On the Min-Max-Delay Problem:
NP-completeness, Algorithm, and Integrality Gap

Qingyu Liu*, Lei Deng†, Haibo Zeng*, Minghua Chen†

*Department of Electrical and Computer Engineering, Virginia Tech, USA

†Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract—We study a delay-sensitive information flow problem where a source streams information to a sink over a directed graph $G \triangleq (V, E)$ at a fixed rate R possibly using multiple paths to minimize the maximum end-to-end delay, denoted as the **Min-Max-Delay** problem. Transmission over an edge incurs a constant delay within the capacity. We prove that **Min-Max-Delay** is weakly NP-complete, and demonstrate that it becomes strongly NP-complete if we require integer flow solution. We propose an optimal pseudo-polynomial time algorithm for **Min-Max-Delay**, with time complexity $O(\log(Nd_{\max})(N^5 d_{\max}^{2.5})(\log R + N^2 d_{\max} \log(N^2 d_{\max})))$, where $N \triangleq \max\{|V|, |E|\}$ and $d_{\max}$ is the maximum edge delay. Besides, we show that the integrality gap, which is defined as the ratio of the maximum delay of an optimal integer flow to the maximum delay of an optimal fractional flow, could be arbitrarily large.

I. INTRODUCTION

Delay-sensitive network flows have strong applications in many domains, including communication networks, cyber-physical systems, transportation networks and evacuation planning [1]. In communication networks, video conferencing requires the video delivery delay to be no more than 250ms to ensure a good interactive conferencing experience [2], and packet delivery delay highly affects user experience and hence revenue in data center networks of cloud service providers like Amazon, Microsoft and Google [3]. In cyber-physical systems, shorter delay for control messages can improve control quality [4]. In transportation networks, timely delivery is critical to deliver perishable goods [5]. In evacuation planning, it is important to move all people from hazardous areas to safe areas as soon as possible [6].

Theoretically, there are mainly two different delay-sensitive flow models: the flow-amount model and the flow-rate model. The flow-amount model is suitable to the applications where the flow is generated once, while the flow-rate model is suitable to the applications where the flow is generated continuously. For both models, two different delay-sensitive network flow problems are important: maximizing the flow (amount or rate) subject to a maximum delay constraint, and minimizing the maximum delay subject to a flow (amount or rate) requirement. We summarize related studies in Tab. I.

For the flow-amount model, the first problem, called *dynamic flow problem* [7], is to maximize the flow amount to be delivered from a source to a sink within a given time horizon. The authors in [7] show that it can be formulated as a min-cost flow problem and thus solved in polynomial time by various min-cost flow algorithms. The second problem, called *quickest flow problem*, is to minimize the time horizon to deliver a given

TABLE I
DELAY-SENSITIVE NETWORK FLOW PROBLEMS.

	Max-flow subject to delay constraint	Min-max delay subject to flow requirement
Flow-amount Model	[7], [11]	[8], [12]
Flow-rate Model	[9], [13]	[10], [14], our paper

amount of flow from a source to a sink and is also solvable in polynomial time [8].

For the flow-rate model, the first problem, called *delay-constrained max-flow problem* [9], is to maximize the flow rate to be sent from a source to a sink while the end-to-end delay is bounded above by a given delay constraint. The problem has been proved to be NP-complete [9]. The second problem, called *maximum latency problem* [10], is to minimize the maximum end-to-end delay that flow units experience from a source to a sink while satisfying a given flow rate requirement and has been proved to be NP-complete [10].

This paper studies the **Min-Max-Delay** problem, which is similar to the maximum latency problem [10], [14] except for two key differences:

- In our **Min-Max-Delay** problem, each edge has an integer capacity such that the assigned flow rate cannot exceed the given capacity, while there is no capacity constraint in the maximum latency problem;
- Edge delay is an integer in **Min-Max-Delay**. But it is a flow-dependent function in the maximum latency problem.

These two differences can capture many applications that cannot be handled by the maximum latency problem. For example, in the communication networks, the flow-dependent queueing delay becomes negligible and thus the constant propagation delay dominates the edge delay in the light load scenario [15]. In the transportation networks, the ground vehicle speed (or equivalently the time to pass the road) remains nearly constant before reaching a certain flow rate [16]. Due to these two key differences, existing results on the maximum latency problem including the complexity analysis and approximation algorithms [10], [14] are not applicable to our problem.

In this paper, we make the following contributions:

▷ We prove that **Min-Max-Delay** is weakly NP-complete in Thm. 4 based on the results in Sec. III-A and Sec. IV.

▷ We propose a binary-search algorithm which can find the optimal solution to **Min-Max-Delay** in pseudo-polynomial time in Sec. IV. The time complexity of the algorithm is $O(\log(Nd_{\max})(N^5 d_{\max}^{2.5})(\log R + N^2 d_{\max} \log(N^2 d_{\max})))$, where $N \triangleq \max\{|V|, |E|\}$, R is the rate requirement, and

$d_{\max}$ is the maximum edge delay. The complexity is pseudo-polynomial in the sense that it is polynomial in the numeric value of the problem input $d_{\max}$, but is exponential in the bit length of the problem input, i.e., $\log(d_{\max})$ [17].

▷ We prove that Min-Max-Delay becomes strongly NP-complete if each path can only have an integer flow in Thm. 2 of Sec. III-B.

▷ In Sec. V, we further construct a network to show that the integrality gap, which is defined as the ratio of the maximum delay of an optimal integer flow to the maximum delay of an optimal fractional flow, could be arbitrarily large. This example illustrates additional intriguing difficulties for the integer version of Min-Max-Delay problem.

II. System Model and Problem Formulation

We consider a network modeled as a directed graph $G \triangleq (V, E)$ with $|V|$ nodes and $|E|$ edges. We define $N \triangleq \max\{|V|, |E|\}$. Each edge $e \in E$ has a non-negative integer capacity c_e and a non-negative integer delay d_e. We define $d_{\max} \triangleq \max_{e \in E} d_e$. A source node $s \in V$ needs to send a positive integer rate R to a sink node $t \in V \backslash \{s\}$.

We denote P as the set of all paths from s to t. For any path $p \in P$, we denote its path delay as $d^p \triangleq \sum_{e \in E: e \in p} d_e$. A flow solution f is defined as the assigned flow rate over P, i.e., $f \triangleq \{f^p : f^p \geq 0, p \in P\}$. For a flow solution f, we define $f_e \triangleq \sum_{p \in P: e \in p} f^p$ as the flow rate on edge $e \in E$. We further define the *maximum delay* of a flow solution f as

$$\mathcal{D}(f) \triangleq \max_{p \in P: f^p > 0} d^p, \tag{1}$$

i.e., the maximum delay among paths that carry *positive* rates.

We consider the problem of finding a flow solution f to minimize the maximum delay $\mathcal{D}(f)$ while satisfying both the rate requirement and edge capacity constraints. We denote the problem as Min-Max-Delay. It is formulated as

$$\min \quad D \tag{2a}$$
$$\text{s.t.} \quad \sum_{p \in P} f^p = R, \tag{2b}$$
$$f_e = \sum_{p \in P: e \in p} f^p \leq c_e, \quad \forall e \in E, \tag{2c}$$
$$f^p (d^p - D) \leq 0, \quad \forall p \in P, \tag{2d}$$
$$\text{vars.} \quad f^p \geq 0, \quad \forall p \in P. \tag{2e}$$

where (2a) together with (2d) define our objective to minimize the maximum path delay for $s-t$ paths that carry positive rates (called *flow-carrying paths*). Constraint (2b) restricts that the source s sends R rate to the sink t, and constraint (2c) requires that the flow rate on edge e does not exceed its capacity c_e.

From formulation (2) we observe two difficulties to solve Min-Max-Delay: (i) The number of paths (number of variables) can exponentially increase w.r.t. the network size, and (ii) formulation (2) is non-convex due to constraint (2d).

The integer version of problem Min-Max-Delay, denoted as Int-Min-Max-Delay, further requires that each path carries an integer flow rate, i.e., replacing (2e) by

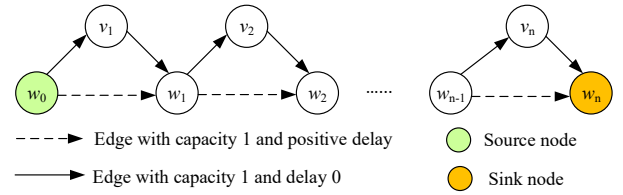$$\text{vars.} \quad f^p \in \mathbb{Z}^+, \quad \forall p \in P. \tag{3}$$

Fig. 1. Reduced network graph from partition problem.

We let f_{MM} (resp. f_{IMM}) to be an optimal flow solution to problem Min-Max-Delay (resp. Int-Min-Max-Delay). Then, we define the integrality gap as,

$$\mathsf{Int\text{-}Gap} \triangleq \mathcal{D}(f_{\mathsf{IMM}})/\mathcal{D}(f_{\mathsf{MM}}), \tag{4}$$

which is the maximum delay ratio of the integer flow solution f_{IMM} to the possibly fractional flow solution f_{MM}.

III. NP-completeness

In this section, we analyze the computational complexity of our two problems, Min-Max-Delay and Int-Min-Max-Delay. In Sec. III-A, we prove that Min-Max-Delay is NP-complete based on the polynomial reduction from the NP-complete partition problem [17]. In Sec. III-B, we further prove that Int-Min-Max-Delay is NP-complete in the strong sense based on the pseudo-polynomial transformation from the classic strongly NP-complete 3-partition problem [17].

A. NP-completeness for Min-Max-Delay

To analyze the computational complexity of Min-Max-Delay, we first define partition and the partition problem.

Definition 1 (Partition). *Given a non-empty set A, its partition is a set of non-empty subsets such that each element in A is in exactly one of these subsets.*

Definition 2 (Partition Problem [17]). *Given a set of n positive integers $A = \{a_1, a_2, ..., a_n\}$ with sum $\sum_{a_i \in A} a_i = 2b$. Is there a partition $\{A_1, A_2\}$ of A such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j = b$?*

The partition problem is known to be NP-complete [17] (in the weak sense). We now leverage it to prove that Min-Max-Delay is NP-complete.

Theorem 1. *The decision version of Min-Max-Delay problem is NP-complete.*

Proof: For any partition problem we construct a graph G' with $(2n+1)$ nodes and $3n$ edges as in Fig. 1. All edges have unit capacity. Each edge (w_{i-1}, w_i) has a delay of a_i for all $i = 1, \cdots, n$, while edges (w_{i-1}, v_i) and (v_i, w_i) have a delay of zero. Obviously it takes polynomial time to construct the graph G'. We then consider the following decision problem of Min-Max-Delay: for graph G' with source $s = w_0$, sink $t = w_n$, and flow rate requirement $R = 2$, is there any feasible flow f such that the maximum delay $\mathcal{D}(f) \leq b$?

Now we prove the partition problem answers "Yes" if and only if the decision version of Min-Max-Delay answers "Yes".

If Part. If the decision problem of Min-Max-Delay answers "Yes", then there exists a flow f such that $\mathcal{D}(f) \leq b$. Since f is feasible, the total rate from w_0 to w_n in f is $R = 2$. Now

due to the capacity constraint and flow conservation, all edges must exactly have a flow rate 1 to satisfy the requirement $R = 2$. The total delay in flow f is

$$\sum_{p \in P} f^p d^p = \sum_{e \in E} f_e d_e = \sum_{e \in E} 1 \cdot d_e = \sum_{i=1}^{n} a_i = 2b. \quad (5)$$

Since $\mathcal{D}(f) \leq b$, we have

$$d^p \leq \mathcal{D}(f) \leq b, \forall p \in P \text{ with } f^p > 0. \quad (6)$$

Also, because the total flow rate is equal to 2, we have

$$2b = \sum_{p \in P} f^p d^p = \sum_{p \in P: f^p > 0} f^p d^p \leq b \cdot \sum_{p \in P: f^p > 0} f^p = 2b. \quad (7)$$

As both ends in (7) are the same, it must be

$$d^p = b, \forall p \in P \text{ with } f^p > 0. \quad (8)$$

Therefore, all flow-carrying paths have a path delay of b. We choose an arbitrary flow-carrying path p. Since all solid edges have a delay of 0, the path delay of p is the delay of all dashed edges that belongs to p. We consider the set A_1 that contains a_i if edge $(w_{i-1}, w_i) \in p$. Clearly, it holds that $\sum_{a_i \in A_1} a_i = b$. We then define $A_2 = A \backslash A_1$. It shall be $\sum_{a_j \in A_2} = \sum_{a_k \in A} - \sum_{a_i \in A_1} = 2b - b = b$. A_1 and A_2 are thus a partition of set A and meet the requirement of the partition problem. Hence, the partition problem answers "Yes".

Only If Part. If the partition problem answers "Yes", then there exists a partition A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j = b$. We now construct two paths p_1 and p_2.

- $\forall i \in [1, n]$, if $a_i \in A_1$, we put edge (w_{i-1}, w_i) into path p_1; otherwise, we put (w_{i-1}, v_i) and (v_i, w_i) into p_1.
- $\forall i \in [1, n]$, if $a_i \in A_2$, we put (w_{i-1}, w_i) into p_2; otherwise, we put (w_{i-1}, v_i) and (v_i, w_i) into p_2.

Due to the definition of a partition (see Definition 1), A_1 and A_2 are two disjoint sets, i.e., $A_1 \cap A_2 = \emptyset$. Thus, we can easily see that p_1 and p_2 are two disjoint $s - t$ paths, i.e., p_1 and p_2 do not share any common edge. We can see that $d^{p_1} = \sum_{a_i \in A_1} = b$, and $d^{p_2} = \sum_{a_i \in A_2} = b$. We then construct the flow f with only two flow-carrying paths p_1 and p_2 and set $f^{p_1} = f^{p_2} = 1$. Since p_1 and p_2 are disjoint, the capacity constraint is satisfied. Also, since $f^{p_1} + f^{p_2} = R = 2$, the rate requirement is satisfied. Thus f is a feasible flow with maximum delay $\mathcal{D}(f) = b$. Therefore, the decision problem of Min-Max-Delay answers "Yes".

Since the partition problem is NP-complete [17] and the reduction can be done in polynomial time, the decision problem of Min-Max-Delay is also NP-complete. ∎

Due to the NP-completeness, it is impossible to solve Min-Max-Delay optimally in polynomial time unless P = NP. Later in Sec. IV, a pseudo-polynomial time algorithm is proposed to solve Min-Max-Delay optimally, which further proves that Min-Max-Delay is NP-complete in the weak sense.

B. Strong NP-completeness for Int-Min-Max-Delay

We now analyze the computational complexity of Int-Min-Max-Delay where only integer flows are allowed. We begin with the definition of the 3-Partition Problem [17].
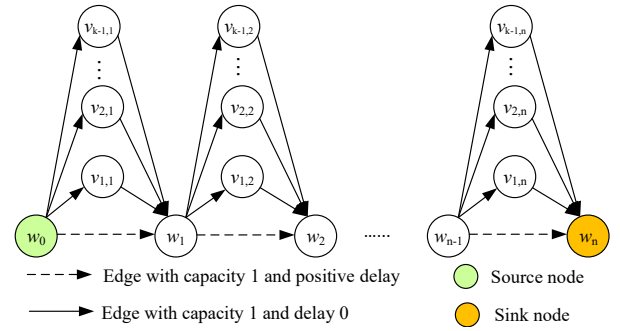

Fig. 2. Reduced network graph from 3-partition problem.

Definition 3 (3-Partition Problem [17])**.** *Given a set of* $n = 3k$ $(k > 0)$ *positive integers* $A = \{a_1, a_2, ..., a_n\}$ *with sum* $\sum_{a_i \in A} a_i = kb$ *and* $b/4 < a_i < b/2$ *for each element* a_i, *is there a partition* $\{A_1, A_2, ..., A_k\}$ *of set* A *such that for each subset* A_j, $\sum_{a_i \in A_j} a_i = b$?

The 3-partition problem has been proved to be NP-complete in the strong sense [17], which can be utilized to prove the strong NP-completeness of our problem Int-Min-Max-Delay.

Theorem 2. *The decision version of* Int-Min-Max-Delay *problem is NP-complete in the strong sense.*

Proof: For any 3-partition problem, we construct a network following Fig. 2. Edge capacity and delay are similarly defined as in the proof to Thm. 1. We set the flow requirement $R = k$ and thus get the problem Int-Min-Max-Delay. Such a reduction is a pseudo-polynomial transformation [17]. Similar to the proof in Thm. 1, we show that the 3-partition problem answers "Yes" if and only if the decision problem of Int-Min-Max-Delay answers "Yes". Due to the space limit, we put the detailed proof in our technical report [18]. ∎

As we will shown later in Thm. 4, Min-Max-Delay is weakly NP-complete. Thus, Thm. 2 shows that Int-Min-Max-Delay is more difficult than Min-Max-Delay. Later in Sec. V we show that the integrality gap defined in (4) could be arbitrarily large, which further illustrates additional intriguing difficulties for the Int-Min-Max-Delay problem.

IV. OPTIMAL PSEUDO-POLYNOMIAL TIME ALGORITHM

In this section we propose a pseudo-polynomial time algorithm to solve Min-Max-Delay optimally. This, combined with the NP-completeness result in Thm. 1, shows that Min-Max-Delay is NP-complete in the weak sense.

A closely-related problem to Min-Max-Delay is the Delay-Constrained Maximum Flow problem [13], denoted as DC-Max-Flow: for the same graph G and a given deadline constraint T, finding the maximum flow (rate) such that the delay of any flow-carrying path does not exceed T. Let us denote P^T as the set of all $s - t$ paths whose path delay does not exceed T. Then DC-Max-Flow can be formulated as

$$\max \quad \sum_{p \in P^T} f^p, \quad (9a)$$

$$\text{s.t.} \quad f_e = \sum_{p \in P^T: e \in p} f^p \leq c_e, \quad \forall e \in E \quad (9b)$$

$$\text{vars.} \quad f^p \geq 0, \quad \forall p \in P^T. \quad (9c)$$

It is shown that DC-Max-Flow can be solved with an edge-based flow formulation in pseudo-polynomial time [13]. They implicitly use the idea of time-expanded graph [19] by converting a delay-constrained max-flow problem in the original graph into a delay-unconstrained max-flow problem in the expanded graph. In [13], the authors only consider unit-delay edges, but it is easy to generalize their result to integer-delay edges. Due to the space limit, we omit the procedure of constructing the expanded graph and directly give the equivalent edge-based flow formulation for DC-Max-Flow as follows [13, Proposition 1],

$$\max \quad \sum_{e \in \mathsf{In}(t)} \sum_{d=0}^{T} f_e^{(d)} \tag{10a}$$

$$\text{s.t.} \quad \sum_{e \in \mathsf{Out}(s)} f_e^{(d_e)} = \sum_{e \in \mathsf{In}(t)} \sum_{d=0}^{T} f_e^{(d)}, \tag{10b}$$

$$\sum_{e \in \mathsf{In}(v)} f_e^{(d)} = \sum_{e \in \mathsf{Out}(v)} f_e^{(d+d_e)},$$

$$\forall v \in V \backslash \{s, t\}, d \in [0, T] \tag{10c}$$

$$\sum_{d=0}^{T} f_e^{(d)} \leq c_e, \quad \forall e \in E \tag{10d}$$

$$\text{vars.} \quad f_e^{(d)} \geq 0, \quad \forall e \in E, d \in [0, T] \tag{10e}$$

where $\mathsf{In}(v) \triangleq \{e = (w, v) : e \in E, w \in V\}$ is the set of incoming edges of node v, $\mathsf{Out}(v) \triangleq \{e = (v, w) : e \in E, w \in V\}$ is the set of outgoing edges of node v, and $f_e^{(d)}$ is the total flow rate that experiences a delay of d after passing edge e from the source s. The objective (10a) is the total flow rate that arrives at the sink t within the delay bound T. Eqs. (10b) requires the rate entering the network should equal to the rate leaving the network. Eqs. (10c) are the flow conservation constraints in the expanded graph. Note that by convention, for any edge $e \in E$, we set $f_e^{(d)} = 0$ for $d < 0$ and $d > T$. Eqs. (10d) are the edge capacity constraints.

We now show the relationship between our problem Min-Max-Delay and problem DC-Max-Flow. For a graph G, denote $d^*(R)$ as the minimum maximum delay with rate requirement R (the optimal value of Min-Max-Delay). For the same graph G and a non-negative integer T, denote $r^*(T)$ as the maximum flow subject to a delay constraint T (the optimal value of DC-Max-Flow). We have the following lemma.

Lemma 1. $d^*(R) \leq T$ if and only if $r^*(T) \geq R$.

Proof: **If Part.** If $r^*(T) \geq R$, then there exists a flow solution over P^T such that $\sum_{p \in P^T} f^p \geq R$. We can thus decrease the flow solution f to construct another flow solution $\tilde{f}$ such that $\sum_{p \in P^T} \tilde{f}^p = R$. Since f satisfies the capacity constraints, $\tilde{f}$ must also satisfy the capacity constraints. Thus, $\tilde{f}$ is a feasible solution to Min-Max-Delay with rate requirement R. In addition, since all flow-carrying paths in $\tilde{f}$ belong to the set P^T, we have $d^*(R) \leq \mathcal{D}(\tilde{f}) \leq T$.

Only If Part. If $d^*(R) \leq T$, then there exists a flow solution f where the path delay of any flow-carrying path does not exceed T. Thus all flow-carrying paths belong to P^T and f is
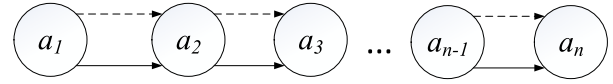

Fig. 3. A building-block graph with $n \geq 3$ to show that the integrality gap could be arbitrarily large. All edges have unit capacity. Each upper dashed edge has a unit delay while each lower solid edge has a delay of zero.

also a feasible solution to DC-Max-Flow with a delay bound T. Thus, $r^*(T) \geq \sum_{p \in P^T} f^p = R$. ∎

Lem. 1 suggests a binary-search algorithm to solve Min-Max-Delay optimally. Given a lower bound T_l ($= 0$ initially) and an upper bound T_u ($= |E|d_{\max}$ initially) of the optimal maximum delay, in each iteration we solve problem (10) with $T = \lceil (T_l + T_u)/2 \rceil$. We then compare $r^*(T)$ with the rate requirement R. If $r^*(T) \geq R$, we update the upper bound as $T_u = T$. Otherwise we update the lower bound as $T_l = T+1$. The algorithm terminates when $T_l \geq T_u$.

Theorem 3. *The binary search scheme solves Min-Max-Delay problem optimally and has a pseudo-polynomial time complexity* $O(\log(Nd_{\max})(N^5 d_{\max}^{2.5})(\log R + N^2 d_{\max} \log(N^2 d_{\max})))$.

Proof: The optimality of the binary search scheme directly follows from Lem. 1. For the time complexity analysis, please see our technical report in [18]. ∎

The time complexity shown above is pseudo-polynomial in the sense that it is polynomial in the numeric value of the input $d_{\max}$, but it is exponential in the bit length of the $d_{\max}$. Thus, our binary search algorithm could still have high complexity for the graph with large $d_{\max}$. To further reduce the complexity, we adapt the *rounding and scaling* approach [20] and design a fully polynomial time approximation scheme (FPTAS) for our problem Min-Max-Delay. For any $\epsilon > 0$, our proposed FPTAS can find a $(1 + \epsilon)$-approximate solution and the time complexity is polynomial in both problem size and $1/\epsilon$. Due to the space limit, the detailed FPTAS design is shown in our technical report [18].

A direct result of Thm. 3 is as follows.

Theorem 4. *The decision version of Min-Max-Delay is NP-complete in the weak sense.*

Proof: It follows from Thm. 1 and Thm. 3. ∎

V. Integrality Gap

We have proved that Min-Max-Delay allowing fractional flow is NP-complete in the weak sense, but Int-Min-Max-Delay only allowing integer flow solution is NP-complete in the strong sense. Thus, there are some fundamental differences between these two problems. To elaborate their differences, in this section, we construct a network to show that the integrality gap defined in (4) could be arbitrarily large.

We will use the network in Fig. 3 as a building block to construct the network with arbitrarily large integrality gap. Towards that end, for the building-block graph in Fig. 3 with source node a_1 and sink node a_n, we first analyze the maximum delay of problem Int-Min-Max-Delay and problem Min-Max-Delay in Lem. 2 and Lem. 3, respectively.

Lemma 2. *The Int-Min-Max-Delay flow in Fig. 3 given $R = 2$ has a maximum delay of $\lceil \frac{n-1}{2} \rceil$.*

Proof: Because of the unit capacity constraint for each edge, any flow-carrying path will be assigned a unit flow rate in the integer flow from a_1 to a_n.

Because of the rate requirement $R = 2$, there are exactly two flow-carrying paths in the Int-Min-Max-Delay flow and each of them carries a unit flow rate.

Moreover, because of the capacity constraint, the two flow-carrying paths must be disjoint, namely they share no edges.

Therefore, since the Int-Min-Max-Delay flow minimizes the maximum flow-carrying path delay among all integer flows, the two flow-carrying paths will have path delay of $\lfloor (n-1)/2 \rfloor$ and $\lceil (n-1)/2 \rceil$ respectively in the Int-Min-Max-Delay flow, leading to a maximum delay of $\lceil (n-1)/2 \rceil$. ∎

Lemma 3. *The Min-Max-Delay flow in Fig. 3 given $R = \frac{n-1}{n-2}$ has a maximum delay of 1.*

Proof: First it is straightforward that the Min-Max-Delay flow has a maximum delay no smaller than 1 since $R > 1$.

Next, we will explicitly construct a feasible flow with unit maximum delay: Note that there are $(n-1)$ different $a_1 - a_n$ paths containing exactly one dashed edge. We then place $1/(n-2)$ flow rate on each of these $(n-1)$ paths with the rate requirement $R = (n-1)/(n-2)$ satisfied. It is easy to verify that edge capacity constraints are satisfied, too. ∎

Note that even though our system model in Sec. II requires an integer rate R, problem Min-Max-Delay is also well defined for fractional rate requirement R, which is the case in Lem. 3. However, later in our constructed graph to show that the integrality gap can be arbitrarily large (see the proof in Thm. 5), we set integer flow rate requirement R, which is in line with our system model in Sec. II.

Based on Lem. 2 and Lem. 3, we use Fig. 3 as a building-block to construct a network with integer edge delay, integer capacity, integer rate R and arbitrarily large Int-Gap.

Theorem 5. *There exists a Min-Max-Delay problem instance such that Int-Gap could be arbitrarily large.*

Proof: We place $(n-2)$ subnetworks with the topology of Fig. 3 side by side disjointly and then connect each of them to both a single source s and a single sink t. All the outgoing edges of s and the incoming edges of t have a capacity of 2 and a delay of 0. We set the rate requirement $R = n - 1$.

First, considering Lem. 3, the Min-Max-Delay flow has a maximum delay of 1 because we can route $(n-1)/(n-2)$ flow rate to each of the $(n-2)$ subnetwork.

Next since the rate requirement $R = n - 1$ is strictly larger than $n - 2$ which is the number of subnetworks, for any feasible integer flow, there will be at least one subnetwork who is assigned a flow rate of 2. According to Lem. 2, corresponding minimal maximum delay to pass the subnetwork is $\lceil (n-1)/2 \rceil$. Thus, the Int-Min-Max-Delay flow will have a maximum delay of $\lceil (n-1)/2 \rceil$.

Overall, the Int-Gap is $\lceil (n-1)/2 \rceil$. Since n can be arbitrarily large, the gap can be arbitrarily large. ∎

Comparing f_{IMM} to f_{MM} under the same rate R, Int-Gap can be infinitely large according to Thm. 5. However, for any $\epsilon \in (0,1)$, Int-Gap is in fact upper bounded by $1/\epsilon$ if we instead compare f_{IMM} with a smaller rate $(1-\epsilon)R$ to f_{MM} with the full rate R, as proved in our report [18]. Overall, Int-Min-Max-Delay is different from Min-Max-Delay. Though we have designed an optimal pseudo-polynomial time algorithm and an FPTAS for Min-Max-Delay, we need to tackle some additional intriguing difficulties to design efficient algorithms for the strongly NP-complete problem Int-Min-Max-Delay.

VI. Conclusion

We study the Min-Max-Delay which minimizes the maximum end-to-end delay under a rate requirement in a single-unicast scenario. We prove it is NP-complete in the weak sense, and propose a pseudo-polynomial time algorithm to find its optima. We also show that if integer flow solution is required, the problem becomes NP-complete in the strong sense. We further construct a network to show that the integrality gap, which is the maximum delay ratio of the optimal integer flow to the optimal fractional flow, could be arbitrarily large.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

[2] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: a low-delay multi-party conferencing solution," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 31, pp. 155–164, 2013.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010.

[4] F. Bai, T. Elbatt, G. Hollan, H. Krishnan, and V. Sadekar, "Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective," in *Proc. IEEE AutoNet*, 2006.

[5] B. H. Ashby, *Protecting Perishable Foods during Transport by Truck*. US Department of Agriculture, 2006.

[6] B. Nolan, "In storm, NO wants no one left behind," *The Times-Picayune*, 2005.

[7] L. Ford and D. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations Research*, vol. 6, no. 3, pp. 419–433, 1958.

[8] M. Lin and P. Jaillet, "On the quickest flow problem in dynamic networks: a parametric min-cost flow approach," in *Proc. SODA*, 2015.

[9] C. Dong, Q. Liu, and H. Zeng, "Traffic assignment with maximum delay constraint in stochastic network," in *Proc. IEEE VTC*, 2016.

[10] J. Correa, A. Schulz, and N. Moses, "Computational complexity, fairness, and the price of anarchy of the maximum latency problem," in *Proc. IPCO*, 2004.

[11] L. Ford and D. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[12] R. E. Burkard, K. Dlaska, and B. Klinz, "The quickest flow problem," *Zeitschrift für Operations Research*, vol. 37, no. 1, pp. 31–58, 1993.

[13] C.-C. Wang and M. Chen, "Sending perishable information: Coding improves delay-constrained throughput even for single unicast," in *Proc. IEEE ISIT*, 2014.

[14] J. Correa, A. Schulz, and N. Stier-Moses, "Fast, fair, and efficient flows in networks," *Operations Research*, vol. 55, no. 2, pp. 215–225, 2007.

[15] Y. Liu, D. Niu, and B. Li, "Delay-optimized video traffic routing in software-defined interdatacenter networks," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 865–878, 2016.

[16] H. C. Chin and A. D. May, *Examination of the speed-flow relationship at the Caldecott Tunnel*, 1991, no. 1320.

[17] M. R. Garey and D. S. Johnson, ""strong" NP-completeness results: motivation, examples, and implications," *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, 1978.

[18] Q. Liu, L. Deng, H. Zeng, and M. Chen, "On the min-max-delay problem: NP-completeness, algorithm, and int-gap," *Technical Report*, 2017. [Online]. Available: http://arxiv.org/abs/1707.02650

[19] A. R. Mahjoub and S. T. McCormick, "Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation," *Mathematical Programming*, vol. 124, no. 1-2, pp. 271–284, 2010.

[20] S. Sahni, "General techniques for combinatorial approximation," *Operations Research*, vol. 25, no. 6, pp. 920–936, 1977.