On Shortest Single/Multiple Path Computation Problems in Fiber-Wireless (FiWi) Access Networks

Chenyang Zhou*, Anisha Mazumder*, Arunabha Sen*, Martin Reisslein† and Andrea Richa*
*School of Computing, Informatics and Decision Systems Engineering
†School of Electrical, Computer, and Energy Engineering
Arizona State University
Email: {czhou24, anisha.mazumder, asen, reisslein, aricha}@asu.edu

***Abstract*—Fiber-Wireless (FiWi) networks have received considerable attention in the research community in the last few years as they offer an attractive way of integrating optical and wireless technology. As in every other type of networks, routing plays a major role in FiWi networks. Accordingly, a number of routing algorithms for FiWi networks have been proposed. Most of the routing algorithms attempt to find the "shortest path" from the source to the destination. A recent paper proposed a novel path length metric, where the contribution of a link towards path length computation depends not only on that link but also every other link that constitutes the path from the source to the destination. In this paper we address the problem of computing the shortest path using this path length metric. Moreover, we consider a variation of the metric and also provide an algorithm to compute the shortest path using this variation. As multipath routing provides a number of advantages over single path routing, we consider disjoint path routing with the new path length metric. We show that while the single path computation problem can be solved in polynomial time in both the cases, the disjoint path computation problem is NP-complete. We provide optimal solution for the NP-complete problem using integer linear programming and also provide two approximation algorithms with a performance bound of 4 and 2 respectively. The experimental evaluation of the approximation algorithms produced a near optimal solution in a fraction of a second.**

I. Introduction

Path computation problems are arguably one of the most well studied family of problems in communication networks. In most of these problems, one or more weight is associated with a link representing, among other things, the *cost, delay* or the *reliability* of that link. The objective most often is to find a *least weighted path* (or "shortest path") between a specified source-destination node pair. In most of these problems, if a link l is a part of a path P, then the contribution of the link l on the "length" of the path P depends only on the weight $w(l)$ of the link l, and is oblivious of the weights of the links traversed before or after traversing the link l on the path P. However, in a recent paper on optical-wireless FiWi network [5], the authors have proposed a path length metric, where the contribution of the link l on the "length" of the path P depends not only on its own weight $w(l)$, but also on the weights of all the links of the path P. As the authors of [5] do not present any algorithm for computing the shortest path between the source-destination node pair using this new metric, we present a polynomial time algorithm for this problem in this paper. This result is interesting because of the nature of new metric proposed in [5], one key property on which the shortest path algorithm due to Dijkstra is based, that is, *subpath of a shortest path is shortest*, is no longer valid. We show that even without this key property, not only it is possible to compute the shortest path in polynomial time using the new metric, it is also possible to compute the shortest path in polynomial time, with a variation of the metric proposed in [5].

The rest of the paper is organized as follows. In section III, we present the path length metric proposed for the FiWi network in [5] and a variation of it. In section IV we provide algorithms for computing the shortest path using these two metrics. As multi-path routing offers significant advantage over single path routing [6], [7], [8], [9], we also consider the problem of computation of a pair of node disjoint paths between a source-destination node pair using the metric proposed in [5]. We show that while the single path computation problem can be solved in polynomial time in all these cases, the disjoint path computation problem is NP-complete. The contributions of the paper are as follows;

- Polynomial time algorithm for single path routing (metric 1) in FiWi networks
- Polynomial time algorithm for single path routing (metric 2) in FiWi networks
- NP-completeness proof of disjoint path routing (metric 1) in FiWi networks
- Optimal solution for disjoint path routing (metric 1) in FiWi networks using Integer Linear Programming
- One approximation algorithm for disjoint path routing in FiWi networks with an approximation bound of 4 and computation complexity $O((n+m)log\ n)$
- One approximation algorithm for disjoint path routing in FiWi networks with an approximation bound of 2 and computation complexity $O(m(n+m)log\ n)$
- Experimental evaluation results of the approximation algorithm for disjoint path routing in FiWi networks

II. Related Work

Fiber-Wirelss (FiWi) networks is a hybrid access network resulting from the convergence of optical access networks such as Passive Optical Networks (PONs) and wireless access networks such as Wireless Mesh Networks (WMNs) capable of providing low cost, high bandwidth last mile access.

978-1-4799-1633-7/14/$31.00 ©2014 IEEE

Because it provides an attractive way of integrating optical and wireless technology, Fiber-Wireless (FiWi) networks have received considerable attention in the research community in the last few years [1], [2], [3], [4], [5], [8], [9]. The minimum interference routing algorithm for the FiWi environment was first proposed in [4]. In this algorithm the path length was measured in terms of the number of hops in the wireless part of the FiWi network. The rationale for this choice was that the maximum throughput of the wireless part is typically much smaller than the throughput of the optical part, and hence minimization of the wireless hop count should lead to maximizing the throughout of the FiWi network. However, the authors of [5] noted that minimization of the wireless hop count does not always lead to throughput maximization. Accordingly, the path length metric proposed by them in [5] pays considerable importance to the traffic intensity at a generic FiWi network node. The results presented in this paper are motivated by the path length metric proposed in [5].

III. Problem Formulation

In the classical path problem, each edge $e \in E$ of the graph $G = (V, E)$, has a weight $w(e)$ associated with it and if there is a path P from the node v_0 to v_k in the graph $G = (V, E)$

$$v_0 \stackrel{w_1}{\rightarrow} v_1 \stackrel{w_2}{\rightarrow} v_2 \stackrel{w_3}{\rightarrow} v_3 \ldots \stackrel{w_k}{\rightarrow} v_k$$

then the *path length* or the *distance* between the nodes v_0 and v_k is given by

$$w(P_{v_0,v_k}) = w_1 + w_2 + \cdots + w_k$$

However, in the path length metric proposed in [5] for optical-wireless FiWi networks [1], [2], [3], the contribution of e_i to the path length computation depends not only on the weight w_i, but also on the weights of the other edges that constitute the path. In the following section, we discuss this metric and a variation of it. We also also formulate the multipath computation problem using this metric.

The *Optimized FiWi Routing Algorithm (OFRA)* proposed in [5] computes the "length" (or weight) of a path P from v_0 to v_k using the following metric

$$w'(P_{v_0,v_k}) = \min_{P} \left(\sum_{\forall u \in P} (w_u) + \max_{\forall u \in P} (w_u) \right)$$

where w_u represents the traffic intensity at a generic FiWi network node u, which may be an optical node in the fiber backhaul or a wireless node in wireless mesh front-end. In order to compute shortest path using this metric, in our formulation, instead of associating a traffic intensity "weight" (w_u) with nodes, we associate them with edges. This can easily be achieved by replacing the node u with weight w_u with two nodes u_1 and u_2, connecting them with an edge (u_1, u_2) and assigning the weight w_u on this edge. In this scenario, if there is a path P from the node v_0 to v_k in the graph $G = (V, E)$

$$v_0 \stackrel{w_1}{\rightarrow} v_1 \stackrel{w_2}{\rightarrow} v_2 \stackrel{w_3}{\rightarrow} \ldots \stackrel{w_k}{\rightarrow} v_k$$

then the *path length* between the nodes v_0 and v_k is given by

$$\begin{aligned} w^+(P_{v_0,v_k}) &= w_1 + w_2 + \ldots + w_k + \texttt{max}(w_1, w_2, \ldots w_k) \\ &= \sum_{i=1}^{k} w_i + \texttt{max}_{i=1}^{k} w_i \end{aligned}$$

In the second metric, the length a path P_{v_0,v_k} : $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$, between the nodes v_0 and v_k is given by

$$\begin{aligned} \bar{w}(P_{v_0,v_k}) &= \sum_{i=1}^{k} w_i + CNT(P_{v_0,v_k}) * \texttt{max}(w_1, w_2, \ldots w_k) \\ &= \sum_{i=1}^{k} w_i + CNT(P_{v_0,v_k}) * \texttt{max}_{i=1}^{k} w_i \end{aligned}$$

where $CNT(P_{v_0,v_k})$ is the count of the number of times $\texttt{max}\ (w_1, w_2, \ldots w_k)$ appears on the path P_{v_0,v_k}. We study the shortest path computation problems in FiWi networks using the above metrics and provide polynomial time algorithms for solution in subsections IV-A and IV-B.

If $w_{max} = \texttt{max}(w_1, w_2, \ldots w_k)$, we refer to the corresponding edge (link) as e_{max}. If there are multiple edges having the weight of w_{max}, we arbitrarily choose any one of them as e_{max}. It may be noted that both the metrics have an interesting property in that in both cases, the contribution of an edge e on the path length computation depends not only on the edge e but also on every other edge on the path. This is so, because if the edge e happens to be e_{max}, contribution of this edge in computation of $w^+(P_{v_0,v_k})$ and $\bar{w}(P_{v_0,v_k})$ will be $2 * w(e)$ and $CNT(P_{v_0,v_k}) * w(e)$ respectively. If e is not e_{max}, then its contribution will be $w(e)$ for both the metrics.

As multipath routing provides an opportunity for higher throughput, lower delay, and better load balancing and resilience, its use have been proposed in fiber networks [6], wireless networks [7] and recently in integrated fiber-wireless networks [8], [9]. Accordingly, we study the problem of computing a pair of edge disjoint paths between a source-destination node pair s and d, such that *the length of the longer path (path length computation using the first metric) is shortest among all edge disjoint path pairs between the nodes s and d.* In subsection IV-C we prove that this problem is NP-complete, in subsection IV-D, we provide an optimal solution for the problem using integer linear programming, in subsections IV-E and IV-F we provide two approximation algorithms for the problem with a performance bound of 4 and 2 respectively, and in subsection IV-F we provide results of experimental evaluation of the approximation algorithms.

IV. Path Problems in FiWi Networks

In this section, we present (i) two different algorithms for shortest path computation using two different metrics, (ii) NP-completeness proof for the disjoint path problem, (iii) two approximation algorithms for the disjoint path problem, and (iv) experimental evaluation results of the approximation algorithms.

It may be noted that, in both metrics $w^+(P_{v_0,v_k})$ and $\bar{w}(P_{v_0,v_k})$, we call an edge $e \in P_{v_0,v_k}$ *crucial*, if $w(e) = max_{i=1}^{k} w(e'), \forall e' \in P_{v_0,v_k}$.

A. Shortest Path Computation using Metric 1

It may be recalled that the path length metric used in this case is the following: $w^+(P_{v_0,v_k}) = \sum_{i=1}^{k} w_i + \texttt{max}_{i=1}^{k} w_i$. If the path length metric was given as $w(P_{v_0,v_k}) = \sum_{i=1}^{k} w_i$, algorithms due to Dijkstra and Bellman-Ford could have been used to compute the shortest path between a source-destination node pair. One important property of the path length metric that is exploited by Dijkstra's algorithm is that "subpath of a shortest path is shortest". However, the new path length metric $\sum_{i=1}^{k} w_i + \texttt{max}_{i=1}^{k} w_i$ does not have this property. We illustrate this with the example below.

Consider two paths P_1 and P_2 from the node v_0 to v_3 in the graph $G = (V, E)$, where $P_1 : v_0 \overset{w_1}{\rightarrow} v_1 \overset{w_2}{\rightarrow} v_2 \overset{w_3}{\rightarrow} v_3$ and $P_2 : v_0 \overset{w_4}{\rightarrow} v_4 \overset{w_5}{\rightarrow} v_2 \overset{w_3}{\rightarrow} v_3$. If $w_1 = 0.25, w_2 = 5, w_3 = 4.75, w_4 = 2, w_5 = 4$, the length of the path P_1, $w^+(P_1) = w_1 + w_2 + w_3 + \texttt{max}(w_1, w_2, w_3) = 0.25 + 5 + 4.75 + \texttt{max}(0.25, 5, 4.75) = 15$ and the length of the path P_2, $w^+(P_2) = w_4 + w_5 + w_6 + \texttt{max}(w_4, w_5.w_6) = 2 + 4 + 4.75 + \texttt{max}(2, 4, 4.75) = 15.5$. Although P_1 is shortest path in this scenario, the length of its subpath $v_0 \overset{w_1}{\rightarrow} v_1 \overset{w_2}{\rightarrow} v_2$ is 0.25 + 5 + max (0.25, 5) = 10.25, which is greater than the length of a subpath of P_2 $v_0 \overset{w_4}{\rightarrow} v_4 \overset{w_5}{\rightarrow} v_2$ 2 + 4 + max (2, 4) = 10, demonstrating that the assertion that "subpath of a shortest path is shortest" no longer holds in this path length metric.

As the assertion "subpath of a shortest path is shortest" no longer holds in this path length metric, we cannot use the standard shortest path algorithm due to Dijkstra in this case. However, we show that we can still compute the shortest path between a source-destination node pair in polynomial time by repeated application of the Dijkstra's algorithm. The algorithm is described next.

For a given graph $G = (V, E)$, w.l.o.g, we assume $|V| = n$ and $|E| = m$. Define G_e as subgraph of G by deleting edges whose weight is greater than $w(e)$.

Also, as Dijkstra's algorithm does, we need to maintain distance vector. We define $dist_v$ be distance (length of shortest path) from s to v, Π_v be predecessor of v and $maxedge_v$ be weight of the crucial edge from s to v via the shortest path, ans_v be optimal solution (length) from s to v.

Different G_e can be treated as different layers of the G. For any path P, we define the function $e^*(P)$ as the crucial edge along P. It is easy to observe that if P_d is the optimal path from s to node d then $w(P_d) = \sum_{e \in P} w(e)$ and $w^+(P_d) = w(P_d) + w(e^*(P_d))$. It may be noted that henceforth, we shorten $P_{s,d}$ to P_d, because we consider that the source is fixed while the destination d is variable.

Lemma 1. *$w(P_d)$ is minimum in $G_{e^*(P_d)}$.*

Proof: It is obvious that P_d still exists in $G_{e^*(P_d)}$, since edges on P_d are not abandoned. Suppose P_d is not shortest, then there must be another path $P_{d'}$ s.t. $w(P'_d) < w(P_d)$. Noting that the crucial edge on P'_d, namely e', is no longer than $e^*(P_d)$ since they both belong to $G_{e^*(P_d)}$. Hence $w^+(P'_d) = w(P'_d) + w(e') < w(P_d) + w(e^*(P_d)) = w^+(P_d)$, contradicting P_d is optimal. ■

Algorithm 1 Modified Dijkstra's Algorithm

```
1:  Initialize ans_v = ∞ for for all v ∈ V
2:  sort all edges according to w(e) in ascending order
3:  for i = 1 to m do
4:      Initialize dist_v = ∞, Π_v = nil, maxedge_v = 0 for
        all v ∈ V
5:      dist_s = 0
6:      Q = the set of all nodes in graph
7:      while Q is not empty do
8:          u = Extract-Min(Q)
9:          for each neighbor v of u do
10:             if e_{u,v} ∈ E(G_{e_i}) then
11:                 t = MAX {maxedge_u , w(e_{u,v})}
12:                 if dist_u + w(e_{u,v}) < dist_v then
13:                     dist_v = dist_u + w(e_{u,v})
14:                     maxedge_v = t
15:                     Π_v = u
16:                 else if dist_u + w(e_{u,v}) == dist_v then
17:                     if maxedge_v > t then
18:                         maxedge_v = t
19:                         Π_v = u
20:                     end if
21:                 end if
22:             end if
23:         end for
24:     end while
25:     for each node v do
26:         ans_v = min{ans_v, dist_v + maxedge_v}
27:     end for
28: end for
```

Lemma 2. *Modified Dijkstra's Algorithm (MDA) computes shortest path while keeping the crucial edge as short as possible in every iteration.*

Proof: Line 4 to 24 works similar to the standard Dijkstra's algorithm does. Besides, when updating distance, MDA also updates the crucial edge to guarantee that it lies on the path and when there is a tie, MDA will choose the edge with the smaller weight. ■

Theorem 1. *Modified Dijkstra's Algorithm computes optimal solution for every node v in $O(m(n+m)logn)$ time.*

Proof: Lemma 1 indicates for any node $v \in V$, optimal solution can be obtained by enumerating all possible crucial edges $e^*(P_v)$ and computing shortest path on $G_{e^*(P_v)}$. By sorting all edges in nondecreasing order, every subgraph $G_{e^*(P_v)}$ is considered and it is shown in lemma 2, MDA correctly computes shortest path for every node v in every $G_{e^*(P_v)}$. Then optimal solution is obtained by examining all shortest path using the $w()$ metric plus the corresponding crucial edge. Dijkstra's algorithm runs $O((n+m)logn)$ time when using

binary heap, hence MDA runs in $O(m(n+m)logn)$ time when considering all layers. ∎

B. Shortest Path Computation using Metric 2

Given a path P, let $e^*(P)$ be the crucial edge along the P and $CNT(P)$ be the number of occurrence of such edge. Now our objective becomes to find a path Q, such that $\bar{w}(P) = \sum_{e \in Q} w(e) + CNT(Q) * w(e^*(Q))$ is minimum.

The layering technique can also be used in this problem. However, shortest path under a ceratin layer may not become a valid candidate for optimal solution. Here, we introduce a dynamic programming algorithm that can solve the problem optimally in $O(n^2m^2)$ time.

Input is a weighted graph $G = (V, E), |V| = n, |E| = m$ with a specified source node s. In this paper, we only consider nonnegative edge weight. As shown before, we use G_e to represent the residue graph by deleting edges longer than e in G. Different from **MDA1**, in order to consider the number of crucial edges, $dist_v$ is replaced by an array $dist_v^0, dist_v^1, ...dist_v^n$. One can think $dist_v^c$ be the shortest distance from s to v by going through exactly c crucial edges and possibly some shorter edges. Similarly, we replace Π_v by $\Pi_v^c, 0 \leq c \leq n$. Each Π_v^c records predecessor of v for the path corresponding to $dist_v^c$. Lastly, ans_v is used as optimal solution from s to v.

Lemma 3. *If P_v is the best path from s to v, i.e., $\bar{w}(P_v)$ is minimum among all s-v path, then P_v is computed in $G_{e^*(P_v)}$ and $dist_v^{CNT(P_v)} = w(P_v)$.*

Proof: By definition, P_v exists in $G_{e^*(P_v)}$ and $CNT(P_v) \geq 1$ since any path should go through at least one crucial edge. Noting $\bar{w}(P_v) = w(P_v) + CNT(P_v) * w(e^*(P_v))$, on one hand if we treat $CNT(P_v)$ as a fixed number, then we need to keep $w(P_v)$ as small as possible. Inspired by idea of bellman-ford algorithm, we can achieve it by enumerating $|P_v|$, i.e., number of edges on P_v. On the other hand, we need to keep tracking number of crucial edges as well. Hence, $dist_v^c$ is adopted to maintain such information, superscript c reflects exact number of crucial edges. From line 12 to line 25, $dist_v^c$ is updated either when it comes from a neighbor who has already witnessed c crucial edges or it comes from a neighbor with $c-1$ crucial edges and the edge between is crucial. In either case, node v gets a path, say P', with exact c crucial edges on it and $w(P)$ is minimum. At last, P_v can be selected by enumerating number of crucial edges and that is what line 30 to 32 does. ∎

Lemma 4. *Maxedge Shortest Path Algorithm(MSPA) runs in $O(n^2m)$ time for each G_e.*

Proof: We can apply similar analysis of bellman-ford algorithm. However, we need to update $dist_v^c$ array, it takes extra $O(n)$ time for every node v in every iteration when enumerating $|P_v|$. Hence, total running time is $O(n^2m)$. ∎

Theorem 2. *MSPA computes optimal path for every $v \in V$ in $O(n^2m^2)$ time.*

Algorithm 2 Maxedge Shortest Path Algorithm

```
1: Initialize ans_v = ∞ for for all v ∈ V
2: sort all edges according to w(e) in ascending order, say
   e_1, e_2, ..., e_m after sorting
3: for i = 1 to m do
4:     Initialize dist_v^c = ∞, Π_v^c = nil for all v ∈ V and all
       0 ≤ c ≤ n
5:     dist_s^0 = 0
6:     for j = 1 to n − 1 do
7:         for k = 0 to j do
8:             for every node v ∈ V do
9:                 if dist_v^k = ∞ then
10:                    continue
11:                end if
12:                for every neighbor u of v do
13:                    if w(e_{u,v}) > w(e_i) then
14:                        continue
15:                    else if w(e_{u,v}) == w(e*) then
16:                        if dist_v^k + w(e_{u,v}) <
                           dist_u^{k+1} then
17:                            dist_u^{k+1} = dist_v^k +
                               w(e_{u,v})
18:                            Π_u^{k+1} = v
19:                        end if
20:                    else
21:                        if dist_v^k + w(e_{u,v}) < dist_u^k
                           then
22:                            dist_u^k = dist_v^k +
                               w(e_{u,v})
23:                            Π_u^k = v
24:                        end if
25:                    end if
26:                end for
27:            end for
28:        end for
29:    end for
30:    for i = 1 to n − 1 do
31:        ans_v = min{ans_v, dist_v^i + i * w(e_i)}
32:    end for
33: end for
```

Proof: By *Lemma* 3, if P_v is obtained when computing $G_{e^* P_v}$. Then, by considering all possible G_{e^*}, we could get P_v in one of these layering. It takes $O(m)$ to generate all G_{e^*}, by *Lemma* 4, MSPA runs in $v \in V$ in $O(n^2m^2)$ time. ∎

C. Computational Complexity of Disjoint Path Problem

In this section, we study edge disjoint path in optical wireless network. By reduction from well known **Min-Max 2-Path Problem**, i.e., min-max 2 edge disjoint path problem under normal length measurement, we show it is also NP-complete if we try to minimize the longer path when w^+ length is applied. Then we give an ILP formulation to solve this problem optimally. At last, we provide two approximation algorithm, one with approximation ratio 4, running time

$O((m+n)logn)$, the other one with approximation ratio 2 while running time is $O(m(m+n)logn)$.

Min-Max 2 Disjoint Path Problem (MinMax2PP)
Instance: An undirected graph $G = (V, E)$ with a positive weight $w_(e)$ associated with each edge $e \in E$, a source node $s \in V$, a destination node $t \in V$, and a positive number X.
Question: Does there exist a pair of edge disjoint paths P_1 and P_2 from s to d in G such that $w(P_1) \leq w(P_2) \leq X$?

The **MinMax2PP** problem is shown to be NP-complete in [10]. With a small modification, we show NP-completeness still holds if w^+ length measurement is adopted.

Min-Max 2 Disjoint Path Problem in Optical Wireless Networks (MinMax2OWFN)
Instance: An undirected graph $G = (V, E)$ with a positive weight $w_(e)$ associated with each edge $e \in E$, a source node $s \in V$, a destination node $t \in V$, and a positive number X.
Question: Does there exist a pair of edge disjoint paths P_1' and P_2' from s to t in G such that $w^+(P_1') \leq w^+(P_2') \leq X'$?

Theorem 3. The **MinMax2OWFN** is NP-complete

Proof: Evidently, **MinMax2OWFN** is in NP class, given two edge joint path P_1' and P_2', we can check if $w^+(P_1') \leq w^+(P_2') \leq X'$ in polynomial time.

We then transfer from **MinMax2PP** to **MinMax2OWFN**. Let graph $G = (V, E)$ with source node s , destination t and an integer X be an instance of MinMax2PP, we construct an instance G' of MinMax2OWFN in following way.

1) Create an identical graph G' with same nodes and edges in G.
2) Add one node s_0 to G'.
3) Create two parallel edges e_{01}, e_{02} between s_0 and s, $w(e_{01}) = w(e_{02}) = \max_{e \in G(E)} w(e)$
4) Choose s_0 to the source node in G' and t to be the destination.
5) Set $X' = X + 2w(e_{01})$

It is easy to see, the construction takes polynomial time.

Now we need to show a instance of **MinMax2OWFN** have two edge disjoint paths from s_0 to t with length at most X' if and only if the corresponding instance have two edge disjoint paths from s to t with length at most X.

Suppose there are two edge disjoint paths P_1' and P_2' from s_0 to t in G', such that $w^+(P_1') \leq w^+(P_2') \leq X'$. By the way we construct G', P_1' and P_2' must go through e_{01} and e_{02}. W.l.o.g. we say $e_{01} \in P_1'$ and $e_{02} \in P_2'$. Since $w(e_{01}) = w(e_{02}) = \max_{e \in E(G')}\{w(e)\}$, therefore e_{01} and e_{02} are the crucial edge on P_1' and P_2' respectively. Hence, $P_1' - e_{01}$ and $P_2' - e_{02}$ are two edge disjoint path in G, with length no greater than $X' - 2w(e_{01}) = X$.

Conversely, now suppose P_1 and P_2 are two edge joint paths in G satisfying $w(P_1) \leq w(P_2) \leq X$. We follow the same argument above, $P_1 + e_{01}$ and $P_2 + e_{02}$ are two desired paths, with length not exceeding $X + 2w(e_{01}) = X'$. ■

D. Optimal Solution for the Disjoint Path Problem

Here, we give an ILP formulation for **MinMax2OWFN**.

ILP for MinMax2OWFN

$$\min \quad MP$$

s.t.

$$\sum_{(i,j)\in E} f_{i,j,1} - \sum_{(j,i)\in E} f_{j,i,1} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & otherwise \end{cases}$$

$$\sum_{(i,j)\in E} f_{i,j,2} - \sum_{(j,i)\in E} f_{j,i,2} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & otherwise \end{cases}$$

$$f_{i,j,1} + f_{i,j,2} \leq 1 \qquad \forall (i,j) \in E$$

$$w_1 \geq f_{i,j,1} * w(i,j) \qquad \forall (i,j) \in E$$

$$w_2 \geq f_{i,j,2} * w(i,j) \qquad \forall (i,j) \in E$$

$$MP \geq w_1 + \sum_{(i,j)\in E} f_{i,j,1} * w(i,j)$$

$$MP \geq w_2 + \sum_{(i,j)\in E} f_{i,j,2} * w(i,j)$$

$$f_{i,j,1} = \{0,1\}, \quad f_{i,j,2} = \{0,1\} \qquad \forall (i,j) \in E$$

The following is a brief description of this ILP formulation. The first two equation represent flow constraint as normal shortest path problem does. $f_{i,j,1} = 1$ indicates path P_1 goes through edge (i, j), and 0 otherwise. So it is with $f_{i,j,2}$ and path P_2. Constraint 3 ensures two edges are disjoint, since $f_{i,j,1}$ and $f_{i,j,2}$ cannot both be 1 at the same time. w_1, w_2 act as the weights of the crucial edges on P_1 and P_2 respectively. Finally, we define MP to be the maximum of $w^+(P_1)$ and $w^+(P_2)$ and therefore try to minimize it.

E. Approximation Algorithm for the Disjoint Path Problem, with approximation factor 4

Next we propose a 4-approximation algorithm which runs in $O((n+m)logn)$ time.

Given $G = (V, E)$ with source s and destination t, the idea of approximation algorithm is to find two disjoint P_1 and P_2 such that $w(P_1) + w(P_2)$ is minimized. Such P_1 and P_2 can be found either using min cost max flow algorithm or the algorithm due to Suurballe presented in [11]. And we need to show both $w^+(P_1)$ and $w^+(P_2)$ are at most four times of the optimal solution.

Algorithm 3 MinMax2OWFN Approximation Algorithm 1 (MAA1)

1: Run Suurballe's algorithm on G, denote P_1, P_2 be two resulting path.
2: Compute $w^+(P_1)$ and $w^+(P_2)$.
3: Output $\max\{w^+(P_1), w^+(P_2)\}$.

Lemma 5. *For any path P, $w^+(P) \leq 2w(P)$.*

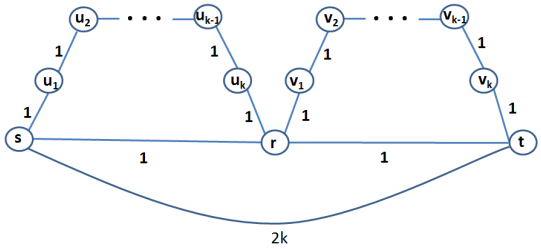
Proof: By definition, $w^+(P) = w(P) + w(e^*(P))$. Since $w(e^*(P)) \leq w(P)$, then $w^+(P) \leq 2 * w(P)$. ■

Lemma 6. *If P_1 and P_2 are two edge joint path from s to t such that $w(P_1) + w(P_2)$ is minimum, then $w^+(P_1)$ and $w^+(P_2)$ are at most four times of the optimal solution.*

Proof: Say opt is the optimal value of a $MinMax2OWFN$ instance and Q_1,Q_2 are two $s-t$ edge disjoint path in one optimal solution. W.l.o.g, we may suppose $w^+(P_1) \geq w^+(P_2)$ and $w^+(Q_1) \geq w^+(Q_2)$. Let $w(P_1)+w(P_2) = p$ and $w(Q_1)+w(Q_2) = q$, by assumption, $p \leq q$. Also, we have $w^+(P_1) = w(P_1) + e^*(P_1) \leq 2p$, $opt = w^+(Q_1) = w(Q_1) + e^*(Q_1) > \frac{q}{2}$. Hence, $\frac{w^+(P_2)}{opt} \leq \frac{w^+(P_1)}{opt} < \frac{2p}{q/2} \leq 4$ ■

Theorem 4. *MAA1 is a 4-approximation algorithm running in $O((n+m)logn)$ time and 4 is a tight bound.*

Proof: By $Lemma$ 5 and 6, MAA1 has approximation ratio at most 4.Then we show MAA1 has approximation at least 4 for certain cases. Consider the following graph.



It is easy to check, $P_1 = \{s \to t\}$, $P_2 = \{s \to r \to t\}$ are two edge disjoint path with minimum length $2k+2$, $w^+(P_1) = 4k > w^+(P_2) = 3$. However, let $Q_1 = \{s \to u_1 \to u_2 \to ... \to u_{k-1} \to u_k \to r \to t\}$, $Q_2 = \{s \to r \to v_1 \to v_2 \to ... \to v_{k-1} \to v_k \to t\}$, then $w(Q_1) + w(Q_2) = 2k+4$ while $w^+(Q_1) = w^+(Q_2) = k+3$. $\frac{w^+(P_1)}{w^+Q_1} = \frac{4k}{k+3} \approx 4$ when k is sufficiently large. Hence, 4 is a tight bound for MAA1.

We need $O((n+m)logn)$ time running Suurballe's algorithm and $O(n)$ time computing $w^+(P_1)$ and $w^+(P_2)$. Therefore total running time is $O((n+m)logn)$.

■

F. Approximation Algorithm for the Disjoint Path Problem, with approximation factor 2

In MAA1, layering technique is not used and we only consider the original graph. However, by taking all G_e of G into account, we can have a better approximation ratio.

Say Q_1, Q_2 are two disjoint paths in one optimal solution. Let $e' = \max\{e^*(Q_1), e^*(Q_2)\}$ and P_1, P_2 be the resulting paths when computing layer $G_{e'}$; w.l.o.g, we may assume $w(P_1) > w(P_2)$. Also, let $ans_{e'} = \max\{w^+(P_1),\ w^+(P_2)\}$.

Lemma 7. $ans_{e'} < 2\max\{w^+(Q_1),\ w^+(Q_2)\}$.

Proof: Noting that $w(e^*(P_1)) \leq w(e')$ and $w(e^*(P_2)) \leq w(e')$ since they both belong to $G_{e'}$. Then $ans_{e'} \leq w(P_1) + w(e')$. It suffices to show $\frac{w(P_1)+w(e')}{\max\{w^+(Q_1),\ w^+(Q_2)\}} < 2$. We prove it by contradiction. Suppose $\frac{w(P_1)+w(e')}{\max\{w^+(Q_1),\ w^+(Q_2)\}} \geq 2$, then

$$w(P_1) + w(e') \geq w^+(Q_1) + w^+(Q_2)$$

Which follows,

$$w(P_1) + w(e') \geq w(Q_1) + w(e^*(Q_1)) + w(Q_2) + w(e^*(Q_2))$$

By definition, e' is one of $e^*(Q_1)$, $e^*(Q_2)$. Hence,

$$w(P_1) > w(Q_1) + w(Q_2)$$

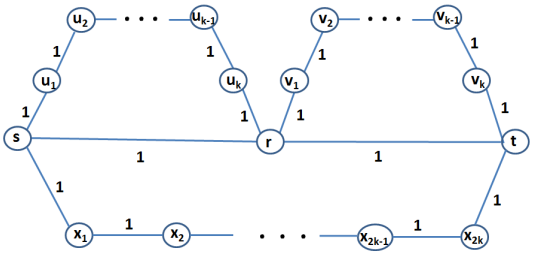
It is impossible since $w(P_1) + w(P_2)$ is minimum in layer $G_{e'}$. ■

Algorithm 4 MinMax2OWFN Approximation Algorithm 2(MAA2)

```
set ans = ∞
for every G_e of G do
    Run Suurballe's algorithm on G_e, denote P_1, P_2 be two resulting path.
    Compute w^+(P_1) and w^+(P_2).
    ans = min{ans, max{w^+(P_1), w^+(P_2)}}.
end for
Output ans.
```

Theorem 5. *MAA2 is a 2-approximation algorithm running in $O(m(n+m)logn)$ time and 2 is a tight bound.*

Proof: By $Lemma$ 7, in one of the layer, we guarantee to have a 2-approximation solution. Since we take minimum outcome among all layers, the final result is no worse than twice of the optimal solution. Now we need to show there exists certain case, such that MAA2 is no good than twice of the optimal solution. Consider the following graph



There is only one layer, and $P_1 = \{s \to x_1 \to x_2 \to ... \to x_{2k-1} \to x_2k \to t\}$, $P_2 = \{s \to r \to t\}$ are two edge disjoint path with minimum length $2k+3$, $w^+(P_1) = 2k+2 > w^+(P_2) = 3$. Again, set $Q_1 = \{s \to u_1 \to u_2 \to ... \to u_{k-1} \to u_k \to r \to t\}$, $Q_2 = \{s \to r \to v_1 \to v_2 \to ... \to v_{k-1} \to v_k \to t\}$, then $w(Q_1) + w(Q_2) = 2k+4$ while $w^+(Q_1) = w^+(Q_2) = k+3$. $\frac{w^+(P_1)}{w^+Q_1} = \frac{2k+2}{k+3} \approx 2$ when k is sufficiently large. Hence, 2 is a tight bound for MAA2.

Finally, it is easy to see that the running time is $O(m(n+m)logn)$. ■

S node	D node	Opt Sol	$Approx$ Sol 1	$Approx$ $ratio$ 1	$Approx$ Sol 2	$Approx$ $ratio$ 2
14	2	47	55	1.17	55	1.17
18	8	46	46	1	46	1
1	6	28	28	1	28	1
18	4	50	58	1.16	57	1.14
20	3	40	40	1	40	1
10	3	27	27	1	27	1
1	11	35	35	1	35	1
14	6	50	52	1.04	52	1.04
20	7	38	38	1	38	1
10	5	36	38	1.05	38	1.05
18	12	22	22	1	22	1
1	20	46	52	1.13	52	1.13
20	13	26	26	1	26	1
14	19	29	29	1	29	1
10	17	36	36	1	36	1
20	16	29	29	1	29	1
5	11	40	48	1.2	48	1.2

TABLE I
COMPARISON OF THE APPROXIMATE SOLUTIONS WITH THE OPTIMAL SOLUTION FOR THE ARPANET GRAPH

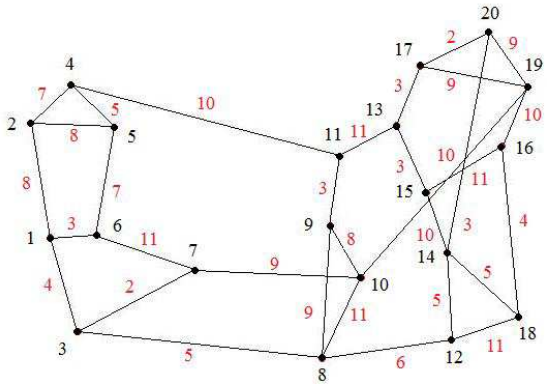


Fig. 1. The ARPANET graph with 20 nodes and 32 links

G. Experimental Results for the Disjoint Path Problem

In this section, we present the results of simulations for comparing the performance of our approximation algorithms with the optimal solution when $w^+()$ metric is applied. The simulation experiments have been carried out on the ARPANET topology (as shown in Fig 1 with nodes and links shown in black) which has twenty nodes and thirty two links. The weights of the links have been randomly generated and lie in the range of two and eleven (as shown in red in Fig 1) and we consider the graph to be undirected. The results of the comparison is presented in Table I. We have compared the lengths of the longer of the two edge disjoint paths computed by the optimal and the approximate solutions for seventeen different source-destination pairs. It may be noted that for almost 65% of the cases, the approximate algorithms obtain the optimal solution. In the remaining cases, the approximate solutions lie within a factor of 1.2 of the optimal solution Thus, even though the approximation ratio in the worst case are proven to be 4 and 2, in practical cases, it is within 1.2. From these experimental results, we can conclude that the approximation algorithms produce optimal or near optimal solutions in majority of the cases. It may be noted that the two approximation algorithms perform in a similar fashion in the ARPANET graph, however, as proven theoretically, the two approximation algorithms differ in their worst case approximation ratio.

V. CONCLUSION

In this paper, we study the shortest path problem in FiWi networks. Based on the path length metrics proposed in [3], [5], we present polynomial time algorithms for the single path scenario. In the disjoint path scenario, we prove that the problem of finding a pair of disjoint paths, where the length of the longer path is shortest, is NP-complete. We provide an ILP solution for the disjoint path problem and propose two approximation algorithms. Both the approximation algorithms have a constant factor approximation bound. However, there is a trade-off between the quality of the solution (approximation bound) and the execution time. Finally, we show that both the approximation algorithms obtain near optimal results through simulation using the ARPANET topology.

REFERENCES

[1] N. Ghazisaidi, M. Maier, and C. M. Assi, "Fiber-Wireless (FiWi) Access Networks: A Survey", *IEEE Communications Magazine*, vol. 47, no. 2, pp 160-167, Feb. 2009.

[2] N. Ghazisaidi, and M. Maier, "Fiber-Wireless (FiWi) Access Networks: Challenges and Opportunities", *IEEE Network*, vol. 25, no. 1, pp 36-42, Feb. 2011.

[3] Z. Zheng, J. Wang, X. Wang, "ONU placement in fiber-wireless (FiWi) networks considering peer-to-peer communications", *IEEE Globecom*, 2009.

[4] Z. Zheng, J. Wang, X. Wang, "A study of network throughput gain in optical-wireless (FiWi) networks subject to peer-to-peer commuincations", *IEEE ICC*, 2009.

[5] F. Aurzada, M. Levesque, M. Maier, M. Reisslein, "FiWi Access Networks Based on Next-Generation PON and Gigabit-Class WLAN Technologies: A Capacity and Delay Analysis", *IEEE/ACM Transactions on Networking*, to appear.

[6] A. Sen, B.Hao . B. Shen , L.Zhou and S. Ganguly, "On maximum available bandwidth through disjoint paths", *Proc. of IEEE Conf. on High Performance Switching and Routing*, 2005.

[7] M. Mosko, J.J. Garcia-Luna-Aceves, "Multipath routing in wireless mesh networks", *Proc. of IEEE Workshop on Wireless Mesh Networks*, 2005.

[8] J. Wang, K. Wu, S. Li and C. Qiao ,"Performance Modeling and Analysis of Multi-Path Routing in Integrated Fiber-Wireless (FiWi) Networks", *IEEE Infocom mini conference*, 2010.

[9] S. Li, J. Wang, C. Qiao, Y. Xu ,"Mitigating Packet Reordering in FiWi Networks", *IEEE/OSA Journal of Optical Communications and Networking*, vol. 3, pp.134-144, 2011.

[10] C. Li, S.T. McCormick and D.Simchi-Levi, "Complexity of Finding Two Disjoint Paths with Min- Max Objective", *Discrete Applied Mathematics*, vol. 26, pp. 105-115, 1990.

[11] J. W. Suurballe, "Disjoint paths in a network", *Networks*, vol. 4, pp. 125-145, 1974.