Edge Disjoint Paths with Minimum Delay Subject to Reliability Constraint

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, multipaths solutions have been proposed to improve the quality-of-service (QoS) in communication networks (CN). This paper describes a problem, λDP/RD, to obtain the λ-edge-disjoint-path-set such that its reliability is at least R and its delay is minimal, for λ≥1. λDP/RD is useful for applications that require non-compromised reliability while demanding minimum delay. In this paper we propose an approximate algorithm based on the Lagrange-relaxation to solve the problem. Our solution produces λDP that meets the reliability constraint R with delay≤(1+k)Dmin for k≥1, and Dmin is the minimum path delay of any λDP in the CN. Simulations on forty randomly generated CNs show that our polynomial time algorithm produced λDP with delay and reliability comparable to those obtained using the exponential time brute-force approach.

Keywords – approximate algorithm; Lagrange relaxation; multi-constrained edge disjoint paths; network reliability; network delay.

I. INTRODUCTION

The disjoint path set solutions [1-6] have been proposed to improve the end-to-end quality-of-service (QoS) of the communication networks (CN). Since the number of vertex disjoint paths in general is very limited, the edge disjoint path (DP) set that do not share edges is more commonly used [7]. References [1, 6, 8] propose algorithms to improve the reliability of CNs using DP. Reference [9] also shows that the lifetime of an end-to-end communication can be improved with a higher reliability DP.

Some CNs, such as those for time critical systems and multimedia applications, are subjected to multi-constrained QoS, e.g., reliability, delay, cost and bandwidth. [8] considers cost and delay as the constraint parameters, [10, 11] consider cost and reliability and [4] uses reliability and delay. Note that the problem for generating a DP with two or more constraints has been shown NP-hard [12], and therefore heuristic and approximation algorithms [8, 13, 14] have been proposed to address the problem.

Orda and Sprintson [13] proposed four approximation algorithms to find two delay-constrained DPs with minimum total cost (2DP/DC). For a CN that contains two DPs with delay≤D and minimal cost OPT, their best algorithm, 2DP-4, always finds 2DP/DC with delay≤(1+1/k)D and cost≤k(1+γ)(1+ε)OPT, where k is a positive integer representing the approximate index, γ is a small value bounded by 2(log k + 1)/k and ε is an approximate factor. Applying Lagrange-relaxation, Peng and Shen proposed an algorithm (PSA for short) [8] that improves the performance of 2DP-4 to a delay≤(1+1/k)D with cost≤(1+k)OPT. They showed that PSA can be used to obtain λDP/DC, for λ≥2. However, both algorithms in [13] and [8] have one significant limitation; they concentrate on finding only 2DP that satisfy the delay and cost constraints whereas other DP may also satisfy the user defined preconditions. In addition, no simulations were performed to benchmark the feasibility of the algorithms and find the optimal value of k. Loh, et al [2] have recently described a problem to obtain λDP/DR – the set of DPs with maximum reliability subject to delay constraint D, for λ≥1. The authors [2] used a similar Lagrange-relaxation method as in [8] to solve this problem.

Our contribution in this paper is twofold. First, we propose an important problem, which is to find λDP/RD – the set of DPs with minimum delay subject to reliability≥R, for λ≥1. The solution to this problem is obviously applicable to some important critical applications, e.g., emergency response, rescue and military operations that demand certain levels of reliability assurances. Such applications require non-compromised reliability while demanding minimum system delay. Second, we present an approximation algorithm to solve this problem. Our solution generates DP with maximum delay no more than (1+k)Dmin where Dmin is the minimum delay of a path set in the network.

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

II. PRELIMINARIES

A. Network Model and Notations

A CN is modeled by an edge-weighted graph N=(V,E,d,p) where G=(V,E) is an undirected graph without multiple edges and self-loops. Each edge e∈E is characterized by its delay d(e) and its reliability p(e), where d(e) is the time taken for traffic to be transferred from one end to the other of e and 0≤p(e)≤1 represents the probability that e is UP. An e is said to be UP (DOWN) if it is functioning (failed). All vertices in V are assumed to be always UP. The vertices and edges in N may represent computers and communication links, respectively.

An (s,t) simple path P between vertices s and t is formed by the set of UP 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 Pst is a set whose elements are (s,t) simple paths. Fig. I shows an
Paths $P_i$ and $P_j$ are edge disjoint paths (DP) if $e_i \neq e_j$ for each $e_i \in P_i$ and $e_j \in P_j$. In other words, there is no edge in $P_i$ that is in $P_j$. Let $\lambda DP \subseteq \delta P_i$ be a DP, where $\lambda \geq 1$ is the total number of paths in the DP, and $\sigma$ is any integer. For a given $P_i$, there can be more than one $\lambda DP$, and none of them is a sub network of any other. For example, the CN in Fig. 1 has six $\lambda DP$: $3DP_1 = \{P_1, P_2, P_7\}$, $2DP_2 = \{P_1, P_9\}$, $2DP_3 = \{P_2, P_4\}$, $2DP_4 = \{P_2, P_8\}$, $3DP_5 = \{P_3, P_7, P_{11}\}$ and $1DP_6 = \{P_5\}$.

The delay of path $P_i$, $\delta(P_i)$, is the sum of edge delays in $P_i$; e.g., $\delta(P_i) = 2 + 2 + 5 + 2 + 3 = 14$. The delay of $\lambda DP$, $\delta(\lambda DP)$, is the maximum $\delta(P_i)$ for all $P_i \in \lambda DP$, i.e.,

$$\delta(\lambda DP) = \max_{P_i \in \lambda DP} (\delta(P_i)) \quad (1)$$

For example, $\delta(3DP_1) = \max\{\delta(P_1), \delta(P_9), \delta(P_7)\} = \max\{14, 14, 11\} = 14$.

The $(s, t)$ reliability, $\rho(P)$, of a simple path $P_n$ is computed by multiplying $p_{ij}$ of each $e_i$ that forms $P_n$, i.e.,

$$\rho(P) = \prod_{e_i \in P_n} p_{ij} \quad (2)$$

For example, $\rho(P_1) = 0.8 \times 0.7 = 0.19208$. The disjoint paths in an $\lambda DP$ can be viewed as the components of a parallel system [15], and therefore its reliability, $\rho(\lambda DP)$, can be computed as:

$$\rho(\lambda DP) = 1 - \prod_{P_i \in \lambda DP} (1 - \rho(P_i)) \quad (3)$$

The multiplicative operations in (3) can be transformed into additive operations as:

$$\rho(\lambda DP) = 1 - \log^{-1}\left( \sum_{P_i \in \lambda DP} \log(1 - \rho(P)) \right) \quad (4)$$

where $\log^{-1}(X)$ is the antilog of X. Using Eq. (3), $\rho(3DP_1) = (1 - (1 - 0.8))(1 - 0.9) = 1 - \left(1 - (1 - 0.8) \times 0.9 \right) = 0.98$. Note, $\rho(3DP_5)$ can be equivalently computed using Eq. (4) as $1 - \log^{-1}(0.9826) = 0.98$. Thus, $\rho(3DP_1) = 0.98$.

B. Related Work

The Peng and Shen algorithm (PSA) [8] utilizes the Lagrange-relaxation to approximately generate 2DP/DC from a graph $N(V,E,d)$, where $c$ is the set of edge cost and $d$ is the set of edge delay in $G(V,E)$. For a given delay constraint $D$, PSA produces a 2DP/DC with delay less than $(1 + 1/4)D$ and a total cost no more than $(1 + k)OPT$, where OPT is the optimal cost among all 2DP that meet the delay constraint. The total cost of the 2DP/DC is defined as the sum of the cost of each path $P_i \in 2DP$. Note that the algorithm can be extended to produce $\lambda DP/DC$, for $\lambda \geq 2$ [8]. Here, PSA aims to generate the minimum cost $\lambda DP/DC$ with the largest $\lambda$ that meets the delay constraint. In contrast, a $\lambda DP/RD$ is not necessarily a DP with the largest $\lambda$. Therefore, PSA is not suitable for generating $\lambda DP/RD$.

The algorithm in [2] uses a similar technique to PSA to produce a $\lambda DP/RD$ for a given delay constraint $D$ and a graph $N(V,E,d,p)$ where $p$ is the set of edge reliability and $d$ is the set of edge delay in $G(V,E)$. The algorithm [2] is guaranteed to produce a $\lambda DP$ with, respectively, delay and reliability bounded by Eq. (5) and Eq. (6):

$$\delta(\lambda DP) \leq (1 + k)(\log(1 - R)) \quad (5)$$

$$\delta(\lambda DP) \leq (1 + k)(\log(1 - R)) \quad (6)$$

where OPT is the maximum reliability among all possible $\lambda DP$ that satisfy the delay requirement, $D$. Like $\lambda DP/RD$, the generated $\lambda DP/RD$ is not necessarily a DP with the maximum $\rho$. One may use the algorithm in [2] to generate $\lambda DP/RD$ if one substitutes $D$ in Eq. (5), and OPT in Eq. (6) with the maximum path delay $D_{max}$ of the network and the reliability requirement $R$, respectively. Notice that the generated $\lambda DP/RD$ would have a reliability value at most $(1 + k)(\log(1 - R))$, which may not satisfy the reliability requirement, $R$. Therefore, a more effective algorithm, presented in this paper, is needed to solve the $\lambda DP/RD$ problem.

III. $\lambda DP/RD$ PROBLEM FORMULATION

For a given reliability constraint $R$, let $\lambda DP^{rd}_r$ be $\lambda DP$, that has reliability at least $R$, i.e., $\rho(\lambda DP^{rd}_r) \geq R$. Let $\lambda DP^{rd}_r$ be $\lambda DP^{rd}_r$ with minimum delay, i.e.,

$$\delta(\lambda DP^{rd}_r) = \min\{\delta(\lambda DP^{rd}_r)\} \quad (7)$$

Notice that there can be more than one $\lambda DP^{rd}_r$. The edge-disjoint-path-set with minimum delay and reliability $\lambda DP^{rd}$ problem ($\lambda DP/RD$) is to find among all $\lambda DP^{rd}_r$ a DP with the highest reliability (called $\lambda DP^{BM}$), i.e.,

$$\rho(\lambda DP^{BM}) = \max\{\rho(\lambda DP^{rd}_r)\} \quad (8)$$

To illustrate the $\lambda DP/RD$ problem, consider $R = 0.4$ and the $\lambda DP$s of Fig. 1. Using Eq. (4), we obtained, $\rho(3DP_1) = 0.4766$, $\rho(2DP_2) = 0.2942$, $\rho(2DP_3) = 0.4836$, $\rho(2DP_4) = 0.3386$, $\rho(3DP_5) = 0.4934$ and $\rho(1DP_6) = 0.1882$. Among the $\lambda DP$s there are three $\lambda DP^{rd}_r$: $3DP_1^{rd}$, $2DP_2^{rd}$, and $3DP_5^{rd}$. Using Eq. (1), we obtain, $\delta(3DP_1^{rd}) = 14$, $\delta(2DP_2^{rd}) = 14$ and $\delta(3DP_5^{rd}) = 15$; thus using Eq. (7), there are two $\lambda DP^{rd}_r$: $3DP_1^{rd}$, $2DP_2^{rd}$. Eq. (8) obtains $\max\{\rho(3DP_1^{rd})\} = 0.4766$, $\rho(2DP_2^{rd}) = 0.4836$, and thus $\lambda DP^{BM} = 2DP_2^{rd}$. Notice that the optimal $2DP_2$ is not a $\lambda DP$ with the largest $\lambda$.

One may obtain $\lambda DP^{BM}$ by exhaustively generating all possible $\lambda DP^{rd}_r$ path sets and using Eq. (7) to select the set of $\lambda DP^{rd}_r$, and using Eq. (8) select the most reliable one. Note that $N(V,E,d,p)$, in general, contains an exponential number (in terms of $|E|$) of $(s,t)$ paths $P_{mn}$, and therefore this brute force (BF) approach may generate an exponential number (in terms of $|P_{mn}|$) of $\lambda DP^{rd}_r$, and thus this solution has double exponential (in terms of $|E|$) time complexity.

Fig. 1. A CN with 11 vertices and 14 edges

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 25, 2010 at 22:03:13 EST from IEEE Xplore. Restrictions apply.
IV. THE λDP/RD APPROXIMATE ALGORITHM

A. Lagrange-relaxation Approach for λDP/RD

We propose to use Lagrange-relaxation technique to solve the λDP/RD problem. As in [8] and [2], we combine the reliability \( p_i \) and delay \( d_i \) of each edge \( e_i \) into a weight \( w_i = d_i + \eta_i \) to get the reliability, rather than considering them separately. In other words, we transform the two constraint-weighted network \( N(V,E,p,d) \) into the one constraint-weighted network \( N(V,E,w) \), for \( w_i \in \mathbb{R} \). The value of \( \eta \) should be set properly to minimize the delay \( \delta(\lambda DP) \), and to maximize reliability \( \rho(\lambda DP) \). Note that for λDP/DC, it has been shown that setting \( \eta \) (called \( \alpha \) in [8]) to \( kOPT/D \) produced good results. Since λDP/RD is similar to λDP/DC, we set \( \eta = k \cdot D_{min}/\log(R) \), i.e., replacing OPT with \( D_{min} \) and the delay constraint \( D \) with the log of the reliability constraint \( R \), where \( D_{min} \) is the minimum delay of all possible DPs in the network. One may obtain \( D_{min} \) by using DPSP [9] or iDPSP [6] from \( N(V,E,p,d) \) assuming perfect edges, i.e., \( p_i = 1 \) for all \( p_i \in \mathbb{R} \). The following lemma states that when the value of \( \eta \) increases, \( \rho(\lambda DP) \) decreases.

\[
\lambda DP^\eta \text{ is the DP with the minimum total weight when we set } w_i = d_i + \eta_i p_i, \text{ where we have,}
\]

\[
W(\lambda DP^\eta, \eta) = \delta(\lambda DP^\eta) + \eta_i \log(\rho(\lambda DP^\eta)) \leq \delta(\lambda DP^{\eta+1}) + \eta_i \log(\rho(\lambda DP^{\eta+1})) \tag{9}
\]

Since \( \lambda DP^\eta \) is the DP with the minimum total weight when we set \( w_i = d_i + \eta_i p_i \), we have,

\[
W(\lambda DP^{\eta+1}, \eta) = \delta(\lambda DP^{\eta+1}) + \eta_i \log(\rho(\lambda DP^{\eta+1})) \leq \delta(\lambda DP^{\eta}) + \eta_i \log(\rho(\lambda DP^{\eta})) \tag{10}
\]

Adding Eq. (9) and Eq. (10), we obtain:

\[
\delta(\lambda DP^\eta) + \eta_i \log(\rho(\lambda DP^\eta)) \leq \delta(\lambda DP^{\eta+1}) + \eta_i \log(\rho(\lambda DP^{\eta+1})) + \delta(\lambda DP^{\eta+1}) + \eta_i \log(\rho(\lambda DP^{\eta+1})), \text{ and thus log}(\rho(\lambda DP^\eta)) \leq \log(\rho(\lambda DP^{\eta+1})).
\]

Theorem 1. \( \rho(\lambda DP^\eta) \leq \rho(\lambda DP^{\eta+1}) \), for \( 1 \leq \alpha \leq \beta \).

Proof: By definition, \( \eta_i = \alpha D_{min}/\log(R) \) and \( \eta_i = \beta D_{min}/\log(R) \), and therefore, \( \eta_i \leq \eta_i \). Since \( 1 \leq \alpha \leq \beta \), \( \eta_i \leq \eta_i \leq \eta_i \), then by Lemma 1 \( \rho(\lambda DP^\eta) \leq \rho(\lambda DP^{\eta+1}) \) for \( \eta_i \leq \eta_i \leq \eta_i \) and thus for \( 1 \leq \alpha \leq \beta \).

Theorem 2. Consider a \( \lambda DP \) that is obtained when \( \eta_i = kD_{min}/\log(R) \), \( D_{min} \leq W(\lambda DP^\eta, \eta) \leq (1+2k)D_{min} \) iff \( D_{min} \leq \delta(\lambda DP^\eta) \leq (1+2k)D_{min} \).

Proof: To prove that \( D_{min} \leq W(\lambda DP^\eta, \eta) \leq (1+2k)D_{min} \) if \( D_{min} \leq \delta(\lambda DP^\eta) \leq (1+2k)D_{min} \), consider Fig. 2, which describes the feasible solution space for λDP/RD. The x-axis represents the logarithm of all possible reliability values \( \rho(\lambda DP) \) when \( \delta(\lambda DP) = 0 \), i.e., when each edge has zero delay. On the other hand, the y-axis represents all possible delay values \( \delta(\lambda DP) \) assuming perfect edges, i.e., \( \rho(\lambda DP) = 1 \) or \( \log(\rho(\lambda DP)) = 0 \). Since the minimum delay (lower bound) of all possible DPs in the network is \( D_{min} \), the delay values of any point in line AB is \( D_{min} \). A feasible solution must have a reliability of at least R, and therefore the value of each point in line BD is \( \log(R) \). Since the algorithm aims to obtain a \( \lambda DP \) with delay at most \( (1+k)D_{min} \), line CD represents the upper bound of \( \delta(\lambda DP) \). Therefore, Fig. 2 shows \( D_{min} \leq \delta(\lambda DP^\eta) \leq (1+2k)D_{min} \) and \( \log(\rho(\lambda DP^\eta)) \geq \log(R) \).

The upper and lower bounds of the weight of \( \lambda DP^\eta \), \( W(\lambda DP^\eta, \eta) \), can be obtained as follows. Substituting \( \log(\rho(\lambda DP^\eta)) = 0 \) into \( W(\lambda DP^\eta, \eta) = \delta(\lambda DP^\eta) + \eta_i \log(\rho(\lambda DP^\eta)) \), the weights of a \( \lambda DP^\eta \) at points A and C. The weight at point A is \( (1+k)D_{min} \) and at point C. Similarly, substituting \( \delta(\lambda DP^\eta) = R \) and \( \eta_i = kD_{min}/\log(R) \) into \( W(\lambda DP^\eta, \eta) = \delta(\lambda DP^\eta) + \eta_i \log(\rho(\lambda DP^\eta)) \), we obtain \( \delta(\lambda DP^\eta) = (1+k)D_{min} \) at point B and \( \delta(\lambda DP^\eta) = (1+k)D_{min} \) at point D, the weights at B and D are \( (1+k)D_{min} \) and \( (1+2k)D_{min} \), respectively. Therefore, \( D_{min} \leq W(\lambda DP^\eta, \eta) \leq (1+2k)D_{min} \).

B. Algorithm

Fig. 3 shows the λDP/RD algorithm. Let \( \lambda DP^\text{max} \) be the λDP with the maximum reliability, and \( \lambda DP^\text{min} \) be the λDP with the minimum total delay in \( N(V,E,p,d) \). One may use the DPSP [9] or iDPSP [6] algorithm to compute both \( \lambda DP^\text{max} \) and \( \lambda DP^\text{min} \) from \( N(V,E,d) \) by setting each edge \( d \) to 0 and \( p \) to 1, respectively. If the reliability of \( \lambda DP^\text{max} \) is less than the reliability constraint \( R \), there is no feasible solution for the network, and therefore the algorithm terminates. Otherwise, the algorithm aims to obtain the optimal \( \lambda DP^\text{opt} \) that has the maximum total delay in \( N(V,E,p,d) \). One may use the DPSP [9] or iDPSP [6] algorithm to compute both \( \lambda DP^\text{max} \) and \( \lambda DP^\text{min} \) from \( N(V,E,d) \) by setting each delay \( d \) to 0 and \( p \) to 1, respectively. Each iteration in the loop uses iDPSP() to obtain a new \( \lambda DP \), \( \lambda DP^\text{new} \), that has the minimum weight from \( N(V,E,w) \), where the edge weights, \( w \), are calculated for each increasing \( k \). Note that \( \lambda DP^\text{new} \) has new metrics \( NM= (delay, NM, reliability, NM) \). Following Theorem 1, if \( \rho(\lambda DP^\text{new}) < R \) is generated when \( k=\alpha \), then all other \( \rho(\lambda DP^\text{new}) \) generated when \( k=\beta \) will always be less than \( R \).
for $\beta<\alpha$. Thus, the algorithm stops the iteration and return $\lambda D_{PB}^{BM}$ as the approximately best $\lambda D$. Otherwise, from Theorem 2, if $D_{min}<W(\lambda D^{P}, \eta_{k}) \leq (1+2k)D_{min}$ we use function optimal() to select the more optimal $\lambda D$ between $\lambda D_{PB}^{BM}$ and $\lambda D_{PB}^{NM}$, and continue the iteration for larger $k$. From Eqs. (7) and (8), $\lambda D_{PB}^{BM}$ is more optimal than $\lambda D_{PB}^{NM}$ if (i) delay$_{BM}$ < delay$_{NM}$, or (ii) delay$_{BM} =$ delay$_{NM}$ and reliability$_{BM}$ > reliability$_{NM}$.

**Input**: $N$ : the directed graph $N=(V,E,p,d)$;
$R$ : the reliability constraint;

**Output**: approximated $\lambda D_{PB}^{BM}$,
1. if $p(\lambda D_{PB}^{BM})=R$ then return “No such solution”
2. $D_{min}=\delta(\lambda D_{PB}^{min})$;
3. $\lambda D_{PB}^{BM}=\emptyset$;
4. $\lambda D_{PB}^{BM} \leftarrow (\infty, 0)$; $k \leftarrow 0$; done $\leftarrow$ false;
5. $k++$;
6. $\eta_{k} \leftarrow k*D_{min}/log(R)$;
7. for each $e \in E$ do $w_{ei} \leftarrow d_{j} + \eta_{k}*log(\rho_{e})$ endfor;
8. $\lambda D_{PB}^{NM} \leftarrow iDPSP(N,V,E,w)$;
9. if $p(\lambda D_{PB}^{NM})=R$ then done $\leftarrow$ true;
10. else if $(D_{min}<W(\lambda D^{P}, \eta_{k}) \leq (1+2k)D_{min})$ then
$\lambda D_{PB}^{BM} \leftarrow \text{optimal}(\lambda D_{PB}^{BM}, \lambda D_{PB}^{NM})$;
11. until done;
12. output $\lambda D_{PB}^{BM}$.

![Fig. 3. The approximate algorithm for $\lambda D_{PB}/RD$](image-url)

The time complexity of the algorithm can be calculated as follows. Either DPSP [9] or iDPSP [6] have the time complexity of $O(|V||E|^2)$, and line 7 and optimal() has $O(E)$ and $O(1)$ time complexity, respectively. Therefore, the time complexity of the algorithm depends on the total number of the loop is repeated, i.e., on the value of $k$. Thus, time complexity is $O(k|V||E|^2)$.

As an illustrative example, consider the network in Fig. 1 with $R=0.4$. Using iDPSP [6], we obtained $p(\lambda D_{PB}^{BM})=0.49343$ and $\delta(\lambda D_{PB}^{min})=11$. Table I shows the delay, reliability and weight of the $\lambda D$ obtained by the algorithm when $k$ was incremented from 1 to 3.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Delay</th>
<th>Reliability</th>
<th>$W(\lambda D^{P}, \eta_{k})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.493437</td>
<td>15.7</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>0.483619</td>
<td>27.5</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>0.338637</td>
<td>39.3</td>
</tr>
</tbody>
</table>

When $k=1$, $W(\lambda D^{P}, \eta_{1})=15.7$ which satisfies $11 < W(\lambda D^{P}, \eta_{1}) \leq 33$; thus, $BM=15.0, 0.493437$. When $k$ was increased to 2, $W(\lambda D^{P}, \eta_{2})=27.5$ which still satisfies $11 < W(\lambda D^{P}, \eta_{2}) \leq 55$ and the delay$_{NM}$ obtained was 14 which was lesser than delay$_{BM}$; therefore, line 10 replaces $\lambda D_{PB}^{BM}$ with $\lambda D_{PB}^{NM}$ with $BM=(14, 0.483619)$. For $k=3$, the reliability$_{BM}<R$ so the algorithm output the $\lambda D_{PB}^{BM}={P_2, P_4}$ obtained when $k=2$ with $BM=(14, 0.483619)$.

### V. SIMULATION AND DISCUSSION

We used BRITE [16] with the RTWaxman configuration to generate a random topology that contains 50 vertices, 72 edges and 1124 s-t paths. From the topology, we constructed 40 random networks, 10 each for the following four different network groups, CN1, CN2, CN3, and CN4. The edges of each network in CN1 and CN3 are randomly assigned with edge reliabilities ranging from 0.1 to 0.5 and 0.1 to 0.9 respectively with incremental value of 0.1. For CN2, we used edge reliability values ranging from 0.5 to 0.75 with incremental value of 0.05, while for CN4 the random reliability values are ranging from 0.9 to 0.99 with incremental values of 0.01. Further, we also assigned a random delay value ranging from 3 to 7 units to each edge of the 40 networks, and used reliability constraints $R$ of 0.1, 0.5, 0.7 and 0.95 for CN1, CN2, CN3, and CN4, respectively. Note that we generated the 40 random networks such that each of them satisfies the constraint $R$. In all our simulations, the value of $k$ starts from 1. We used the C implementation of our $\lambda D_{PB}/RD$ algorithm to obtain the $\lambda D_{PB}$ for the 40 CNs. All simulations were run on a 2x Intel Pentium 2-2.6Ghz with 1.8GB of RAM, running Fedora Core 6.

### A. The Effects of $k$ on Reliability

We ran $\lambda D_{PB}/RD$ algorithm on all the 40 networks with increasing $k=1, 2, \ldots$; this simulation is used to show the correctness of Theorems 1 and 2. The results for each network are consistent with the theorems; Table II shows the results when the algorithm was executed on one of the CN3s that had $D_{min}$=23.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Delay</th>
<th>Reliability</th>
<th>$W(\lambda D^{P}, \eta_{k})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>0.493437</td>
<td>15.7</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>0.483619</td>
<td>27.5</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>0.338637</td>
<td>39.3</td>
</tr>
</tbody>
</table>

Consistent with Theorem 1, when $k=1$, the algorithm produced the $\lambda D$ with the highest reliability. It remained as the $\lambda D_{PB}^{BM}$ until $k=23$ at which $\lambda D_{PB}^{NM}$ had a lower delay; this $\lambda D_{PB}^{NM}$ became the new $\lambda D_{PB}^{BM}$ since its reliability$\geq R=0.7$. For $k=42$, $\lambda D_{PB}^{NM}$ has reliability 0.667086$<R$. Thus, the algorithm outputs as the solution with the smallest delay the $\lambda D_{PB}^{NM}$ generated when $k=23$. Note that for this CN, the delay $\delta(\lambda D_{PB}^{NM})$ decreased as $k$ increased; however, this may not always be true. Therefore, we need to increase $k$ sequentially to evaluate every $\lambda D_{PB}^{NM}$ using Eqs. (7) and (8) until a value of $k$ produces a $\lambda D_{PB}^{NM}$ with $p(\lambda D_{PB}^{NM})=R$ before deciding on the $\lambda D_{PB}^{BM}$. To validate Theorem 2, we checked that each feasible solution has $D_{min}\leq W(\lambda D^{P}, \eta_{k}) \leq (1+2k)D_{min}$.

### B. The Accuracy of the Algorithm

We used our algorithm to generate $\lambda D_{PB}^{BM}$ and computed its delay $(D_{ours})$ and reliability $(R_{ours})$ as shown in Table III. To evaluate the optimality of our algorithm, we compared $D_{ours}$ and $R_{ours}$ with $D_{BF}$ and $R_{BF}$, respectively, which were generated using an exponential time brute force (BF) algorithm (described in Section III). Note that $D_{ours}$ (or $R_{ours}$) always satisfies the delay bound $D_{bound}=(1+k)D_{min}$ (reliability constraint $R$) for each CN.

We consider four possible comparison results: (i) $D_{ours} < D_{BF}$ and $R_{ours} = R_{BF}$; (ii) $D_{ours} = D_{BF}$ and $R_{ours} < R_{BF}$; (iii) $D_{ours} > D_{BF}$ and $R_{ours} = R_{BF}$; and (iv) $D_{ours} > D_{BF}$ and $R_{ours} < R_{BF}$. The columns “%D_{ours}” and “%R_{ours}” in Table III show the percentage differences between $D_{ours}$ and $R_{ours}$ against $D_{BF}$ and $R_{BF}$.
and $R_{\text{BF}}$ respectively. As shown in column 8 of Table III our algorithm is optimal 25% of the time (marked with (i)) both in terms of delay and reliability, and 47.5% of the time it generates the same delay as BF but with, on average, 2.6% less reliable (a negative value marked with (ii)). Note that 5% of the time our algorithm produced results in category (iv) (marked with (iv)), D$_{\text{ours}}$ is at most 9.76% higher than $D_{\text{BF}}$ and $R_{\text{ours}}$ is at most 2.38% lower than $R_{\text{BF}}$. Our DP/RD algorithm and the BF approach took on average 1.2 seconds and 191 seconds respectively to generate the $\lambda\text{DP}^{\text{BM}}$ of each CN.

Table III shows $k_{\text{max}}$ and $k_{\text{opt}}$ that denote the value of the smallest $k$ for which the algorithm results in $\rho(\lambda\text{DP}^{\text{BM}}) \geq R$ (thus it outputs $\lambda\text{DP}^{\text{BM}}$) and the value of $k$ when $\rho(\lambda\text{DP}^{\text{BM}}) = \max(\rho(\lambda\text{DP}^{\text{BM}}))$, respectively. As shown in column 3, $k_{\text{max}} < 85$, i.e., our algorithm in Fig. 3 iterates the repeat loop at most 84 times for generating each $\lambda\text{DP}^{\text{BM}}$.

<table>
<thead>
<tr>
<th>Table III COMPARISON BETWEEN DP/RD AND BF RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delay</strong></td>
</tr>
<tr>
<td><strong>CN</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

| CONCLUSION |

We have addressed an important $\lambda\text{DP}$/$\text{RD}$ problem to generate a $\lambda\text{DP}$ with minimum delay while meeting a reliability constraint, R. An approximate Lagrange-relaxation algorithm has been presented to solve the problem. Our simulations on forty randomly generated CNs with random edge reliabilities and delays show that our polynomial time method is able to generate $\lambda\text{DP}$/$\text{RD}$ with delay and reliability values comparable to those generated using the optimal but time-expensive brute force approach.

We are investigating a method to bound the value of $k$ to further reduce the complexity of our approach. We also plan to use some alternative heuristic algorithms for the problem.

REFERENCES


To see the optimality of our algorithm in terms of the reliability of the $\lambda\text{DP}^{\text{BM}}$, we compared its results with those generated by the BF approach while setting $D_{\text{ours}}=D_{\text{BF}}$. The column $\%R_{\text{ours}}$ in Table III shows the percentage differences between the reliability obtained by our algorithm against that obtained by BF. As indicated earlier (shown in column 8 in Table III), 25% of the time, our algorithm generates optimal reliability (0% in the column). Even though 75% of the time our approach does not produce the $\lambda\text{DP}^{\text{BM}}$ with optimal reliability, $R_{\text{ours}}$ is at most only -5.77% off the optimal result, while using only 0.79% of the CPU time required by the optimal BF approach.