB[j++]);
10.     end if;
11.   end while;
12.   store P_i to λDP^B;
13.   remove the edges in P_i from G(V,E,b), ES, and EB;
14. end while;
15. output λDP^B;
```

Figure 4. Maximum Bandwidth Algorithm (MBA)

C. Illustrating Example

To illustrate MBA, let us use it to generate λDP^B in G(V,E,b) shown in Fig. 5(a). MBA finds the edges {*a*}, {*d*}, {*e*}, {*h*}, {*k*} and {*r*} that are connected to the source and store the edges into ES sorted in decreasing order of bandwidth as {*h*, *k*, *d*, *r*, *a*, *e*}. Similarly, those edges that are not stored in ES are stored in EB sorted as {*f*, *b*, *t*, *j*, *c*, *w*, *g*, *m*, *n*, *u*, *v*, *y*, *p*, *q*, *s*, *x*, *z*}. MBA then assigned 90 to b_i since 90 is the highest bandwidth in ES and form G_i(V,E',b) with those edges in G(V,E,b) that has bandwidth at least 90. MBA then transform G_i(V,E',b) into G_i'(V,E',b) using Eq. (4) with ζ=100 because the maximum edge bandwidth in Fig. 5(a) is 90, thus any value ≥90+1 can be the ζ.

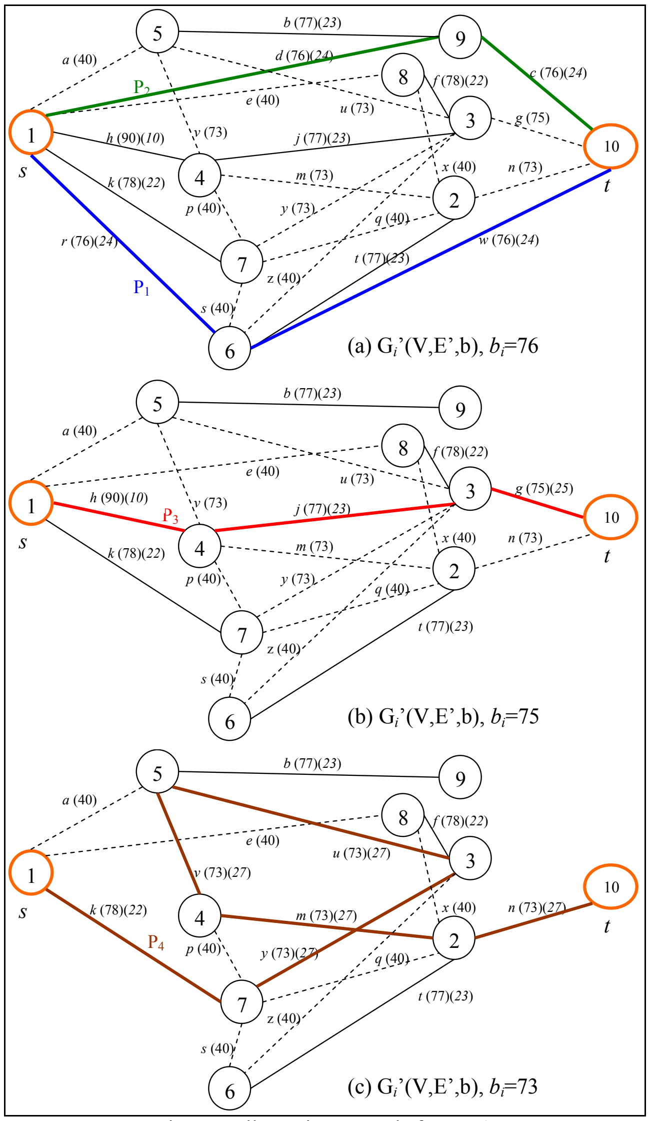


Figure 5. Illustrating example for MBA

The weights of the edges are indicated as italics in brackets in Fig. 5(a). The max_bandwidth_path( ) function is then used to find the shortest (*s*,*t*) path in G_i'(V,E',b). However, no path can be found, so b_i is decreased to 78 (bandwidth of edge *f*), 77 (bandwidth of edges *b*, *t* and *j*) and 76 (bandwidth of edges *c* and *w*). When b_i=76, the G_i'(V,E',b) formed is shown as non-dotted lines in Fig. 5(a). Those dotted lines in Fig. 5(a) have bandwidth less than 76 and will not be considered when the max_bandwidth_path( ) function is executed next. The max_bandwidth_path( ) function finds two paths, {*r*, *w*} and {*d*, *c*} sequentially, each with bandwidth of 76 and stores them in λDP^B. The edges *r*, *w*, *d* and *c* are removed from G(V,E,b), ES and EB. Thus, ES={*h*, *k*, *a*, *e*} and EB={*f*, *b*, *t*, *j*, *g*, *m*, *n*, *u*, *v*, *y*, *p*, *q*, *s*, *x*, *z*}. b_i is decreased further to 75 (bandwidth of edge *g*) and path {*h*, *j*, *g*} with bandwidth 75 is found as shown in Fig. 5(b). ES and EB are then updated with {*k*, *a*, *e*} and {*f*, *b*, *t*, *m*, *n*, *u*, *v*, *y*, *p*, *q*, *s*, *x*, *z*}, respectively. b_i is then further decreased to 73 (bandwidth of edges *m*, *n*, *u*, *v* and *y*) and path {*k*, *y*, *u*, *v*, *m*, *n*} with bandwidth 73 is found as shown in Fig. 5(c). ES and EB are then updated with {*a*, *e*} and {*f*, *b*, *t*, *p*, *q*, *s*, *x*, *z*}, respectively. The value of b_i is decreased further to 40 (bandwidth of edges *a* and *e*) and no more (*s*,*t*) paths can be found. Thus, MBA outputs the λDP^B as {*r*, *w*}, {*d*, *c*}, {*h*, *j*, *g*} and {*k*, *y*, *u*, *v*, *m*, *n*} with a total bandwidth of 76+76+75+73=300.

Note that the proposed MBA may not produce the optimal λDP^B. As an example, consider the G(V,E,b) and G'(V,E,b') in Fig. 1 and Fig. 3, respectively with ES={*a*, *f*, *n*} and EB={*m*, *c*, *d*, *e*, *b*, *g*, *k*, *h*, *r*, *p*, *q*}. In the first iteration of MBA, b_i=4 (bandwidth of edge *a*), thus, MBA obtained the G_i'(V,E',b) shown in Fig. 6(a).

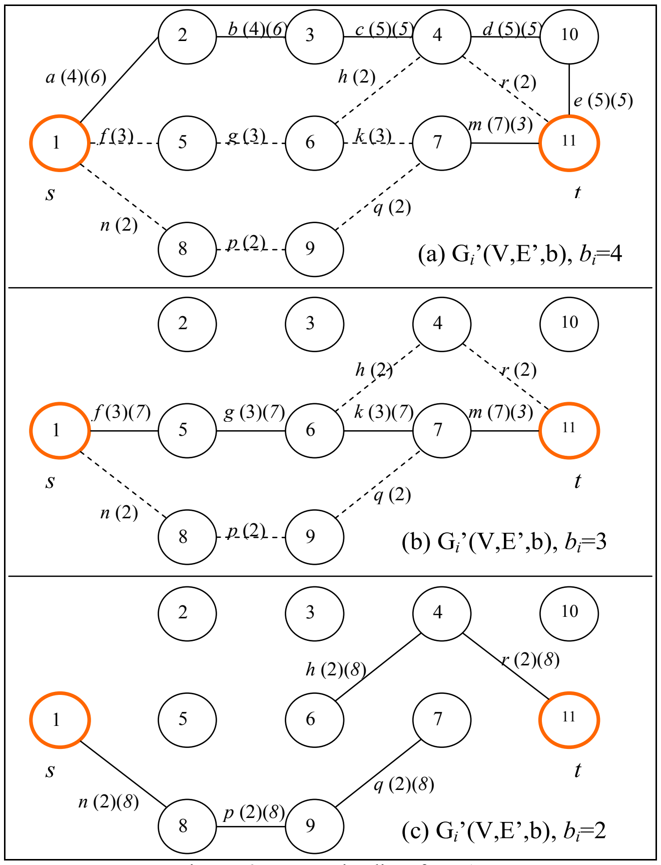


Figure. 6. Non-optimality of MBA

Substituting ζ=10 into Eq. (4), the weights of the edges are indicated as italics in brackets in Fig. 6(a).

Utilizing the max_bandwidth_path( ) function, MBA found the path {*a*, *b*, *c*, *d*, *e*} with bandwidth 4. In the next iteration with b_i=3 (bandwidth of edge *f*), MBA obtained the G_i'(V,E',b) shown in Fig. 6(b). MBA found the path {*f*, *g*, *k*, *m*} with bandwidth 3. Using the last edge in ES, we obtained b_i=2 (bandwidth of edge *n*) and the G_i'(V,E',b) is shown in Fig. 6(c). Since there is no (*s*,*t*) path found in G_i, MBA outputs λDP^B containing two paths, {*a*, *b*, *c*, *d*, *e*} and {*f*, *g*, *k*, *m*}, with a total bandwidth of 7. The MBA could not obtain the λDP^B with the optimal bandwidth, $\beta(\{P_1, P_6, P_7\})$=8.

V. Simulation and Discussion

A. Simulation Setup

We have evaluated the performances of MBA against our implementations of Algorithm-1 [5], CBA-G', DPSP', and BF algorithms. We have modified CBA-G [6] into CBA-G' as follows. First, we convert G(V,E,b) into G'(V,E,b'); see Section IV.A for the conversion steps. Then, we used G'(V,E,b') as the input to the Yen's algorithm utilized in CBA-G. Notice that since CBA-G' (CBA-G) considers bandwidth (reliability), we have replaced all reliability parameters in CBA-G into bandwidth for CBA-G'. We have used a similar conversion steps to modify the optimal but exponential time CBA algorithm [6] into BF. One may refer to [6] for the detail implementations of both CBA-G and CBA algorithms. We have also modified the DPSP algorithm [12] into DPSP' as follows. Similar to CBA-G' and BF, we first convert G(V,E,b) into G'(V,E,b'). Then, we replace the Dijkstra algorithm used in DPSP with our max_bandwidth_path() function (discussed in Section IV.A) since DPSP (DPSP') aims to generate λDP with maximum reliability (bandwidth). Finally, we replaced the two reliability metrics in DPSP with their equivalent bandwidth metrics.

We have implemented all of the 5 algorithms (BF, Algorithm-1, MBA, CBA-G' and DPSP') in C, and ran them on a 2x Intel Pentium 2-2.6Ghz with 1.8GB of RAM, running Fedora Core 6.

B. Simulation-1

In Simulation-1, we compare the performances of MBA, CBA-G' and DPSP' against Algorithm-1 [5] and BF on the 20 vertices ARPANET topology in [5]. Following [5], we randomly generated uniformly distributed edge bandwidth between 2 to 12, and selected the 17 pairs of source-destination nodes as shown in Table I. Unlike in [5], we repeated this simulation 10 times using different random edge assignments to generate 10 random CNs to obtain better average results.

Table I shows the average percentage difference in bandwidth obtained by each of the algorithms from that using the BF for the 10 networks. Notice that our implementation of Algorithm-1 produced similar results with the implementation in [5]. The average CPU time (in seconds) required by each of the algorithms to obtain each λDP^B is shown in Table II.

As shown in Tables I and II, our MBA produced optimal results while on average using less than 0.005% of the CPU time needed by the optimal BF. The tables also show that MBA outperformed CBA-G', DPSP', and Algorithm-1. MBA generated λDPs with higher bandwidths than those obtained using CBA-G', DPSP', or Algorithm-1, while using a comparable CPU time than CBA-G' or DPSP'. Notice that either CBA-G' or DPSP' generated λDPs with comparable bandwidth than those obtained by Algorithm-1 [5] while using on average less than 0.005% of the CPU time.

TABLE I
AVERAGE PERCENTAGE DIFFERENCE IN BANDWIDTH FROM BF

Source	Destination	$\|P_{st}\|$	Existing Algorithms			
			Algorithm-1	CBA-G'	DPSP'	MBA
1	6	596	20.00	0.00	10.00	0.00
1	11	682	0.00	0.00	9.09	0.00
1	20	712	14.29	0.00	0.00	0.00
5	11	974	0.00	0.00	0.00	0.00
10	3	534	7.69	7.69	15.38	0.00
10	5	827	0.00	0.00	0.00	0.00
10	17	755	15.38	7.69	7.69	0.00
14	2	923	0.00	9.09	18.18	0.00
14	6	835	0.00	0.00	0.00	0.00
14	19	753	13.04	13.04	13.04	0.00
18	4	935	0.00	0.00	0.00	0.00
18	8	579	15.38	7.69	15.38	0.00
18	12	713	8.70	4.35	0.00	0.00
20	3	793	9.09	0.00	0.00	0.00
20	7	597	0.00	9.09	9.09	0.00
20	13	789	10.00	0.00	0.00	0.00
20	16	953	9.09	0.00	0.00	0.00

TABLE II
AVERAGE CPU TIME IN SECONDS FOR ARPANET

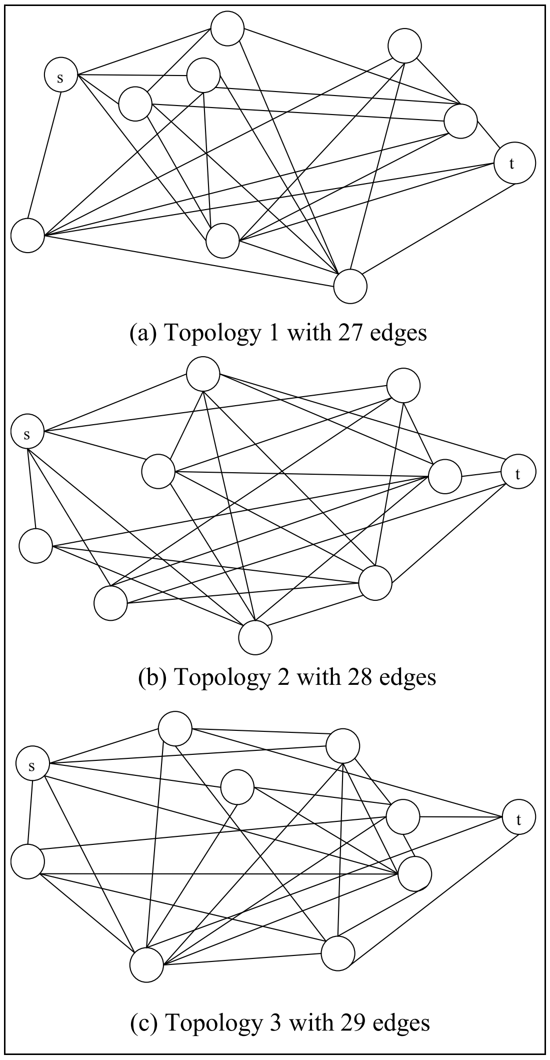
BF	Algorithm-1	CBA-G'	DPSP'	MBA
435.551	417.973	0.020	0.015	0.019

C. *Simulation-2*

In Simulation-2 we used BRITE [26] with the RTWaxman configuration to generate 3 random topologies that contains 10 vertices with 1970, 2458 and 3086 (*s*,*t*) paths as shown in Fig. 7(a), (b) and (c) respectively. Simulation-2 was aimed at showing the effects of setting different ranges of edge bandwidths to the optimality of the algorithms.

For each topology, we randomly assigned each of its edges with bandwidth values and uniformly distribute them ranging from 1 to 10. We then ran the 5 algorithms to obtain a λDP^B. We repeated this simulation 10 times using different random edge assignments for the given value range to generate 30 random CNs. Further, we repeat the described simulation on each topology using a wider range of bandwidth (*i.e.*, from 1 to 100) to generate the other 30 random CNs. In the simulations, all 5 algorithms (BF, Algorithm-1, MBA, CBA-G' and DPSP') produced λ=4 for all the 60 random CNs. We evaluated the effectiveness of the 4 heuristics algorithms with the optimal BF in Table III and IV.

Table III tabulates the results obtained when the link bandwidths were assigned ranging from 1 to 10. Each number in the data cells of Table III indicates the average percentage of the occurrence of λDP^B that has bandwidth difference from the optimal bandwidth within the corresponding columns percentage range. As shown in the table, MBA generated all optimal λDP^B for Topology 1, Topology 2, and Topology 3. Notice that Algorithm-1 was the worst in all topologies. For Topology-1, Algorithm-1 produced only 40% optimal λDP^B, and 20%, 10%, 20%, 10% λDP^B with bandwidth (6% to 10%), (11% to 15%), (16% to 20%), and (21% to 30%) less optimal, respectively.



(a) Topology 1 with 27 edges
(b) Topology 2 with 28 edges
(c) Topology 3 with 29 edges

Figure 7. The 3 randomly generated topologies

We further investigated the effectiveness of the algorithm on networks with larger ranges of bandwidth (*i.e.*, 1 to 100) and tabulated the results in Table IV. As shown in Table IV, MBA produced optimal λDP^B almost 99% of the time, with the remaining 1% generated λDP^B having bandwidth within 10% off the optimal. Thus, increasing the edge bandwidth range from (1 to 10) to (1 to 100) only slightly reduces the optimality of MBA. Similarly, as shown in Tables III and IV, setting bandwidth range differently only slightly affected the optimality of Algorithm-1, CBA-

G' and DPSP'. Note that the computational time of each of the algorithms was not affected by the different bandwidth range setting.

TABLE III
AVERAGE PERCENTAGE OCCURRENCE FOR LINK BANDWIDTH RANGING FROM 1 TO 10

	Bandwidth difference from optimal (in %)					
	0	1-5	6-10	11-15	16-20	21-30
Topology 1						
Algorithm-1	40	0	20	10	20	10
CBA-G'	70	0	20	10	0	0
DPSP'	50	0	0	30	20	0
MBA	100	0	0	0	0	0
Topology 2						
Algorithm-1	50	0	10	10	20	10
CBA-G'	50	10	20	0	10	10
DPSP'	50	0	10	10	20	10
MBA	100	0	0	0	0	0
Topology 3						
Algorithm-1	40	0	20	20	10	10
CBA-G'	60	0	20	20	0	0
DPSP'	50	0	20	10	10	10
MBA	100	0	0	0	0	0

TABLE IV
AVERAGE PERCENTAGE OCCURRENCE FOR LINK BANDWIDTH RANGING FROM 1 TO 100

	Bandwidth difference from optimal (in%)					
%	0	1-5	6-10	11-15	16-20	21-30
Topology 1						
Algorithm-1	40	10	30	20	0	0
CBA-G'	70	10	20	0	0	0
DPSP'	50	10	20	10	10	0
MBA	100	0	0	0	0	0
Topology 2						
Algorithm-1	50	0	40	10	0	0
CBA-G'	60	30	10	0	0	0
DPSP'	40	10	30	20	0	0
MBA	98.3	0	1.7	0	0	0
Topology 3						
Algorithm-1	40	10	50	0	0	0
CBA-G'	50	20	30	0	0	0
DPSP'	30	10	20	30	10	0
MBA	98.3	1.7	0	0	0	0

We compare the time efficiency of the 5 algorithms in Table V which shows the average CPU time in seconds required for each of the algorithms to generate λDP^B for the 3 random topologies. The average number of cliques generated by BF is 2703241 which is much higher compared to the average of 19 cliques generated by CBA-G'.

TABLE V
AVERAGE CPU TIME IN SECONDS FOR THE 3 RANDOM TOPOLOGIES

BF	Algorithm-1	CBA-G'	DPSP'	MBA
680.151	610.753	0.037	0.021	0.035

As shown in Tables III, IV and V, MBA (CBA-G' and DPSP') produces a λDP^B with larger (equivalent) bandwidth than Algorithm-1 while using just 0.006% of the CPU time compared to Algorithm-1.

VI. CONCLUSION

In this paper, we have addressed an important λDP/B problem to generate a λDP^B – a set of λ-edge-disjoint path with maximum bandwidth. A heuristics polynomial time algorithm, MBA, has been proposed to solve the problem. We have evaluated the optimality and time efficiency of MBA using simulations on seventy CNs with random edge bandwidths. The simulations showed that MBA was able to produce the optimal λDP^B about 99% of the time while using only 0.005% CPU time of the optimal but exponential time algorithm, BF. Further, we have shown that MBA produced significantly more optimal λDP^B than the existing approaches: Algorithm-1, CBA-G', and DPSP'.

The optimality of MBA is affected by the use of non-optimal max-bandwidth-path() function for generating an (s,t) path with maximum bandwidth. In future work, we plan to improve the performance of the function to increase the performance of MBA. The use of λDP^B in some applications, *e.g.*, multimedia, may increase jitter and buffer management complexity. We leave this issue for future study.

REFERENCES

[1] Y. Guo, F. Kuipers, and P. Van Mieghem, "Link-disjoint paths for reliable QoS routing," *Int'l J. Comm. Sys.,* vol. 16, no. 9, pp. 779-798, 2003.
[2] W. Xu, P. Yan, and D. Xia, "Method of load balancing based on disjoint multi-pathsrouting," *Jisuanji Gongcheng/ Computer Engineering,* vol. 32, no. 24, pp. 36-39, 2006.
[3] R. C. Loh, S. Soh, and M. Lazarescu, "Edge Disjoint Paths with Minimum Delay Subject to Reliability Constraint," *Proc. IEEE APCC, Shanghai, China*, 2009.
[4] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," *Seventh International Workshop on Quality of Service, IWQoS*, pp. 119-128, 1999.
[5] A. Sen, B. Hao, B. H. Shen, L. Zhou, and S. Ganguly, "On maximum available bandwidth through disjoint paths," *Workshop on HPSR*, pp. 34-38, 2005.
[6] R. C. Loh, S. Soh, M. Lazarescu, and S. Rai, "A Greedy Technique for Finding the Most Reliable Edge-disjoint-path-set in a Network," *Proc. IEEE PRDC, Taiwan*, pp. 216-223, 2008.
[7] A. K. Andreas, and J. C. Smith, *Mathematical Programming Algorithms for Two-Path Routing Problems with Reliability Considerations*, Working Paper, Tucson, AZ, 2005.
[8] M. Conforti, R. Hassin, and R. Ravi, "Reconstructing edge-disjoint paths," *Operations Research Letters,* vol. 31, no. 4, pp. 273-276, 2003.
[9] K. Kar, M. Kodialam, and T. V. Lakshman, "Routing restorable bandwidth guaranteed connections using maximum 2-route flows," *Proc. 21st IEEE INFOCOM,* vol. 1, 2002.
[10] M. Kodialam, and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," *Proc. 19th IEEE INFOCOM,* vol. 2, 2000.
[11] A. Agarwal, and B. Jain, "Routing reliability analysis of segmented backup paths in mobile ad hoc networks," *ICPWC*, pp. 52-56, 2005.
[12] P. Papadimitratos, Z. J. Haas, and E. G. Sirer, "Path set selection in mobile ad hoc networks," *Proc. ACM MobiHoc, Lausanne, Switzerland*, pp. 1-11, 2002.
[13] C. Peng, and H. Shen, "A New Approximation Algorithm for Computing 2-Restricted Disjoint Paths," *IEICE Trans. Info. and Sys.,* vol. 90, no. 2, pp. 465-472, 2007.
[14] R. C. Loh, S. Soh, and M. Lazarescu, "Finding the Best Approximate Multi-Constrained QoS Edge-disjoint-path-set," *The 9th Postgraduate Electrical Engineering and Computing Symposium, PEECS*, pp. 31-36, 2008.

[15] D. H. Lorenz, and D. Raz, “A simple efficient approximation scheme for the restricted shortest path problem,” *Op. Research Letters,* vol. 28, no. 5, pp. 213-219, 2001.

[16] T. Korkmaz, and M. Krunz, “Bandwidth-delay constrained path selection under inaccurate state information,” *IEEE/ACM TON,* vol. 11, no. 3, pp. 384-398, 2003.

[17] S. Bistarelli, and F. Santini, “A Formal and Practical Framework for Constraint-Based Routing,” *7th ICN*, pp. 162-167, 2008.

[18] A. Chakrabarti, and G. Manimaran, “Reliability constrained routing in QoS networks,” *IEEE/ACM TON,* vol. 13, no. 3, pp. 662-675, 2005.

[19] X. Zhuang, and S. Zhu, “Disjoint Multipath QoS Routing,” *4th IEEE CCNC*, pp. 556-559, 2007.

[20] H. Zhuge, “Knowledge flow network planning and simulation,” *Decision Support Systems,* vol. 42, no. 2, pp. 571-592, 2006.

[21] Y. K. Lin, “Evaluate the performance of a stochastic-flow network with cost attribute in terms of minimal cuts,” *Reliability Engineering and System Safety,* vol. 91, no. 5, pp. 539-545, 2006.

[22] S. N. Kabadi, R. Chandrasekaran, K. P. K. Nair, and Y. P. Aneja, “Integer version of the multipath flow network synthesis problem,” *Discrete Applied Mathematics,* vol. 156, no. 18, pp. 3376-3399, 2008.

[23] L. Ford, and D. Fulkerson, “Max flow through a network,” *Can. J. Mathematics*, pp. 399–404, 1956.

[24] P. M. Pardalos, and J. Xue, “The maximum clique problem,” *Journal of Global Optimization,* vol. 4, no. 3, pp. 301-328, 1994.

[25] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik,* vol. 1, no. 1, pp. 269-271, 1959.

[26] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: Universal Topology Generation from a User’s Perspective,” *Proc. IEEE MASCOTS, Ohio, USA*, pp. 346-353, 2001.