Collusion in Atomic Splittable Routing Games

Chien-Chung Huang[⋆]

Humboldt-Universität zu Berlin,
Unter den Linden 6, 10099, Berlin, Germany.
`villars@informatik.hu-berlin.de`

Abstract. We investigate how collusion affects the social cost in atomic splittable routing games. Suppose that players form coalitions and each coalition behaves as if it were a single player controlling all the flows of its participants. It may be tempting to conjecture that the social cost would be lower after collusion, since there would be more coordination among the players.

We construct examples to show that this conjecture is not true. Even in very simple single-source-single-destination networks, the social cost of the post-collusion equilibrium can be higher than that of the pre-collusion equilibrium. This counter-intuitive phenomenon of collusion prompts us to ask the question: *under what conditions would the social cost of the post-collusion equilibrium be bounded by the social cost of the pre-collusion equilibrium?*

We show that if (i) the network is "well-designed" (satisfying a natural condition), and (ii) the delay functions are affine, then collusion is always beneficial for the social cost in the Nash equilibria. On the other hand, if either of the above conditions is unsatisfied, collusion can worsen the social cost.

Our main technique is a novel flow-augmenting algorithm to build Nash equilibria. Our positive result for collusion is obtained by applying this algorithm simultaneously to two different flow value profiles of players and observing the difference in the derivatives of their social costs. Moreover, for a non-trivial subclass of selfish routing games, this algorithm finds the *exact* Nash equilibrium in polynomial time.

[⋆] Research supported by an Alexander von Humboldt fellowship.

1 Introduction

In an *atomic splittable routing game*, each player controls a non-negligible amount of flow and he can route his flow fractionally over the network. His strategy space consists of all possible ways of routing his flow. Each edge of the network is associated with a *delay function* of the flow value. A player routing his flow on an edge incurs a *cost*, which is the product of his flow value on that edge and the delay of that edge, determined by the total flow value on that edge. A player's total cost is the sum of his cost on all edges. Players are selfish. They aim for minimizing their own total costs while disregarding the cost of others. The *social cost* is the sum of all players' costs.

Atomic splittable routing games abstract real world situations such as communication networks and transportation, where a player can be an ISP or a freight/logistic company and he would try to minimize the delay experienced by his customers. A special case of this setting, mentioned as early as 1952 by Wardrop [32], is where each player controls an infinitesimal amount of flow. In the scenarios above, these players could be individual messages or drivers. A player controlling an infinitesimal amount of flow is conventionally called a *nonatomic* player. Furthermore, a Nash equilibrium in which all players are nonatomic is often called a *Wardrop equilibrium.*

We are interested in the social cost of a Nash equilibrium, a situation where no player can change his strategy unilaterally to decrease his cost. It can be expected that, given the selfishness of the players, the social cost of a Nash equilibrium would be sub-optimal. The degree of the worsening of the social cost in Nash equilibria is measured by the *price of anarchy* [23], and it has been intensively studied in recent years [12, 18, 25, 28, 30].

Life can be a bit more complex. Players form coalitions; companies merge or cooperate. So we are concerned about the social cost of the Nash equilibrium after the collusion. There can be different models to describe the colluding behavior among the players. The one we adopt is introduced by Hayrapetyan, Tardos, and Wexler [22]. In this model, once a coalition is formed, its participants care about their collective welfare. Thus this coalition would behave as if it were just a single player controlling all the flows of its participants.

It may be tempting to conjecture that the social cost will decrease after the collusion, since the colluding players have better coordination among themselves. In the extreme case where all players join in a single coalition, then the resultant equilibrium would become optimal. However, this conjecture has been shown to be false in several types of games [22]. Particularly for the atomic splittable routing games, if the network has multiple sources and destinations, it is known that the post-collusion equilibrium can have higher cost than the pre-collusion equilibrium [10, 12][1]

But maybe in the less chaotic single-source-single-destination networks, this counter-intuitive phenomenon of collusion may not occur? However, in this paper, we construct examples to show that even in very simple single-source-single-destination networks, collusion can worsen the social cost in the post-collusion equilibrium. These examples prompt us to investigate the following question:

> In an atomic splittable routing game, suppose that all players share a common source and destination. Under what conditions would the social cost of the post-collusion equilibrium be bounded by the social cost of the pre-collusion equilibrium?

Our Contribution

We first introduce a class of networks, which is a main focus of this work. Let the optimal flow be the flow that minimizes the social cost.

[1] More precisely, the examples in [10, 12] show that the social cost in the post-collusion equilibrium can be higher than in the pre-collusion *Wardrop* equilibrium. See the discussion in the related work for details.

Definition 1. *A single-source-single-destination network is* well-designed*, if as the value of the optimal flow is being increased, its flow value is monotonically non-decreasing on all edges. Precisely, let $O(t)$ denote an optimal flow of value t, indexed over all edges $e \in E$. A single-source-single-destination network is* well-designed*, if $t > t'$, then $O_e(t) \geq O_e(t')$ for all edges $e \in E$.*

A well-designed network thus conforms to the intuition that as the total flow becomes larger, we expect each edge to be more heavily used. In general, whether a network is well-designed depends on both the underlying graph topology and the delay functions. But there is an exception.

Proposition 2. *Suppose that the underlying graph of the network is series-parallel and all delay functions are convex. Then such a network is always well-designed, independent of the convex delay functions.*

The proof of this proposition is in the appendix. We now state our main result.

Theorem 3. *Suppose that the given network has a single source and a single destination and all delay functions are convex. If*

(i) the network is well-designed, and
(ii) all delay functions are affine,

then the social cost of the post-collusion equilibrium is bounded by that of the pre-collusion equilibrium.

On the other hand, if either of the two conditions is unsatisfied, then the social cost of the post-collusion equilibrium can be strictly larger than that of the pre-collusion equilibrium.

Our Technique Let $\sigma = (v_1, v_2, \cdots, v_k)$ be a *profile*, where $v_1 \geq v_2 \geq \cdots \geq v_k$ and each v_i represents the flow value of the i-th player. Note that if there are $k' < k$ players, we still can regard the game as one of k players with the last $k - k'$ players having zero flow values, i.e., the last $k - k'$ entries of σ are 0.

Definition 4. *Let $\sigma = (v_1, v_2, \cdots, v_k)$ and $\sigma' = (v'_1, v'_2, \cdots, v'_k)$ be two profiles and $\sum_{i=1}^{k} v_i = \sum_{i=1}^{k} v'_i = 1$. Then σ* majorizes *σ' if, for all $1 \leq i \leq k$, $\sum_{j=1}^{i} v_j \geq \sum_{j=1}^{i} v'_j$.*

We establish the first part of Theorem 3 via the following lemma.

Lemma 5. *Let σ and σ' be two profiles and the former majorizes the latter. In a well-designed network with affine delays, the social cost of the Nash equilibrium for σ is bounded by the social cost of the Nash equilibrium for σ'.*

Proof of the First Part of Theorem 3. Let σ' be the profile of the given game, and let σ be the profile after some players form coalitions. Observe that σ must majorize σ'. Hence Lemma 5 gives the proof. □

The main bulk of this paper (Section 3) is devoted to proving Lemma 5. Here we present the high-level idea.

We first characterize a well-designed network and show that the optimal flow is updated according to certain rate equations when its flow value is being increased in such a network. Exploiting these rate equations, we design a flow-augmenting algorithm to build a Nash equilibrium. Then we apply this algorithm to the input profiles σ and σ' simultaneously. We show that the derivative of the social cost for the flow based on σ is always no larger than that for the flow based on σ', thereby proving Lemma 5.

Counter-Examples Our positive result for collusion is tight: dropping either of the two conditions may cause collusion to become detrimental for the social cost. We carefully construct two counter-examples in which after some players form coalitions, the social cost in the post-collusion equilibrium is strictly higher than in the pre-collusion equilibrium. We briefly summarize the two examples (see Appendix A for details):

– *When the network has affine delays but is not well-designed*: in the Braess graph, which is known to be the smallest non-series-parallel graph, we build a not well-designed network with affine delays.
– *When the network is well-designed but the delay functions are not affine*: we use a network of just 3 parallel links (a special case of a series-parallel graph, hence well-designed). This network has polynomials as delay functions.

Computational Result Given Theorem 3, it would be of practical interest to test whether a network is well-designed. We have the following computational result.

Theorem 6. *Suppose that all delay functions are affine and the coefficients of the delays are rational. There is a polynomial time algorithm to test whether a network is well-designed. Moreover, if it is, we can find the* exact *Nash equilibrium and the* exact *Wardrop equilibrium in polynomial time.*

When all delay functions are affine, it is known that one can use convex programming to approximate the Nash equilibrium [12] or to approximate the Wardrop equilibrium [31]. To our knowledge, our algorithm is the first to find the exact equilibria for a nontrivial sub-class of atomic splittable routing games. See Appendix C for the proof of Theorem 6.

Related Work

Atomic splittable routing games are the least understood among various versions of selfish routing games. The exact price of anarchy in such games was only recently obtained by Roughgarden and Schoppmann [30]. For related results about other versions of the routing games, see [2, 7, 13, 18–20, 27, 29, 31, 32].

Hayrapetyan, Tardos, and Wexler [22] investigated the effect of collusion in various games by measuring the *price of collusion*, which is the worst ratio between the social cost of a post-collusion equilibrium against that of a pre-collusion equilibrium. Using their terminology, our results can be rephrased as identifying the conditions in atomic splittable routing games for the price of collusion to be bounded by 1. Fotakis, Kontogiannis, and Spirakis [16] investigated algorithmic questions about the Nash equilibrium after collusion in atomic congestion games.

When collusion is among non-atomic players A closely related scenario about collusion is that initially each player is nonatomic, i.e., he controls an infinitesimal amount of flow. The players may organize themselves into coalitions and each coalition cares about its collective welfare. Thus, each coalition would behave as if it were an atomic player controlling a splittable flow. In this scenario, the comparison between the social cost of the post-collusion equilibrium and the social cost of the pre-collusion *Wardrop* equilibrium is studied in [8, 12, 22]. In particular, these works investigate under what conditions would the social cost of the pre-collusion Wardrop equilibrium (with non-atomic players) be bounded by that of the post-collusion Nash equilibrium (with atomic players). Our work is the first to study the effect of collusion among the atomic players themselves.

In [22], it is shown that in a parallel-links graph with convex delays, the post-collusion Nash equilibrium has its social cost bounded by the social cost of the pre-collusion Wardrop equilibrium. The same result has been generalized to series-parallel graphs with convex delays in [8]. Our second counter-example offers an interesting contrast: even in a graph of just 3 links, if the collusion is among the atomic players themselves, collusion can worsen the social cost.

In the case that the network is well-designed with affine delays, [8] shows that the post-collusion Nash equilibrium has its social cost bounded by the social cost of the pre-collusion Wardrop equilibrium[2]. Our Lemma 5 can establish the same fact[3]. The result of [8] uses an earlier result of Cominetti, Correa, and Stier-Moses [12]. It proposes a flow re-distribution strategy to update the flow value profile, and, using linear algebraic arguments, shows that the social cost is monotonically non-decreasing. We consider the technique presented in this work more intuitive and simpler.

Collusion in single-source-single-destination networks Earlier works [10, 12] have shown that collusion may cause the social costs of equilibria to increase. We construct the first examples showing this is still true even if the network has a single source and a single destination. However, our examples are built assuming that the collusion is among atomic players themselves. Whether the post-collusion equilibrium can have higher social than the pre-collusion Wardrop equilibrium in such networks remains an open question.

Other models of collusion Another line of investigation about collusion is the *strong price of anarchy* [5] introduced by Andelman, Feldman, and Mansour, which is the worst ratio of the social cost in a *strong Nash equilibrium* [6] against that of the optimal solution. In a strong Nash equilibrium, no coalition of players can change their strategies so that every one of them improves his cost. Further results for the strong price of anarchy can be found in [3, 11, 14, 15, 21].

2 Preliminaries

Let $G = (V, E)$ be a directed graph, with two special vertices s and d called *source* and *destination* respectively. The vector f, indexed by edges $e \in E$, is defined as a *flow* of *value* v if the following conditions are satisfied.

$$\sum_{w:(u,w)\in E} f_{uw} - \sum_{w:(w,u)\in E} f_{wu} = 0, \quad \forall u \in V \backslash \{s, d\}. \tag{1}$$

$$\sum_{w:(s,w)\in E} f_{sw} - \sum_{w:(w,s)\in E} f_{ws} = v. \tag{2}$$

$$f_e \geq 0, \quad \forall e \in E. \tag{3}$$

A flow is a *circulation* if its value is 0. If f satisfies only conditions (1) and (2), f is a *pseudo flow* of value v. If there are several flows $\{f^1, f^2, \cdots, f^k\}$, the total flow f is defined as $f := \sum_{i=1}^k f^i$. We define $f^{-i} := \sum_{j\neq i} f^j = f - f^i$.

Each edge e is associated with a delay function $l_e : \mathcal{R}^+ \rightarrow \mathcal{R}^+$. A delay function is *affine*, if $l_e(x) = a_e x + b_e$, where $a_e > 0$ and $b_e \geq 0$. For a flow f, player i incurs a cost $C(f^i, f^{-i}) = \sum_{e\in E} f^i_e l_e(f_e)$

[2] Even though the result for affine delays as stated in [8] is only for series-parallel networks, the technique of [8] indeed also works for well-designed networks. This is because a well-designed network always satisfies the *nesting property* (see Section 3.2 for definition). A minor contribution of this paper is to identify a larger class of networks for this result of [8] to hold.

[3] The proof proceeds as follows. Let σ' be the profile of k players, each with the same amount of flow. Let σ be the profile of the post-collusion Nash equilibrium. Observe that σ majorizes σ'. It is established in [12] that the social cost of the Nash equilibrium for σ' has social cost no larger than the pre-collusion Wardrop equilibrium. Now applying Lemma 5 to compare social costs for the Nash equilibria based on σ' and σ would give the proof.

The social cost of a flow f is defined as $C(f) = \sum_{e \in E} f_e l_e(f_e)$, which is also the sum of the costs of all players $\sum_i C(f^i, f^{-i})$. A flow $O(t)$ of value t is optimal if it minimizes the social cost among all flows f of value t.

An *atomic splittable routing game* is a tuple $(\sigma, (G, \mathbf{l}, s, d))$ where $\sigma = (v_1, \cdots, v_k)$ is a profile indicating the flow values of the players from 1 to k, ordered non-increasingly, and $(G, \mathbf{l}, s, d)$ a network and $\mathbf{l}$ its vector of delay functions for edges in G. A s-d path is a directed simple path from s to d. $\mathcal{P}$ denotes the set of all s-d paths. We often abuse notation by writing $e \in \mathcal{P}' \subseteq \mathcal{P}$ if e belongs to any path $p \in \mathcal{P}'$. An edge e is *used* if $f_e > 0$ and is *used by player i* if $f_e^i > 0$. Similarly, a path p is used (by player i) if on every edge $e \in p$, $f_e > 0$ ($f_e^i > 0$).

Each player i has a strategy space consisting of all possible s-d flows of value v_i. His objective is to minimize his cost $C(f^i, f^{-i})$. A set of players are said to be *symmetric* if each of them has the same flow value.

A flow is a Nash equilibrium if no player can unilaterally alter his flow to reduce his cost.

Definition 7 (Nash Equilibrium). *In an atomic splittable routing game $(\sigma, (G, \mathbf{l}, s, d))$, flow f is a Nash equilibrium if and only if, for every player i and every s-d flow g of value v_i, $C^i(f^i, f^{-i}) \leq C^i(g, f^{-i})$.*

For a player i and a path p, his *marginal cost on path p* is the rate of change of his cost when he adds flow along path p: $\sum_{e \in p} l_e(f_e) + f_e^i l_e'(f_e)$. The following lemma follows from Karush-Kuhn-Tucker optimality conditions for convex programs [24] applied to player i's minimization problem.

Lemma 8. *Flow f is a Nash equilibrium if and only if for any player i and any two directed paths p and q between the same pair of vertices and player i uses path p,*

$$\sum_{e \in p} l_e(f_e) + f_e^i l_e'(f_e) \leq \sum_{e \in q} l_e(f_e) + f_e^i l_e'(f_e).$$

Note that the Nash equilibrium of just one player is exactly the optimal flow.

Lemma 9 (Existence and Uniqueness of Nash Equilibrium). [4, 25, 26] *In an atomic splittable routing game $(\sigma, (G, \mathbf{l}, s, d))$, if all functions in $\mathbf{l}$ are affine, or if the underlying graph of G is parallel-links and all functions in $\mathbf{l}$ are convex, then there exists a Nash equilibrium and it is unique.*

When All Delay Functions are Affine. We introduce the notion of *ϕ-delay*, which plays a prominent role in our algorithms and analysis. Let the *ϕ-delay* of an edge e be $L_e(f, \phi) = \phi a_e f_e + b_e$ and along a path p be $L_p(f, \phi) = \sum_{e \in p} L_e(f, \phi)$.

Definition 10. *A flow f is ϕ-*optimal *for some $\phi > 0$ if and only if, given any two directed paths p and q between the same pair of vertices and f uses path p,*

$$L_p(f, \phi) \leq L_q(f, \phi).$$

By Definition 10, a 2-optimal flow is exactly the optimal flow; 1-optimal flow is a Wardrop equilibrium[4]; and a Nash equilibrium of k symmetric players is $\frac{k+1}{k}$-optimal[5].

[4] A Wardrop equilibrium f has the following characterization [27, 32]: if p and q are directed paths between the same pair of vertices and f uses p, $\sum_{e \in p} l_e(f_e) \leq \sum_{e \in q} l_e(f_e)$.

[5] This follows from the fact that in a Nash equilibrium of symmetric players, the flows of all players are identical [12].

3 Proof of Lemma 5

In this section, we assume that all delay functions are affine. We prove Lemma 5 through the following three steps: (1) we characterize well-designed networks (Section 3.1); (2) using this characterization, we design an algorithm to construct a Nash equilibrium based on the given profile (Section 3.2). This algorithm proceeds by increasing flow value gradually and maintains a ϕ-optimal flow, with a dynamically changing ϕ; (3) we apply this algorithm to two different profiles and observe their relative growing speeds of the social cost while the flow values are being increased (Section 3.3).

3.1 Characterizing Well-Designed Networks

A well-designed network is associated with several sets of items, whose exact properties will be captured by Lemma 12 below. Here we summarize them. A well-designed network $(G, \mathbf{l}, s, d)$ has

- a sequence of nested sets of paths $\mathcal{P}_0 = \emptyset \subset \mathcal{P}_1 \subset \mathcal{P}_2 \subset \cdots \mathcal{P}_x \subseteq \mathcal{P}$, which are the sets of s-d paths that the optimal flow uses when its flow value is increased;
- a sequence of flow values $t_0 = 0 < t_1 < \cdots < t_{x-1} < t_x = \infty$, which are the values by which the optimal flow begins to use a different set of paths[6];
- a sequence of vectors $\alpha^1, \alpha^2, \cdots, \alpha^x$, which are vectors indexed over the edges in E. Each of these vectors indicates how the optimal flow updates itself when its flow value is increased;
- a sequence of *ϕ-delay* thresholds $\Psi_0 = \min_{q \in \mathcal{P}} \sum_{e \in q} b_e < \Psi_1 < \Psi_2 < \cdots < \Psi_{x-1}$, each of which indicates the 2-delay of a path in $\mathcal{P}_1$ when the flow value of the optimal flow is t_i.

Before explaining the details about these items, we need a fact from linear algebra. Assume that $\mathcal{P}_i \subseteq \mathcal{P}$ is a subset of s-d paths used by a flow f of value t. Consider the following linear system.

$$\sum_{e \in p} a_e f_e - \sum_{e \in q} a_e f_e = \frac{1}{2}(\sum_{e \in q} b_e - \sum_{e \in p} b_e) \quad \forall p, q \in \mathcal{P}_i \tag{4}$$

$$\sum_{u:(s,u) \in E} f_{su} = t \tag{5}$$

$$\sum_{v:(u,v) \in E} f_{uv} - \sum_{v:(v,u) \in E} f_{vu} = 0 \qquad \forall u \notin \{s, d\} \tag{6}$$

$$f_e = 0 \qquad \forall e \notin \mathcal{P}_i \tag{7}$$

Proposition 11. *If the system (4)-(7) has a unique solution, then the flow value on each edge $e \in E$ can be expressed as $f_e = \alpha_e^i t + \beta_e^i$, where α_e^i and β_e^i depend on $\mathcal{P}_i$ and $\{a_e, b_e\}_{e \in \mathcal{P}_i}$.*

Proof. This follows from Gaussian elimination. □

Lemma 12 (Characterization of a Well-designed Network). *Suppose that $(G, \mathbf{l}, s, d)$ is well-designed.*

(i) There exists a sequence of nested sets of paths $\mathcal{P}_0 = \emptyset \subset \mathcal{P}_1 \subset \mathcal{P}_2 \subset \cdots \subset \mathcal{P}_x \subseteq \mathcal{P}$, and a set of values $t_0 = 0 < t_1 < \cdots < t_{x-1} < t_x = \infty$ so that when $t \in (t_{i-1}, t_i]$, $\forall 1 \leq i \leq x$, $O(t)$ uses the set of edges in $\mathcal{P}_i$.

(ii) There exists a sequence of vectors α^i, $1 \leq i \leq x$, which has the following properties:

(iia) if $t_{i-1} \leq t < t' \leq t_i$, then $O(t') - O(t) = \alpha^i (t' - t)$;

[6] To simplify our presentation, we slightly abuse notation by writing a value that goes to infinity as a fixed value t_x.

(iib) α^i*, by itself, is also a flow of value 1; specifically,* $\alpha_e^i \geq 0$, $\forall i, e$*; and* $\alpha_e^i = 0$ *if* $e \notin \mathcal{P}_i$*;*

(iic) For every two paths $p, q \in \mathcal{P}_i$, $\sum_{e \in p} a_e \alpha_e^i = \sum_{e \in q} a_e \alpha_e^i$.

(iii) There exists a sequence of ϕ-delay thresholds $\Psi_0 = \min_{q \in \mathcal{P}} \sum_{e \in q} b_e < \Psi_1 < \Psi_2 < \cdots < \Psi_{x-1}$ *so that when* $t = t_i$, $0 \leq i \leq x-1$*, all paths* $q \in \mathcal{P}_{i+1}$ *(including those in* $\mathcal{P}_{i+1} \backslash \mathcal{P}_i$*) have the same minimum 2-delay* $\Psi_i = L_p(O(t_i), 2)$*. Furthermore,*

(iiia) $\Psi_i = \Psi_{i-1} + 2(t_i - t_{i-1}) \sum_{e \in p} a_e \alpha_e^i$*, given a path* $p \in \mathcal{P}_1$*, for* $1 \leq i \leq x-1$*.*

Proof. (i) follows from the definition of well-designedness: if an edge is used by the optimal flow, then it continues to be used by the optimal flow when the latter increases its flow value.

For (ii), first assume that $t_{i-1} < t$ (note the strict inequality). Then by (i), both $O(t)$ and $O(t')$ use the same set of paths $\mathcal{P}_i$. Recall that the optimal flow is also a Nash equilibrium. Thus, by the characterization of Nash equilibrium (Lemma 8), and the existence and uniqueness of Nash equilibrium (Lemma 9), we can apply Proposition 11 to get

$$O_e(t') - O_e(t) = \alpha_e^i (t' - t) + \beta_e^i - \beta_e^i = \alpha_e^i (t' - t), \quad \forall e \in E, \text{ if } t_{i-1} < t < t' \leq t_i. \tag{8}$$

This proves (iia) except for the case that $t = t_{i-1}$.

For (iib), note that by (8), we have $\alpha^i = (O(t') - O(t))/(t' - t)$, and $O(t') - O(t)$ is a pseudo flow of value $t' - t$, and since the network is well-designed, $O_e(t') - O_e(t) \geq 0$ for all edges $e \in E$. Hence all components of α^i are non-negative, implying that $O(t') - O(t)$ is a flow of value $t' - t$ and α^i is a flow of value 1. Finally, $\alpha_e^i = 0$ if $e \notin \mathcal{P}_i$, since α^i is a solution to the system of (4)-(7).

For (iic), since $O(t)$ and $O(t')$ are solutions to the system (4)-(7), we have

$$\frac{1}{2}\left(\sum_{e \in q} b_e - \sum_{e \in p} b_e\right) = \sum_{e \in p} a_e O_e(t) - \sum_{e \in q} a_e O_e(t) = \sum_{e \in p} a_e O_e(t') - \sum_{e \in q} a_e O_e(t'),$$

which then implies that

$$\sum_{e \in p} a_e (O_e(t') - O_e(t)) = \sum_{e \in q} a_e (O_e(t') - O_e(t)).$$

By (8), the LHS equals $\sum_{e \in p} a_e \alpha_e^i (t' - t)$ and the RHS $\sum_{e \in q} a_e \alpha_e^i (t' - t)$. (iic) follows.

To finish the proof of (iia), we need to handle the case that $t = t_{i-1}$. Define $\tilde{O}(t^*) := O(t') - \alpha^i (t' - t^*)$. We claim that $\tilde{O}(t^*) = O(t^*)$ if $t^* = t_{i-1}$, and the proof would follow if the claim holds. Suppose, for a contradiction, that $\tilde{O}(t_{i-1}) \neq O(t_{i-1})$. There can be two reasons for $\tilde{O}(t_{i-1})$ not being optimal: either $\tilde{O}_e(t_{i-1}) < 0$ on some edge $e \in E$, or there exists a path $p \in \mathcal{P}$ used by $\tilde{O}_e(t_{i-1})$ so that p's 2-delay $L_p(\tilde{O}(t_{i-1}), 2)$ is not minimum among all paths $q \in \mathcal{P}$. In both cases, choose an arbitrary small $\epsilon > 0$ so that $t_{i+1} + \epsilon < t'$. Then $\tilde{O}(t_{i-1} + \epsilon)$ is still not an optimal flow, contradicting (8).

For (iii), first consider the case that $t = t_i$, $i \geq 1$. By Lemma 8, all paths $p \in \mathcal{P}_i$ have the same minimum 2-delay $L_p(O(t_i), 2)$ among all paths in $\mathcal{P}$. Let this 2-delay be Ψ_i. To see that $q \in \mathcal{P}_{i+1} \backslash \mathcal{P}_i$ also has the same 2-delay, observe that by (iia) and (iic), the 2-delay of all paths $p \in \mathcal{P}_{i+1}$ increases in the same speed, i.e., $L_p(O(t_i + \epsilon), 2) - L_p(O(t_i), 2) = 2\epsilon \sum_{e \in p} a_e \alpha_e^i$, which is constant for all paths $p \in \mathcal{P}_{i+1}$. If there is some path $q \in \mathcal{P}_{i+1} \backslash \mathcal{P}_i$ such that $L_q(O(t_i), 2) > \Psi_i$. Then as t increases from t_i to $t_i + \epsilon \leq t_{i+1}$, $L_q(O(t_i + \epsilon), 2) > L_{p \in \mathcal{P}_i}(O(t_i + \epsilon), 2)$, a contradiction to Lemma 8.

When $t = t_0 = 0$, by the same argument as above, all paths in $\mathcal{P}_1$ increase their 2-delays in the same speed when t increases. Hence in $O(0)$, all paths $p \in \mathcal{P}_1$ should have the same minimum 2-delay among all paths in $\mathcal{P}$, which is $\sum_{e \in p} b_e$, and we define this value to be Ψ_0. This completes the proof of (iii).

Finally, (iiia) holds because, by (iia) and (iic), from t_{i-1} to t_i, the path $p \in \mathcal{P}_1$ increases its 2-delay by the amount of $2(t_i - t_{i-1}) \sum_{e \in p} a_e \alpha_e^i$. □

In the following, we assume that the nested sets of paths $\mathcal{P}_i$, the vectors α^i, and the ϕ-delay thresholds Ψ_i are known. See Appendix C about finding them in polynomial time if all coefficients $\{a_e, b_e\}_{e \in E}$ are rational.

The Vectors $\boldsymbol{\alpha^i}$ as "Accelerators" and $\boldsymbol{\phi}$-Delay Thresholds $\boldsymbol{\Psi_i}$ as "Landmarks" The vectors α^i and the ϕ-delay thresholds Ψ_i play a pivotal role in our algorithm in Section 3.2. Here we explain why they are useful.

Lemma 12(ii) suggests an algorithmic view on an optimal flow when its value is increased: when its value is increased from t to $t + \epsilon$ so that $t_{i-1} \le t \le t + \epsilon \le t_i$, the optimal flow is *increased in the speed of* α^i in the following sense

$$O(t + \epsilon) := O(t) + \alpha^i \epsilon.$$

In other words, the vector α^i serves as an accelerator to tell the optimal flow how it should update itself when the flow value increases. Lemma 12(iii) implies that once the flow value reaches t_i, the 2-delay of all paths in $\mathcal{P}_{i+1}$ will become exactly Ψ_i. And after this point, the optimal flow begins to use the paths in $\mathcal{P}_{i+1}$, instead of $\mathcal{P}_i$. This suggests that we can view these thresholds Ψ_i as a sort of "landmarks": pick any path $p \in \mathcal{P}_1$ (and note that then $p \in \mathcal{P}_i$ for any i). Once its 2-delay becomes Ψ_i, it is time to change gear: the optimal flow thence increases in the speed of α^{i+1}, instead of α^i.

Our main algorithm in Section 3.2 is inspired by these observations. Initially, the flow value is 0 and we increase it gradually up to 1. In this process, the flow is updated based on these accelerators α^i, and each time the ϕ-delay of a path in $\mathcal{P}_1$ reaches a landmark Ψ_i, we use a different accelerator α^{i+1} to update the flow.

To better illustrate our idea, we first present a simpler algorithm, which we call Algorithm A, in Figure 1. It constructs a Nash equilibrium for k symmetric players. Note that when $k \to \infty$, the Nash equilibrium would just be a Wardrop equilibrium.

Algorithm A maintains a $\frac{k+1}{k}$-optimal flow $f(t)$ of value t when t is increased from 0 to 1. The index $\mathtt{h}$ records the set of paths $\mathcal{P}_{\mathtt{h}}$ that $f(t)$ uses. $f(t)$ is increased in the speed of $\alpha^{\mathtt{h}}$ (see Lines 5-6). Each time the $\frac{k+1}{k}$-delay of a path $p \in \mathcal{P}_1$ reaches Ψ_h, $f(t)$ is then increased in the speed of α^{h+1} (see Lines 2 and 4). Line 3 just records the value $t_h(k)$ by which $f(t)$ shifts from using $\mathcal{P}_h$ to $\mathcal{P}_{h+1}$.[7] As a clarification, in the following discussion, $f(t^*)$ refers to the current flow maintained by Algorithm A between Lines 1 and 2 when $t = t^*$.

Lemma 13. *Let $t_0(k) = 0$ and $t_z(k) = 1$. (So when Algorithm A terminates $\mathtt{h} = z$).*

(i) For $0 \le i \le z$, $f(t_i(k))$ is an equilibrium flow with total flow value $t_i(k)$ for k symmetric players, and it uses the same set of paths $\mathcal{P}_i$ as $O(t_i)$.
(ii) Let q be a path in $\mathcal{P}$ and $p \in \mathcal{P}_1$ the path chosen in Line 0.
(iia) $0 \le i \le z-1$, $L_q(f(t_i(k)), \frac{k+1}{k}) = L_q(O(t_i), 2)$ and $L_p(f(t_i(k)), \frac{k+1}{k}) = L_p(O(t_i), 2) = \Psi_i$.
(iib) $L_q(f(t_z(k)), \frac{k+1}{k}) \le L_q(O(t_z), 2)$ and $L_p(f(t_z(k)), \frac{k+1}{k}) \le L_p(O(t_z), 2) = \Psi_z$.

[7] Since we increase the flow by an infinitesimal amount ϵ, this algorithm, along with the main algorithm in the next section, does not run in polynomial time. But they can be easily modified to run in polynomial time. We choose to present in this manner since we consider it as simpler and it makes the analysis in Section 3.3 go smoother.

```
   Input: $k$ // the number of symmetric players;
   Initialization: $t := 0$; $f_e(t) := 0, \forall e \in E$; $\mathtt{h} = 1$; $\epsilon > 0$ is an infinitesimal amount;
         // $f(t)$ is the current flow of value $t$; $\mathtt{h}$ is the index of the set of paths $\mathcal{P}_{\mathtt{h}}$ that $f(t)$ uses;
0.  pick $p \in \mathcal{P}_1$;
1.  While $t < 1$
2.     if $L_p(f(t), \frac{k+1}{k}) = \Psi_h$ then
3.        $t_{\mathtt{h}}(k) := t$;
4.        $\mathtt{h} := \mathtt{h} + 1$;
5.     $f(t + \epsilon) := f(t) + \alpha^h \epsilon$;
6.     $t := t + \epsilon$;
7.  End
8.  $t_{\mathtt{h}}(k) := 1$;
```

Fig. 1. Algorithm A.

Proof. We first note that $f(t)$ is a flow for all $t \in [0, 1]$. This is because by Lemma 12(iib), $\alpha^i \epsilon$ is a flow of value ϵ, and $f(t)$ is the sum of many flows of value ϵ, which is then still a flow.

We now prove the lemma by induction on i. In the base case $i = 0$, (i) is trivial and (ii) holds because $L_q(f(t_0(k) = 0), \frac{k+1}{k}) = \sum_{e \in q} b_e = L_q(O(t_0 = 0), 2)$, and this value equals Ψ_0 if $q = p$ according to Lemma 12(iii). For the induction step, observe that by the induction hypothesis,

$$L_q(f(t_{i-1}(k)), \frac{k+1}{k}) = L_q(O(t_{i-1}), 2) \quad \text{and this quantity equals } \Psi_{i-1} \text{ if } q = p \in \mathcal{P}_1. \tag{9}$$

Let $\tau := \frac{k+1}{2k}(t_i(k) - t_{i-1}(k))$. Then the quantity $2\tau \sum_{e \in q} a_e \alpha_e^i$ represents the increase of path q's $\frac{k+1}{k}$-delay from $t_{i-1}(k)$ to $t_i(k)$. We claim that

$$t_{i-1} + \tau \le t_i \quad \text{and the inequality holds with equality if } 1 \le i \le z - 1. \tag{10}$$

If the claim holds, then when $t \in (t_{i-1}, t_{i-1} + \tau]$, the optimal flow sticks to using $\mathcal{P}_i$. This claim follows from

$$t_i - t_{i-1} = \frac{\Psi_i - \Psi_{i-1}}{2\sum_{e \in p} a_e \alpha_e^i} \ge \frac{L_p(f(t_i(k)), \frac{k+1}{k}) - L_p(f(t_{i-1}(k)), \frac{k+1}{k})}{2\sum_{e \in p} a_e \alpha_e^i} = \frac{2\tau \sum_{e \in p} a_e \alpha_e^i}{2\sum_{e \in p} a_e \alpha_e^i} = \tau,$$

where the first equality follows from Lemma 12(iiia) and the inequality from (9) and the fact that p's $\frac{k+1}{k}$-delay at $t_i(k)$ is no larger than Ψ_i. And this inequality holds with equality if $1 \le i \le z - 1$. Now,

$$L_q(f(t_i(k)), \frac{k+1}{k}) = L_q(f(t_{i-1}(k)), \frac{k+1}{k}) + 2\tau \sum_{e \in q} a_e \alpha_e^i$$
$$= L_q(O(t_{i-1}), 2) + \tau(2 \sum_{e \in q} a_e \alpha_e^i) = L_q(O(t_{i-1} + \tau), 2) \le L_q(O(t_i), 2),$$

where the second equality follows from (9) and the third from (10). So we establish (ii). (The second part of (iia) holds because $t_{i-1} + \tau = t_i$ if $1 \le i \le z - 1$ and by Lemma 12(iii), $L_p(O(t_i), 2) = \Psi_i$.)

By the algorithm, $f(t_i(k)) = f(t_{i-1}(k)) + \alpha^i(t_i(k) - t_{i-1}(k))$ and by (i) of the induction hypothesis, $f_{i-1}(k)$ uses $\mathcal{P}_{i-1}$. Thus $f(t_i(k))$ uses $\mathcal{P}_i$ as $O(t_i)$. Now $f(t_i(k))$ and $O(t_i + \tau)$ use the same set of paths $\mathcal{P}_i$ and the $\frac{k+1}{k}$-delay of the former is the same as the 2-delay of the latter along all paths. Since the latter is 2-optimal, the former must be $\frac{k+1}{k}$-optimal. This proves (i). □

We remark that it is not difficult to infer that with the same flow value t, the smaller the k, the larger set of the paths $\mathcal{P}_i$ that the $\frac{k+1}{k}$-optimal flow $f(t)$ uses. (Since a larger k implies a slower growth-rate of the ϕ-delay.) The extreme case is the optimal flow with $k = 1$ player. Thus, the optimal flow shifts to larger sets of paths the fastest. This implies that *the growing speed of the social cost using a larger set of paths is less than the growing speed of the social cost using a smaller set of paths* (under a certain condition). Our analysis in Section 3.3 is centered around this intuition.

3.2 Constructing Nash Equilibria in a Well-Designed Network

The nice thing about a well-designed network, as we will show, is that it guarantees its Nash equilibria satisfy the *nesting property.*

Definition 14 (Nesting Property). *A flow f satisfies the* nesting property*, if when players i and j have flow values $v_i > v_j$, then $f_e^i > f_e^j$ on any edge $e \in E$ used by player j, and when $v_i = v_j$, then $f^i = f^j$.*

If a Nash equilibrium satisfies the nesting property, it has an alternative characterization, which will be exploited by our main algorithm. Let

$$b_e^r := a_e\left(\sum_{i=r+1}^{k} f_e^i\right) + b_e.$$

Lemma 15. *Suppose that flow f satisfies the nesting property. If given any two directed paths p and q between the same pair of vertices and p is used by player r, and the following holds:*

$$\sum_{e\in p} a_e f_e^r + \frac{1}{r+1} b_e^r \leq \sum_{e\in q} a_e f_e^r + \frac{1}{r+1} b_e^r, \tag{11}$$

then the flow f is a Nash equilibrium.

Proof. To prove that a flow f fulfilling the condition in the lemma is a Nash equilibrium, by Lemma 8, we need to show that for any player r, if he uses path p, then

$$\sum_{e\in p} a_e\left(\sum_{i=1,i\neq r}^{k} f_e^i + 2f_e^r\right) + b_e \leq \sum_{e\in q} a_e\left(\sum_{i=1,i\neq r}^{k} f_e^i + 2f_e^r\right) + b_e, \tag{12}$$

for any directed path q connecting the same pair of vertices as p.

Now suppose that player i uses path p and that q is any other directed path between the same pair of vertices as p. Define

$$X_i := \sum_{e\in p} a_e f_e^i + \frac{b_e^i}{i+1}, Y_i := \sum_{e\in q} a_e f_e^i + \frac{b_e^i}{i+1}.$$

Since we assume that the flow satisfies the nesting property, all the first r players use path p. Therefore, $X_i \leq Y_i$ for all $1 \leq i \leq r$. Hence the expression in (12) can be written as

$$X_1 + \frac{1}{2}X_2 + \frac{1}{3}X_3 + \cdots + \frac{1}{r-1}X_{r-1} + \frac{r+1}{r}X_r \leq Y_1 + \frac{1}{2}Y_2 + \frac{1}{3}y_3 + \cdots + \frac{1}{r-1}Y_{r-1} + \frac{r+1}{r}Y_r.$$

□

High-Level Ideas of Algorithm *B* Let $\sigma = (v_1, \cdots, v_k)$ be the input profile. Our algorithm maintains a ϕ-optimal flow when the total flow value is increased from 0 to 1. But unlike the previous algorithm, ϕ is dynamically changing. We maintain two indices $\mathtt{h}$ and $\mathtt{r}$, where $\mathtt{h}$ indicates the set of paths $\mathcal{P}_{\mathtt{h}}$ the current flow uses and $\mathtt{r}$ the index of the player whose flow is about to be created (we will explain how). Initially, let $\mathtt{h} := 1$ and $\mathtt{r} := k$.

While maintaining the invariant the current flow is $\frac{\mathtt{r}+1}{\mathtt{r}}$-optimal, our algorithm increases the flow in the speed of $\alpha^{\mathtt{h}}$. Two things may happen in this process.

– The $\frac{\mathtt{r}+1}{\mathtt{r}}$-delay of a path $p \in \mathcal{P}_1$ attains $\Psi_{\mathtt{h}}$. Then we increase the flow thence in the speed of $\alpha^{\mathtt{h}+1}$ (so we increment the index $\mathtt{h}$).
– The current flow value attains $\mathtt{r}v_{\mathtt{r}}$. Then we "freeze" $\frac{1}{\mathtt{r}}$ fraction of the current flow, give it to the $\mathtt{r}$-th player, treat the remaining unfrozen $\frac{\mathtt{r}-1}{\mathtt{r}}$ fraction as the current flow, and "update" the network (to be explained shortly).
  We will prove that the unfrozen $\frac{\mathtt{r}-1}{\mathtt{r}}$ fraction of the flow is $\frac{\mathtt{r}}{\mathtt{r}-1}$-optimal in the *updated* network. This is a crucial point of our algorithm. After the freezing, we decrement the index $\mathtt{r}$.

In the end, we can show that the k frozen flows that are assigned to the players satisfy the nesting property and the condition in (11). Thus by Lemma 15 they constitute a Nash equilibrium in the *original* network.

We now explain how the "freezing" is done and how the network is updated. Let $f(\sigma, t)$ denote the *current flow* and we will maintain the invariant that its flow value is $t - \sum_{j=\mathtt{r}+1}^{k} v_j$. Assume that in the current network the delay function of each edge e is $l_e(x) = a_e x + b_e^{\mathtt{r}}$ and the flow value of $f(\sigma, t)$ is $\mathtt{r}v_{\mathtt{r}}$. We freeze a fraction of the current flow as follows:

$$f^{\mathtt{r}} := \frac{1}{\mathtt{r}} f(\sigma, t), f(\sigma, t) := \frac{\mathtt{r}-1}{\mathtt{r}} f(\sigma, t).$$

Thus $f(\sigma, t)$ is split into two parts: the frozen part, $f^{\mathtt{r}}$, which is given to the $\mathtt{r}$-th player; the unfrozen part $\frac{\mathtt{r}-1}{\mathtt{r}} f(\sigma, t)$, which is now treated as the current flow of value $(\mathtt{r}-1)v_{\mathtt{r}}$.

We also update the network in the process of freezing as follows.

$$l_e(x) := a_e x + (b^{\mathtt{r}} + a_e f_e^{\mathtt{r}}) = a_e x + b_e^{\mathtt{r}-1}, \forall e \in E.$$

In words, the newly frozen flow $f^{\mathtt{r}}$ adds a constant term to the delay function on each edge e. Finally, we decrement the index $\mathtt{r}$.

Now let $L_{e,\mathtt{r}}(f(\sigma, t), \frac{\mathtt{r}+1}{\mathtt{r}})$ denote the $\frac{\mathtt{r}+1}{\mathtt{r}}$-delay of edge e in the current network (which has been updated $k - \mathtt{r}$ times because of those frozen flows), i.e.,

$$L_{e,\mathtt{r}}(f(\sigma, t), \frac{\mathtt{r}+1}{\mathtt{r}}) = \frac{\mathtt{r}+1}{\mathtt{r}} a_e f_e(\sigma, t) + b_e^{\mathtt{r}},$$

and in the following we refer to $L_{e,\mathtt{r}}(f(\sigma, t), \frac{\mathtt{r}+1}{\mathtt{r}})$ as the ϕ-delay of the current flow on edge e.

We make a crucial observation about the consequence of the freezing: *The ϕ-delay of the current flow on each edge remains unchanged after the freezing.* To be precise, assume that after the freezing, the decremented index $\mathtt{r} = r - 1$. Then *the $\frac{r}{r-1}$-delay of each edge (path) in the new network with the remaining unfrozen flow $\frac{r-1}{r} f(\sigma, t)$ would be identical to its $\frac{r+1}{r}$-delay in the previous network with its entire flow $f(\sigma, t)$.*

Claim. For each edge $e \in E$, $L_{e,r-1}(\frac{r-1}{r} f(\sigma, t), \frac{r}{r-1}) = L_{e,r}(f(\sigma, t), \frac{r+1}{r})$.

Proof. This follows from

$$L_{e,r-1}(\frac{r-1}{r}f(\sigma,t),\frac{r}{r-1}) = \frac{r}{r-1}\left[a_e\frac{r-1}{r}f_e(\sigma,t)\right]+(a_e f_e^r+b_e^r) = \frac{r+1}{r}a_e f_e(\sigma,t)+b_e^r = L_{e,r}(f(\sigma,t),\frac{r+1}{r}).$$

□

By this claim, if we can show that immediately before the freezing, the current flow is $\frac{r+1}{r}$-optimal in the previous network, then the unfrozen remaining flow will be $\frac{r}{r-1}$-optimal in the updated network.

We now present our main algorithm, Algorithm B, in Figure 2. We record the values $t_h(\sigma)$ and $t^r(\sigma)$ (see Lines 4 and 7) as the total flow values by which the current flow shifts to a larger set of paths and by which the flow of the r-th player is created. The inner while loop is necessary because it is possible the flow values of multiple players are the same, or at some point, the indice $\mathtt{h}$ and $\mathtt{r}$ are updated at the same time. As a clarification, in the following lemma, its proof, and the subsequent discussion in Section 3.3, we assume that Lines 2-14 of Algorithm B are executed *instantaneously.* When we say "at time t^*", we refer to the moment Algorithm B is executing between Lines 1 and 2 when $t = t^*$. Similarly, $f(t^*,\sigma)$ refers to the current flow maintained by Algorithm B between Lines 1 and 2 when $t = t^*$. These assumptions are made to avoid possible confusion due to the changes in the indices between Lines 2-14. It is easy to verify the invariant that the flow value of $f(\sigma,t)$ being $t - \sum_{j=\mathtt{r}+1}^{k} v_j$ is maintained. The next lemma shows that the outcome will be a Nash equilibrium.

Input: A profile $\sigma = (v_1,\cdots,v_k)$, where $\sum_{j=1}^{k} v_j = 1$;
Initialization: $t := 0$; $\mathtt{r} := k$; $\mathtt{h} := 1$; $f_e(\sigma,t) := 0, \forall e \in E$; $\epsilon > 0$ an infinitesimal amount;
// $f(\sigma,t)$ is the current flow, $\mathtt{h}$ is the index of the set of paths $\mathcal{P}_{\mathtt{h}}$ that $f(\sigma,t)$ uses;
// $\mathtt{r}$ is the index of the player whose flow is about to be created;

```
0.  pick $p \in \mathcal{P}_1$;
1.  While $t \le 1$ and $\mathtt{r} \ge 1$
2.      While $L_{p,\mathtt{r}}(f(\sigma,t),\frac{\mathtt{r}+1}{\mathtt{r}}) = \Psi_{\mathtt{h}}$ or $t - \sum_{j=\mathtt{r}+1}^{k} v_j = \mathtt{r}v_{\mathtt{r}}$
3.        if $L_{p,\mathtt{r}}(f(\sigma,t),\frac{\mathtt{r}+1}{\mathtt{r}}) = \Psi_{\mathtt{h}}$ then      // the $\phi$-delay of path $p \in \mathcal{P}_1$ reaches a $\phi$-delay threshold
4.            $t_{\mathtt{h}}(\sigma) := t$;
5.            $\mathtt{h} := \mathtt{h}+1$;
6.        else if $t - \sum_{j=\mathtt{r}+1}^{k} v_j = \mathtt{r}v_{\mathtt{r}}$ then      // the current flow value reaches $\mathtt{r}v_{\mathtt{r}}$
7            $t^{\mathtt{r}}(\sigma) := t$;
8.            $f^{\mathtt{r}} := \frac{1}{\mathtt{r}}f(\sigma,t)$;
9.            $f(\sigma,t) := \frac{\mathtt{r}-1}{\mathtt{r}}f(\sigma,t)$;
10.           $\forall e \in E$, $l_e(x) := a_e x + a_e f_e^{\mathtt{r}} + b_e^{\mathtt{r}} = a_e x + b_e^{\mathtt{r}-1}$;
11.           $\mathtt{r} := \mathtt{r}-1$;
12.     End
13.     $f(\sigma,t+\epsilon) := f(\sigma,t) + \alpha^{\mathtt{h}}\epsilon$;
14.     $t := t+\epsilon$;
15. End
16. $t_{\mathtt{h}}(\sigma) := 1$;
```

Fig. 2. Algorithm B.

Lemma 16. *Let $t_0(\sigma) = 0$ and $t_z(\sigma) = 1$ (So when Algorithm B terminates $\mathtt{h} = z$).*

(i) For all $0 \le i \le z$, at time $t^* = t_i(\sigma)$, $f(\sigma,t^*)$ uses the set of paths $\mathcal{P}_i$ as $O(t_i)$. Furthermore, if at time $t^* = t_i(\sigma)$, $\mathtt{r} = r$, then $f(\sigma,t^*)$ is $\frac{r+1}{r}$-optimal in the current network.

(ii) Let q be a path in $\mathcal{P}$ and $p \in \mathcal{P}_1$ the path chosen at Line 0.

(iia) Suppose that $0 \le i \le z-1$ and at time $t^* = t_i(\sigma)$, $\mathbf{r} = r$. Then $L_{q,r}(f(\sigma, t^*), \frac{r+1}{r}) = L_q(O(t_i), 2)$ and $L_{p,r}(f(\sigma, t^*), \frac{r+1}{r}) = L_p(O(t_i), 2) = \Psi_i$;

(iib) At time $t^* = t_z(\sigma)$, if $\mathbf{r} = r$, then $L_{q,r}(f(\sigma, t^*), \frac{r+1}{r}) \le L_q(O(t_z), 2)$ and $L_{p,r}(f(\sigma, t^*), \frac{r+1}{r}) \le L_p(O(t_z), 2) = \Psi_z$

(iii) At time $t^* = t^r(\sigma)$, for some $1 \le r \le k$,

(iiia) $f(\sigma, t^*)$ is $\frac{r+1}{r}$-optimal in the current network;

(iiib) if $r > 1$, then for each edge $e \in E$, $L_{e,r}(f(\sigma, t^*), \frac{r+1}{r}) = L_{e,r-1}(\frac{r-1}{r} f(\sigma, t^*), \frac{r}{r-1})$, which is the ϕ-delay of edge e immediately before the freezing where $\phi = \frac{r+1}{r}$. As a consequence, immediately after the freezing, i.e., Algorithm B is executing Line 12 when $t = t^r(\sigma)$ (note that then the index $\mathbf{r}$ is now set to $r-1$ and each edge has now the delay function $l_e(x) = a_e x + b_e^{\mathbf{r}}$), the unfrozen part of the flow still has the same ϕ-delay (where $\phi = \frac{\mathbf{r}+1}{\mathbf{r}}$) on each path in $\mathcal{P}$ and it is still $\frac{\mathbf{r}+1}{\mathbf{r}}$-optimal in the updated network.

(iv) f^i is a flow of value v_i for all $1 \le i \le k$. Furthermore, $f^1, \cdots, f^k$ satisfy the nesting property and they constitute a Nash equilibrium in the original network.

Proof. We prove (i)-(iii) together by induction on the number of times $t^r(\sigma)$ and $t_h(\sigma)$ are defined (i.e. the number of times that Lines 4 and 7 are executed). To simplify notation, we write $f(t)$, instead of $f(\sigma, t)$ to denote the current flow maintained by the algorithm.

Let the base case be when at time $t^* = 0$, $t_0(\sigma)$ is defined to be 0. Then (i) is trivial and there is nothing to prove in (iii) and (iia) holds because $L_{q,k}(f(t_0(\sigma)), \frac{k+1}{k}) = \sum_{e \in q} b_e = L_q(O(t_0), 2)$. For the induction step, either $t^* = t_i(\sigma)$ or $t^* = t^r(\sigma)$. In the former case, assume that at time $t^* = t_i(\sigma)$, the index $\mathtt{r} = r$. In the latter case, assume that at time $t^* = t^r(\sigma)$ the index $\mathtt{h} = i$.

Assume that at time $t_{i-1}(\sigma)$, the index $\mathtt{r} = r^\dagger$. Then by these assumptions,

$$\begin{array}{ll} t_{i-1}(\sigma) \le t^{r^\dagger}(\sigma) \le t^{r^\dagger - 1}(\sigma) \le \cdots \le t^{r+1}(\sigma) \le t^* & \text{if } r^\dagger > r, \\ t_{i-1}(\sigma) \le t^* \le t^{r^\dagger}(\sigma) & \text{if } r^\dagger = r. \end{array}$$

(In the former case, between $t_{i-1}(\sigma)$ and t^*, the current flow has been frozen at least once, while in the latter, never.)

Below we assume that $r^\dagger > r$. The case that $r^\dagger = r$ follows similar and simpler arguments.

(iiib) of the induction hypothesis implies that the ϕ-delay of path q is unchanged at $t^{r^\dagger}(\sigma), t^{r^\dagger - 1}(\sigma), \cdots, t^{r+1}(\sigma)$. (ii) of the induction hypothesis states that

$$L_{q,r^\dagger}(f(t_{i-1}(\sigma)), \frac{r^\dagger + 1}{r^\dagger}) = L_q(O(t_{i-1}), 2) \quad \text{and this values equals } \Psi_{i-1} \text{ if } q = p \in \mathcal{P}_1. \tag{13}$$

Letting

$$\tau = 1/2 \left[(t^{r^\dagger}(\sigma) - t_{i-1}(\sigma)) \frac{r^\dagger + 1}{r^\dagger} + \left[\sum_{j=r+1}^{r^\dagger - 1} (t^j(\sigma) - t^{j+1}(\sigma)) \frac{j+1}{j} \right] + (t^* - t^{r+1}(\sigma)) \frac{r+1}{r} \right],$$

then the quantity $2\tau \sum_{e \in q} a_e \alpha_e^i$ represents the increase of q's ϕ-delay from $t_{i-1}(\sigma)$ up to t^*. We claim

$$t_{i-1} + \tau \le t_i \quad \text{and this inequality holds with equality if } t \in \{t_i(\sigma)\}_{i=1}^{z-1}. \tag{14}$$

If the claim holds, then in the interval $(t_{i-1}, t_{i-1}+\tau]$, the optimal flow sticks to using $\mathcal{P}_i$. This claim follows from

$$t_i - t_{i-1} = \frac{\Psi_i - \Psi_{i-1}}{2\sum_{e\in p} a_e \alpha_e^i} \geq \frac{L_{p,r}(f(t_i(\sigma)), \frac{r+1}{r}) - L_{p,r^\dagger}(f(t_{i-1}(\sigma)), \frac{r^\dagger+1}{r^\dagger})}{2\sum_{e\in p} a_e \alpha_e^i} = \frac{2\tau \sum_{e\in p} a_e \alpha_e^i}{2\sum_{e\in p} a_e \alpha_e^i} = \tau,$$

where the first equality follows from Lemma 12(iiia) and the inequality from (13) and the fact that p's ϕ-delay at time t^* is no larger than Ψ_i. Note that this inequality holds with equality if $t \in \{t_i(\sigma)\}_{i=1}^{z-1}$. And we have

$$L_{q,r}(f(t), \frac{r+1}{r}) = L_{q,r^\dagger}(f(t_{i-1}(\sigma)), \frac{r^\dagger+1}{r^\dagger}) + 2\tau \sum_{e\in q} a_e \alpha_e^i$$
$$= L_q(O(t_{i-1}), 2) + \tau(2\sum_{e\in q} a_e \alpha_e^i) = L_q(O(t_{i-1}+\tau), 2) \leq L_q(O(t_i), 2)$$

where the second equality follow from (13) and the third from (14). So we establish (ii). (The second half of (iia) holds because $t_{i-1}+\tau = t_i$ if $t \in \{t_i(\sigma)\}_{t=1}^{z-1}$ and Lemma 12(iii)).

Now by (i) of the induction hypothesis, the flow $f(t_{i-1}(\sigma))$ uses $\mathcal{P}_{i-1}$. In the case that $t^* \in \{t_i(\sigma)\}_{i=1}^{z-1}$, from $t_{i-1}(\sigma)$ up to t^*, the current flow is updated based on α^i, therefore, $f(t^*)$ uses $\mathcal{P}_i$. (Though the current flow could have been frozen multiple times from $t_{i-1}(\sigma)$ up to t^*, each freezing does not change the set of edges used by the current flow.) And $\mathcal{P}_i$ is also used by $O(t_{i-1}+\tau)$ due to (14). Since $f(t^*)$ and $O(t_{i-1}+\tau)$ use the same set of edges, if the latter flow is 2-optimal, the former flow must be $\frac{r+1}{r}$-optimal. This proves (i).

In the case that $t^* \in \{t^r(\sigma)\}_{r=1}^k$, (iiia) can be proved by the same arguments: since $f(t^*)$ and $O(t_{i-1}+\tau)$ use the same set of edges, if the latter flow is 2-optimal, the former flow must be $\frac{r+1}{r}$-optimal. (iiib) follows by the same argument as shown in the claim immediately preceding this lemma.

Finally for (iv), it follows easily from the algorithm that flow f^i is of value v_i for all i. Moreover, the flows $f^1, \cdots, f^k$ satisfy the nesting property, since by Lemma 12(iib), $\alpha_e^i \geq 0$ for all i and e. To see that they form a Nash equilibrium in the original network, note that at time $t^* = t^r(\sigma)$, (iiia) states that the current flow $f(t^*)$ is $\frac{r+1}{r}$-optimal in the current network. So given a path p used by $f(t^*)$,

$$\sum_{e\in p} \frac{r+1}{r} a_e f_e(t^*) + b_e^r = \sum_{e\in p} (r+1) a_e f_e^r + b_e^r \leq \sum_{e\in q} \frac{r+1}{r} a_e f_e(t^*) + b_e^r = \sum_{e\in q} (r+1) a_e f_e^r + b_e^r,$$

for any other path q linking the same pair of vertices as p. Dividing the above inequality by $r+1$ gives the expression in (11). Hence $f = \sum_{i=1}^k f^i$ is a Nash equilibrium by Lemma 15. □

3.3 Comparing the Social Cost of Two Different Profiles

In this section, we prove Lemma 5 by applying Algorithm B to two different profiles σ and σ' simultaneously.

Let $h(\sigma, t)$ and $r(\sigma, t)$ denote the values of the indices h and r in the execution of Algorithm B for σ. More precisely,

$$\begin{cases} h(\sigma,t) = i \text{ if } t \in (t_{i-1}(\sigma), t_i(\sigma)] & (t_0(\sigma) \text{ is understood to be } 0) \\ r(\sigma,t) = r \text{ if } t \in (t^{r+1}(\sigma), t^r(\sigma)] & (t^{k+1}(\sigma) \text{ is understood to be } 0) \end{cases}$$

Let $C(\sigma, t)$ be the social cost of the *accumulated flow*, which is the sum of the current flow $f(\sigma, t)$ and those frozen flows, in the *original* network, i.e.,

$$C(\sigma, t) = \sum_{e \in E} \left[f_e(\sigma, t) + \sum_{j=r(\sigma,t)+1}^{k} f_e^j \right] \left[a_e(f_e(\sigma, t) + \sum_{j=r(\sigma,t)+1}^{k} f_e^j) + b_e \right].$$

We want to observe the growing speed of the social cost $C(\sigma, t)$, in other words, $\frac{dC(\sigma,t)}{dt}$[8]. Suppose that σ majorizes σ'. If we can show that

$$\frac{dC(\sigma, t)}{dt}|_{t=t^*} \le \frac{dC(\sigma', t)}{dt}|_{t=t^*} \qquad \text{for all } 0 \le t^* \le 1,\ t^* \notin \{t_i(\sigma), t_i(\sigma')\}_{\forall i}, \tag{15}$$

then we can conclude the equilibrium for σ has no greater cost than the equilibrium for σ'.

The following lemma can be proved by calculus and the observation that the accumulated flow on edge e, which is $f_e(\sigma, t) + \sum_{j=r(\sigma,t)+1}^{k} f_e^j$, can be expressed as

$$\left[\sum_{i=1}^{h(\sigma,t)-1} \alpha_e^i (t_i(\sigma) - t_{i-1}(\sigma)) \right] + (t - t_{h(\sigma,t)-1}(\sigma))\alpha_e^{h(\sigma,t)} = \alpha_e^{h(\sigma,t)} t + \sum_{i=1}^{h(\sigma,t)-1} t_i(\sigma)(\alpha_e^i - \alpha_e^{i+1}). \tag{16}$$

Lemma 17. *Let σ be an input profile of Algorithm B.*

(i) For $t \in [0, 1] \backslash \{t_i(\sigma)\}_{\forall i}$, $\frac{dC(\sigma,t)}{dt} = Y_{h(\sigma,t)} t + Z_{h(\sigma,t)}$, where $Y_{h(\sigma,t)} = \sum_{e \in E} 2a_e(\alpha_e^{h(\sigma,t)})^2$ and $Z_{h(\sigma,t)} = \sum_{e \in E} \alpha_e^{h(\sigma,t)} b_e$;
(ii) For any $h \ge 2$, $Y_{h-1} t_{h-1} + Z_{h-1} = Y_h t_{h-1} + Z_h$; moreover, $Y_{h-1} > Y_h$.
(iii) If $h' < h$, $h \ge 2$, and $t^* > t_{h-1}$, then $Y_h t^* + Z_h < Y_{h'} t^* + Z_{h'}$;
(iv) If $h' = h$, then $Y_{h'} t^* + Z_{h'} = Y_h t^* + Z_h$.

Proof. For (i), by calculus and the fact that the accumulated flow on edge $e \in E$ can be expressed as (16),

$$\frac{dC(\sigma, t)}{dt} = 2 \sum_{e \in E} a_e \alpha_e^{h(\sigma,t)} \left[(\alpha_e^{h(\sigma,t)} t + \sum_{i=1}^{h(\sigma,t)-1} t_i(\sigma)(\alpha_e^i - \alpha_e^{i+1}) \right] + \sum_{e \in E} \alpha_e^{h(\sigma,t)} b_e.$$

So if we can show that

$$\sum_{e \in E} a_e \alpha_e^{h(\sigma,t)} (\alpha_e^i - \alpha_e^{i+1}) = 0, \forall 1 \le i \le h(\sigma, t) - 1, \tag{17}$$

we can prove (i). We make use of the following fact that has been proved elsewhere.

Proposition 18. [8] *Let z be a flow of value η and c the vector of cost indexed by $e \in E$ such that for every path $p \in \mathcal{P}' \subseteq \mathcal{P}$, $\sum_{e \in p} c_e = \kappa$, a fixed constant, and $z_e = 0$ for each $e \notin \mathcal{P}'$. Then $\sum_{e \in E} c_e z_e = \kappa\eta$.*

[8] To be precise, $C(\sigma, t)$ is not differentiable in the breakpoints $t_i(\sigma)$. Hence the domain of $\frac{dC(\sigma,t)}{dt}$ is $[0, 1] \backslash \{t_i(\sigma)\}_{\forall i}$. For our purpose, it suffices to consider the open intervals between these breakpoints, because the functions $C(\sigma, t)$ and $C(\sigma', t)$ are both continuous.

By Lemma 12(iic) and (iib), for any path $p \in \mathcal{P}_{h(\sigma,t)}$, $\sum_{e \in p} a_e \alpha_e^{h(\sigma,t)}$ is constant and α^i and α^{i+1} are flows of value 1. Thus, we can make use of Proposition 18 as follows: let $\mathcal{P}' := \mathcal{P}_{h(\sigma,t)}$, $c_e := a_e \alpha_e^{h(\sigma,t)}$, and $z := \alpha^i$ or $z := \alpha^{i+1}$ (note that $z_e = 0$, $\forall e \notin \mathcal{P}'$ by Lemma 12(iib).) This gives (17) and hence the proof of (i).

For (ii), let $\sigma^o = (1, 0, \cdots, 0)$ denote the optimal flow profile. Then (i) implies that

$$\frac{dC(\sigma^o, t)}{dt}|_{t=t^*} = Y_{h-1}t^* + Z_{h-1} \text{ for } t^* \in (t_{h-2}, t_{h-1}), \text{ and } \frac{dC(\sigma^o, t)}{dt}|_{t=t^*} = Y_h t^* + Z_h \text{ for } t^* \in (t_{h-1}, t_h).$$

If $Y_{h-1}t_{h-1} + Z_{h-1} < Y_h t_{h-1} + Z_h$, then choose a small $\epsilon > 0$ so that $t_{h-1} + \epsilon < t_h$, and $Y_{h-1}t + Z_{h-1} < Y_h t + Z_h$ in the interval $t \in [t_{h-1}, t_{h-1} + \epsilon]$. This implies that when the flow value is increased from t_{h-1} to $t_{h-1} + \epsilon$, if the optimal flow sticks to the set of paths $\mathcal{P}_{h-1}$, the growing speed of its social cost, would have been less than if it uses $\mathcal{P}_h$, an impossibility. More specifically, define a flow $\tilde{O}(t)$ as follows,

$$\forall e \in E, \tilde{O}_e(t) = \alpha_e^{h-1} t + \sum_{i=1}^{h-2} t_i(\alpha_e^i - \alpha_e^{i+1}),$$

Then $\tilde{O}(t_{h-1}) = O(t_{h-1})$ and $\tilde{O}(t)$ uses $\mathcal{P}_{h-1}$ when $t_{h-1} \le t \le t_{h-1} + \epsilon$. Letting $C(\tilde{O}(t))$ denote the cost of $\tilde{O}(t)$, then by the same calculation as has been done in (i),

$$\text{if } \; t_{i-1} \le t \le t + \epsilon, \qquad \frac{dC(\tilde{O}(t))}{dt} = Y_{h-1}t + Z_{h-1} < Y_h t + Z_h,$$

implying that the cost of $\tilde{O}(t_{h-1} + \epsilon)$ is strictly less than the cost $O(t_{h-1} + \epsilon)$, a contradiction. On the other hand, if $Y_{h-1}t_{h-1} + Z_{h-1} > Y_h t_{h-1} + Z_h$, then choose similarly an arbitrarily small $\epsilon > 0$ so that $t_{h-1} - \epsilon > t_{h-2}$, and $Y_{h-1}t + Z_{h-1} > Y_h t + Z_h$ in the interval $t \in [t_{h-1} - \epsilon, t_{h-1}]$. Then the optimal flow should have shifted to using the set of paths $\mathcal{P}_h$, in the interval $t \in [t_{h-1} - \epsilon, t_{h-1}]$ by the same arguments as above. But this contradicts Lemma 12(i). So we establish $Y_{h-1}t_{h-1} + Z_{h-1} = Y_h t_{h-1} + Z_h$.

To show the last part of (ii), suppose that $Y_{h-1} \le Y_h$. Then choose an $\epsilon > 0$ so that $t_{h-1} + \epsilon < t_h$, and $Y_{h-1}t + Z_{h-1} \le Y_h t + Z_h$ in the interval $t \in [t_{h-1}, t_{h-1} + \epsilon]$. Then this implies that in this interval, if the optimal flow sticks to the set of paths in $\mathcal{P}_{h-1}$, instead of using $\mathcal{P}_h$, either its social cost would have been smaller (contradicting Lemma 12(i)), or there exist multiple optimal flows of the same value (one using $\mathcal{P}_{h-1}$ and the other using $\mathcal{P}_h$), thus contradicting Lemma 9. This completes (ii).

For (iii), observe that by (ii),

$$\text{if } t^* > t_{i-1} \text{ and } i \ge 2, \text{ then } Y_i t^* + Z_i < Y_{i-1}t^* + Z_{i-1}.$$

Since $t^* > t_{h-1}$ and $h \ge 2$, $t^* > t_{i-1}$ for all $2 \le i \le h$. Repeatedly applying the above inequality gives

$$Y_h t^* + Z_h < Y_{h-1}t^* + Z_{h-1} < \cdots < Y_{h'}t^* + Z_{h'}.$$

(iv) is trivial. □

Our proof of Lemma 5 has a geometric interpretation. By Lemma 17(i), if we treat $\frac{dC(\sigma,t)}{dt}$ as a function of t with domain $[0,1] \backslash \{t_i(\sigma)\}_{\forall i}$, then such a function is piece-wise linear (but not necessarily continuous, unless σ is the optimal flow profile). To prove (15), we just need to show that the function $\frac{dC(\sigma,t)}{dt}$ is always above the other function $\frac{dC(\sigma',t)}{dt}$ in the intervals $[0,1] \backslash \{t_i(\sigma), t_i(\sigma')\}_{\forall i}$.

Specifically, by Lemma 17(i), we have

$$\frac{dC(\sigma,t)}{dt}|_{t=t^*} = Y_{h(\sigma,t^*)}t^* + Z_{h(\sigma,t^*)}, \frac{dC(\sigma',t)}{dt}|_{t=t^*} = Y_{h(\sigma',t^*)}t^* + Z_{h(\sigma',t^*)}.$$

In the following, we will establish the following fact.

$$\text{if } h(\sigma',t^*) \leq h(\sigma,t^*), \text{ then } Y_{h(\sigma,t^*)}t^* + Z_{h(\sigma,t^*)} \leq Y_{h(\sigma',t^*)}t^* + Z_{h(\sigma',t^*)}. \tag{18}$$

If this fact holds, then to prove (15), all we need to do is to find a way to compare $h(\sigma,t^*)$ and $h(\sigma',t^*)$ (See Lemmas 19 and 20.) Before doing so, let us first explain how this fact can be established. Supposing that the "if" statement in (18) holds, there are two cases:

1. if $h(\sigma,t^*) = h(\sigma',t^*)$. The "then" statement in (18) follows trivially from Lemma 17(iv).
2. if $h(\sigma,t^*) > h(\sigma',t^*)$ (implying $h(\sigma,t^*) \geq 2$), then we will show in Lemma 21 that $t^* > t_{h(\sigma,t^*)-1}$. The "then" statement in (18) would follow from Lemma 17(iii)[9]. .

Recall that in Algorithm B, the index h is incremented every time the ϕ-delay of a path $p \in \mathcal{P}_1$ reaches a threshold Ψ_i (see Line 2). Thus, $h(\sigma,t^*)$ is determined by how much p's ϕ-delay has increased during the interval $[0,t^*]$. We use the following device to capture the increase of p's ϕ-delay.

Let $r^j(\sigma)[t^\dagger,t^\ddagger]$ denote the amount of time in the interval $[t^\dagger,t^\ddagger]$ during which Algorithm B, with input σ, sets the index r to be j. Precisely,

$$r^j(\sigma)[t^\dagger,t^\ddagger] = \max\{0, \min\{t^\ddagger, t^j(\sigma)\} - \max\{t^\dagger, t^{j+1}(\sigma)\}\}.$$

Now suppose that in the interval $[t^\dagger,t^\ddagger]$, the current flow $f(\sigma,t)$ uses the set of paths $\mathcal{P}_h$, then the quantity $\sum_{e\in p} a_e\alpha_e^h \sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[t^\dagger,t^\ddagger]$ is exactly the increase of path p's ϕ-delay in $[t^\dagger,t^\ddagger]$. The proof of following lemma uses this observation.

Lemma 19. *Suppose that $t^* \in [0,1]$. If $\sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[0,t^*] \geq \sum_{j=1}^k \frac{j+1}{j} r^j(\sigma')[0,t^*]$, then $h(\sigma,t^*) \geq h(\sigma',t^*)$.*

Proof. Let $p \in \mathcal{P}_1$.

Claim. $\sum_{i=1}^{h(\sigma,t^*)-1} \frac{\Psi_i - \Psi_{i-1}}{\sum_{e\in p} a_e\alpha_e^i} < \sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[0,t^*] \leq \sum_{i=1}^{h(\sigma,t^*)} \frac{\Psi_i - \Psi_{i-1}}{\sum_{e\in p} a_e\alpha_e^i}$.

Proof. By Lemma 16(iia) , if $1 \leq i \leq h(\sigma,t^*)-1$, Ψ_i is p's ϕ-delay at time $t_i(\sigma)$; moreover, p's ϕ-delay is at most $\Psi_{h(\sigma,t^*)}$ at time t^*. The growth of p's ϕ-delay from $t_{i-1}(\sigma)$ to $t_i(\sigma)$ is $\sum_{e\in p} a_e\alpha_e^i \sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[t_{i-1}(\sigma), t_i(\sigma)]$, therefore,

$$\sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[t_{i-1}(\sigma), t_i(\sigma)] = \frac{\Psi_i - \Psi_{i-1}}{\sum_{e\in p} a_e\alpha_e^i}, \qquad \forall 1 \leq i \leq h(\sigma,t^*)-1 \tag{19}$$

$$\sum_{j=1}^k \frac{j+1}{j} r^j(\sigma)[t_{h(\sigma,t^*)-1}(\sigma), t^*] \leq \frac{\Psi_{h(\sigma,t^*)} - \Psi_{h(\sigma,t^*)-1}}{\sum_{e\in p} a_e\alpha_e^{h(\sigma,t^*)}} \qquad (\text{since } t^* \leq t_{h(\sigma,t^*)}(\sigma)) \tag{20}$$

[9] Recall our earlier remark at the end of Section 3.1: the growing speed of the social cost using a larger set of paths $\mathcal{P}_{h(\sigma,t^*)}$, which is $Y_{h(\sigma,t^*)}t^* + Z_{h(\sigma,t^*)}$, is less than the growing speed of the social cost using a smaller set of paths $\mathcal{P}_{h(\sigma',t^*)}$, which is $Y_{h(\sigma',t^*)}t^* + Z_{h(\sigma',t^*)}$, under a *certain condition*, which is $t^* > t_{h(\sigma,t^*)-1}$ and it is proved in Lemma 21.

Summing up (19) for all $1 \le i \le h(\sigma, t^*) - 1$ and (20), we derive the second inequality. Summing up (19) for all $1 \le i \le h(\sigma, t^*) - 1$ and the LHS of (20) gives the first inequality (note that the quantity $\sum_{j=1}^{k} r^j(\sigma)[t_{h(\sigma,t^*)-1}(\sigma), t^*] > 0$ is due to the way we define $h(\sigma, t)$). □

We now prove the lemma by contradiction. If $h(\sigma, t^*) < h(\sigma', t^*)$, then by the above claim,

$$\sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma)[0, t^*] \le \sum_{i=1}^{h(\sigma,t^*)} \frac{\Psi_i - \Psi_{i-1}}{\sum_{e \in p} a_e \alpha_e^i} \le \sum_{i=1}^{h(\sigma',t^*)-1} \frac{\Psi_i - \Psi_{i-1}}{\sum_{e \in p} a_e \alpha_e^i} < \sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma')[0, t^*],$$

a contradiction to the statement of the lemma. □

Lemma 19 suggests the way to compare $h(\sigma, t^*)$ and $h(\sigma', t^*)$.

Lemma 20. *If $\sigma = (v_1, \cdots, v_k)$ majorizes $\sigma' = (v'_1, \cdots, v'_k)$, then for any $t^* \in [0, 1]$, $\sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma)[0, t^*] \ge \sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma')[0, t^*]$.*

Proof. We define a function $g(t) = \sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma)[0, t] - \sum_{j=1}^{k} \frac{j+1}{j} r^j(\sigma')[0, t]$ and we show that $g(t)$ is never negative in $t \in [0, 1]$. Observe that $g(t)$ is continuous, and piecewise linear with breakpoints in $t^r(\sigma)$ and $t^r(\sigma')$ for all $1 \le r \le k$. Moreover, between two breakpoints, the slope of $g(t)$ is $\frac{r(\sigma,t)+1}{r(\sigma,t)} - \frac{r(\sigma',t)+1}{r(\sigma',t)}$. If given any $t \in [0, 1]$, $r(\sigma, t) \le r(\sigma', t)$, then $g(t)$ is obviously never negative. So suppose that at $t^\dagger < 1$, $r(\sigma, t^\dagger) > r(\sigma', t^\dagger)$ so that the slope of $g(t)$ is negative at $t^\dagger$.

Claim. Let $t^\dagger$ be as defined. Then there exists $t^\ddagger > t^\dagger$, $t^\ddagger \in \{t^r(\sigma)\}_{r=2}^{k}$, so that

(i) Between the interval $[t^\dagger, t^\ddagger)$, the slope of $g(t)$ is always negative and if $t^\ddagger < 1$, then the slope of $g(t)$ between $t^\ddagger$ and the next breakpoint of $g(t)$ is 0;
(ii) $g(t^\ddagger) \ge 0$.

Proof. Order all points $t^j(\sigma)$ and $t^j(\sigma')$ that are no less than $t^\dagger$ non-decreasingly. We look at these points by this order. Each time we see a point $t^j(\sigma)$ (resp. $t^j(\sigma')$), then at $t = t^j(\sigma)$ (resp. at $t = t^j(\sigma')$), the difference between $r(\sigma, t)$ and $r(\sigma', t)$ becomes smaller (respectively, larger). (Recall the way we define $r(\sigma, t)$: when $t \in (t^{r+1}(\sigma), t^r(\sigma)]$, $r(\sigma, t) = r$. And *beyond* $t^r(\sigma)$, the index $\mathtt{r}$ decreases.) By this procedure, either: case (1) we find a point $t^r(\sigma)$ so that the slope between $t^r(\sigma)$ and the next breakpoint is 0, or case (2) we do not find such point and reach the last point $t^r(\sigma) = 1$. In both cases, let $t^\ddagger = t^r(\sigma)$. And we argue that $r \ge 2$. In the former case, $t^r(\sigma) < 1 = t^1(\sigma)$, hence $r > 1$. In the latter case, we claim that $t^2(\sigma) = t^1(\sigma) = 1$. In this case, we can let $t^\ddagger$ be $t^2(\sigma)$. If the claim does not hold, then $t^2(\sigma) < t^1(\sigma) = 1$. But in the interval $(t^2(\sigma), t^1(\sigma)]$, $r(\sigma, t) = 1 \le r(\sigma', t)$, a contradiction to case (2). Now (i) is proved.

For (ii), note that the above procedure for picking $t^{\ddagger} = t^r(\sigma)$ implies that $t^r(\sigma') \leq t^r(\sigma)$, and for all $j \geq r$, $t^j(\sigma)$, $t^j(\sigma') \leq t^r(\sigma)$ and for all $j < r$, $t^j(\sigma)$, $t^j(\sigma') \geq t^r(\sigma)$. Let $t^{k+1}(\sigma) := t^{k+1}(\sigma') := 0$.

$$\begin{aligned}
g(t^{\ddagger}) &= \sum_{j=r}^{k} \frac{j+1}{j}(t^j(\sigma) - t^{j+1}(\sigma)) - \left[(\sum_{j=r}^{k} \frac{j+1}{j}(t^j(\sigma') - t^{j+1}(\sigma'))) + \frac{r}{r-1}(t^r(\sigma) - t^r(\sigma')) \right] \\
&= \sum_{j=r}^{k} \frac{t^j(\sigma') - t^j(\sigma)}{j(j-1)} \\
&= \sum_{j=r}^{k} \frac{j(v'_j - v_j) + \sum_{i=j+1}^{k} v'_i - v_i}{j(j-1)} \qquad \text{(using the fact that } t^j(\sigma) = jv_j + \textstyle\sum_{i=j+1}^{k} v_i.\text{)} \\
&= \frac{\sum_{j=r}^{k} v'_j - v_j}{r-1} \\
&\geq 0. \qquad \text{(using the fact that } \sigma \text{ majorizes } \sigma'.\text{)} \qquad \square
\end{aligned}$$

Now by (i) of the claim, if the slope of $g(t)$ is negative at $t^{\dagger}$, we can find a point $t^{\ddagger} > t^{\dagger}$ so that either the slope of $g(t)$ becomes 0 beyond $t^{\ddagger}$, or $t^{\ddagger} = 1$. As $g(t^{\ddagger}) \geq 0$ by (ii) of this claim, we prove the lemma. $\square$

Lemma 21. *If $h(\sigma, t^*) \geq 2$, then $t^* > t_{h(\sigma,t^*)-1}$.*

Proof. Let $\sigma^o = (1, 0, \cdots, 0)$ be the optimal flow profile. Then σ^o majorizes σ. By Lemmas 19 and 20, we have $h(\sigma^o, t^*) \geq h(\sigma, t^*) > h(\sigma, t^*) - 1 = h(\sigma^o, t_{h(\sigma,t^*)-1})$. Thus $t^* > t_{h(\sigma,t^*)-1}$. $\square$

Proof of (15) and Lemma 5 As σ majorizes σ', by Lemmas 19 and 20, $h(\sigma, t^*) \geq h(\sigma', t^*)$ for all $t^* \in [0, 1]$. Now, by Lemma 17(i)(iii)(iv) and Lemma 21, given any $t^* \in [0, 1] \backslash \{t_i(\sigma), t_i(\sigma')\}_{\forall i}$,

$$\frac{dC(\sigma, t)}{dt}|_{t=t^*} = Y_{h(\sigma,t^*)}t^* + Z_{h(\sigma,t^*)} \leq Y_{h(\sigma',t^*)}t^* + Z_{h(\sigma',t^*)} = \frac{dC(\sigma', t)}{dt}|_{t=t^*},$$

establishing (15). And Lemma 5 follows from (15). $\square$

Acknowledgment

I thank Lisa Fleischer and Kurt Mehlhorn for their comments on earlier drafts. I also thank Khaled Elbassioni and Michael Sagraloff for discussions on several technical issues.

References

1. Ahuja, R., Magnanti, T., and Orlin, J. *Network Flows.* Prentice Hall, 1993.
2. Aland, S., Dumrauf, D., Gairing, M., Monien, B., and Schoppmann, F.
3. Albers, S. Strong and pareto price of anarchy in congestion games. *SIAM Journal on Computing 38*, 6 (2009), 2273–2302.
4. Altman, E., Basar, T., Jimenez, T., and Shimkin, N. Competitive routing in networks with polynomial costs. *IEEE Transactions on Automatic Control 47*, 1 (Jan. 2002), 92–96.
5. Andelman, N., Feldman, M., and Mansour, Y. Strong price of anarchy. In *SODA* (2007), pp. 189–198.
6. Aumann, R. Acceptable points in geneeral cooperative n-person games. *Contributions to the Theory of Games 4* (1959).

7. Awerbuch, B., Azar, Y., and Epstein, A. The price of routing unsplittable flow. In *STOC* (New York, NY, USA, 2005), ACM, pp. 57–66.
8. Bhaskar, U., Fleischer, L., and Huang, C.-C. The price of collusion in series-parallel networks. In *IPCO* (2010).
9. Bollobás, B. *Modern Graph Theory*. Springer, 1998.
10. Catoni, S., and Pallottino, S. Traffic equilibrium paradoxes. *Transportation Science 25*, 3 (August 1991), 240–244.
11. Chien, S., and Sinclair, A. Strong and pareto price of anarchy in congestion games. In *ICALP* (2009), pp. 279–291.
12. Cominetti, R., Correa, J. R., and Stier-Moses, N. E. The impact of oligopolistic competition in networks. *Operations Research 57*, 6 (2009), 1421–1437.
13. Dimitris Fotakis, S. C. K., and Spirakis, P. Selfish unsplittable flows. In *TCS* (2005), Springer-Verlag, pp. 593–605.
14. Epstein, A., Feldman, M., and Mansour, Y. Strong equilibrium in cost-sharing connection games. In *EC* (2007), pp. 84–92.
15. Fiat, A., Levy, M., Kaplan, H., and Olonetsky, S. Strong price of anarchy for machine load balancing. In *ICALP* (2007), pp. 583–594.
16. Fotakis, D., Kontogiannis, S., and Spirakis, P. Atomic congestion games among coalitions. *ACM Transactions on Algorithms 4*, 4 (2008). The conference version appeared in ICALP 2006.
17. Hariharan, R., Kavitha, T., and Mehlhorn, K. Faster algorithms for minimum cycle basis in directed graphs. *SIAM Journal on Computing 38*, 4 (2008), 1430–1447.
18. Harks, T. Stackelberg strategies and collusion in network games with splittable flow. In *WAOA* (2008), pp. 133–146.
19. Harks, T., Höfer, M., and Max Klimm, A. S. Computing pure nash and strong equilibria in bottleneck congestion games. In *ESA* (2010), pp. 29–38.
20. Harks, T., and Klimm, M. On the existence of pure nash equilibria in weighted congestion games. In *ICALP* (2010), pp. 79–89.
21. Harks, T., Klimm, M., and Möhring, R. Strong nash equilibria in games with the lexicographical improvement property. In *WINE* (2009), pp. 463–470.
22. Hayrapetyan, A., Tardos, E., and Wexler, T. The effect of collusion in congestion games. In *STOC* (New York, NY, USA, 2006), ACM Press, pp. 89–98.
23. Koutsoupias, E., and Papadimitriou, C. Worst-case equilibria. In *in STACS* (1999), pp. 404–413.
24. Nocedal, J., and Wright, S. *Numerical Optimization*. Springer, 2006.
25. Orda, A., Rom, R., and Shimkin, N. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking 1*, 5 (1993), 510–521.
26. Rosen, J. B. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica 33*, 3 (July 1965), 520–534.
27. Roughgarden, T. *Selfish Routing and the Price of Anarchy*. The MIT Press, 2005.
28. Roughgarden, T. Selfish routing with atomic players. In *SODA* (2005), pp. 1184–1185.
29. Roughgarden, T. Intrinsic robustness of the price of anarchy. In *STOC* (2009). 513-522.
30. Roughgarden, T., and Schoppmann, F. Local smoothness and the price of anarchy in atomic splittable congestion game. In *SODA* (2011).
31. Roughgarden, T., and Tardos, E. How bad is selfish routing? *Journal of the ACM 49*, 2 (March 2002), 236–259.
32. Wardrop, J. G. Some theoretical aspects of road traffic research. In *Proc. Institute of Civil Engineers, Pt. II* (1952), vol. 1, pp. 325–378.

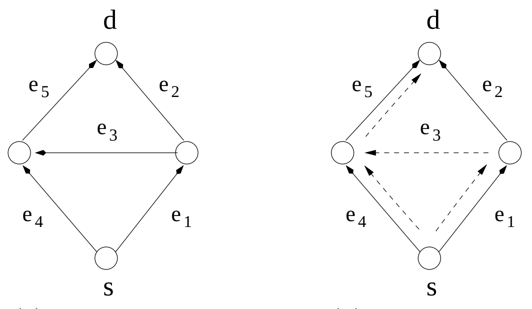
A Counter-Examples

A.1 Affine Delays in the Braess Graph

To construct an example in which the collusion increases the social cost, as indicated by Theorem 3, we need to avoid well-designed networks. For this purpose, we use the Braess graph shown in Figure 3(a), which is the smallest non-series-parallel graph, to get around Proposition 2. The delay functions of the 5 edges are shown in Table 1(a). When $t \in (1.083333, 1.973568]$, the flow on edge e_3 can be expressed as $O_{e_3}(t) = -0.468041t + 0.923711$, hence this network is not well-designed.

Initially, one player has 2.4 units and the other six players have 0.1 unit each. Figure 3(b) shows their flow patterns in the pre-collusion equilibrium, where the solid lines are the flow of the player with 2.4 units while the dotted lines are the flows of the other six players. The latter six players have identical flows in the equilibrium. It can been seen that this Nash equilibrium does not satisfy the nesting property.

Suppose now that every three of the players who have 0.1 unit flow decide to form themselves into a coalition. Thus, after the collusion, there are three players, one with 2.4 units while the other two with 0.3 unit each. The flow pattern of the post-collusion equilibrium is similar to the previous as shown in Figure 3(b): the solid lines are the flow of the player with 2.4 units while the dotted lines are the identical flows of the other two players. The social cost in this equilibrium is higher than that in the previous equilibrium.



(a) Braess graph (b) Flow patterns

Fig. 3. The graph and the flow patterns. Note that in (b), for both the pre-collusion and the post-collusion equilibria, the solid lines are the flow pattern of the player with the largest flow value and the dotted lines are the flow pattern of the players with smaller flow values.

(a) Delay Functions

Edge	Delay function
e_1	$7x$
e_2	$1.8x + 18$
e_3	$x + 2$
e_4	$2x + 6$
e_5	$7x$

(b) Flow values in the pre-collusion equilibrium

Edge	Large player	(Each of the) Small players
e_1	1.002592	0.01681
e_2	1.002592	0
e_3	0	0.01681
e_4	1.397408	0.08319
e_5	1.397408	0.1

(c) Flow values in the post-collusion equilibrium

Edge	Large player	(Each of the) Small Players
e_1	1.001325	0.052936
e_2	1.001325	0
e_3	0	0.052936
e_4	1.398675	0.247064
e_5	1.398675	0.3

Table 1. The delay functions and the flow values in both Nash equilibria

(a) Flow value, delay, and cost in the pre-collusion equilibrium

Edge	Flow value	Delay	Cost
e_1	1.103449651046859	7.724147557328014	8.523207926768047
e_2	1.002592223330010	19.80466600199402	19.85600411924744
e_3	0.1008574277168492	2.100857427716849	0.2118870761593580
e_4	1.896550348953141	9.793100697906281	18.57310854594740
e_5	1.997407776669990	13.98185443668993	27.92746478411227
SUM			75.09167245223452

(b) Marginal costs of players in the pre-collusion equilibrium

Path	Large player	(Each of the) Small Players
e_1-e_2	36.35162512462612	27.64648055832502
e_1-e_3-e_5	40.60685942173480	24.64133599202393
e_4-e_5	36.35162512462612	24.64133599202393

(c) Flow value, delay, and cost in the post-collusion equilibrium

Edge	Flow value	Delay	Cost
e_1	1.107196467991170	7.750375275938190	8.581188131124852
e_2	1.001324503311258	19.80238410596026	19.82861242927941
e_3	0.1058719646799118	2.105871964679912	0.2229528022650080
e_4	1.892803532008830	9.785607064017661	18.52223161362319
e_5	1.998675496688742	13.99072847682119	27.96292618744792
SUM			75.11791116374037

(d) Marginal costs of players in the post-collusion equilibrium

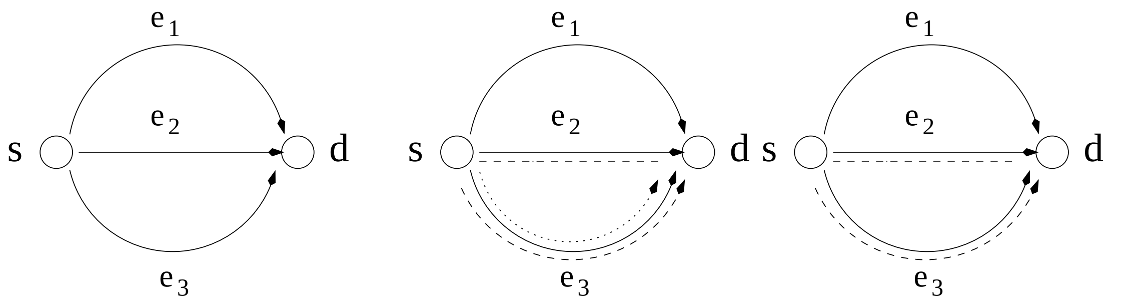
Path	Large player	(Each of the) Small Players
e_1-e_2	36.3644150110375	27.92331125827815
e_1-e_3-e_5	40.59403973509934	26.37046357615894
e_4-e_5	36.3644150110375	26.37046357615894

Table 2. Details of the two Nash equilibria

A.2 Convex Delays in the Parallel-Links Graph

Consider the three-links graph shown in Figure 4(a) whose delay functions are shown in Table 3(a). Initially, we have three players. Player 1 has 200 units, player 2 has 20.9 units, and player 3 has 0.1 unit. The flow patterns in the pre-collusion equilibrium are shown in Figure 4(b).

Suppose that players 2 and 3 decide to form a coalition, with a total flow of 21 units. The new flow patterns in the post-collusion equilibrium are now shown in Figure 4(c). The social cost of this equilibrium is higher than in the pre-collusion equilibrium.



(a) Three links (b) Flow patterns in the pre-collusion equilibrium (c) Flow patterns in the post-collusion equilibrium

Fig. 4. The solid lines in (b) and (c) are the flow pattern of the player with the largest flow value, the dotted lines the player with the second largest value, and the thinly dotted line the player with the smallest value.

(a) Delay functions

Edge	Delay function
e_1	$20x + 5000$
e_2	$x^2 + 500$
e_3	x^{11}

(b) Flow values in the pre-collusion equilibrium

Edge	Largest player	Second largest player	Smallest player
e_1	152.5058085	0	0
e_2	46.36711109	20.18230154	0
e_3	1.127080409	0.7176984568	0.1

(c) Flow values in the post-collusion equilibrium

Edge	Largest player	Second largest
e_1	152.4922717	0
e_2	46.32694762	20.243744
e_3	1.180780656	0.7562559985

Table 3. The delay functions and the flow values in both Nash equilibria

(a) Flow value, delay, and cost in the pre-collusion equilibrium

Edge	Flow value	Delay	Cost
e_1	152.5058085	8050.11617	1227689.475024774
e_2	66.54941263	4928.82432	328010.3635454956
e_3	1.944778865	1505.12472	2927.134751868525
SUM			1558626.973322137

(b) Marginal costs of players in the pre-collusion equilibrium

Path	Largest player	Second largest player	Smallest player
e_1	11100.23234	8050.1162	8050.11617
e_2	11100.23234	7615.0649	4928.82432
e_3	11100.23234	7615.0649	2356.44886

(c) Flow value, delay, and cost in the post-collusion equilibrium

Edge	Flow value	Delay	Cost
e_1	152.4922717	8049.845434	1227539.217064532
e_2	66.57069162	4931.656983	328303.8161752776
e_3	1.9370366545	1440.509837	2790.320355463135
SUM			1558633.353595273

(d) Marginal costs of players in the post-collusion equilibrium

Path	Largest player	Second largest player
e_1	11099.69087	8049.845434
e_2	11099.69087	7626.937061
e_3	11099.69087	7626.937061

Table 4. Details of the two Nash equilibria

B Proof of Proposition 2

We first define series-parallel graphs. Given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and vertices $v_1 \in V_1$, $v_2 \in V_2$, the operation merge(v_1, v_2) creates a new graph $G' = (V' = V_1 \cup V_2, E' = E_1 \cup E_2)$, replaces v_1 and v_2 in V' with a single vertex v, and replaces each edge $e \in E'$ incident to v_1 or v_2 by an edge incident to v.

Definition 22. *A tuple (G, s, d) is* series-parallel *if G is a single edge $e = (s, d)$, or is obtained by a series or parallel composition of two series-parallel graphs (G_1, s_1, d_1) and (G_2, s_2, d_2). Vertices s and d are* terminals *of G. And series and parallel compositions are defined as follows.*
– Parallel Composition*: $s = \mathit{merge}(s_1, s_2)$, $d = \mathit{merge}(d_1, d_2)$;*
– Series Composition*: $s := s_1$, $d := d_2$, $v = \mathit{merge}(s_2, d_1)$.*

In the following proof, we implicitly assume that all edges are directed from the source to the destination and the graph is acyclic in the directed edges. This does not hurt generality, since any edge not on an s-d path is not used in an optimal flow, and no optimal flow is sent along a directed cycle.

Proof of Proposition 2: Let $O(t)$ and $O(t')$ be optimal flows of values t and t' and $t > t'$. Suppose that the proposition does not hold. Then in the pseudo-flow $z = O(t) - O(t')$, there exists some edge e^* on which $z_{e^*} < 0$. Let $G' = (V', E')$ be the largest subgraph of G containing e such that G' is also series-parallel and $\sum_{e \in E'} z_e < 0$. Then there exists another subgraph $G'' = (V'', E'')$ such that $\sum_{e \in E''} z_e > 0$ and G'' and G' share the same two terminals s' and d'.

Claim. : There exists an s'-d' path p' in G' so that $z_e < 0$, $\forall e \in p'$ and another s'-d' path p'' in G'' so that $z_e > 0$, $\forall e \in p''$.

If the claim holds, then $\sum_{e \in p'} O_e(t) < \sum_{e \in p'} O_e(t')$ and $\sum_{e \in p''} O_e(t) > \sum_{e \in p''} O_e(t')$. Recall that optimal flow is a Nash equilibrium of one player, so we can apply Lemma 8 and the preceding two inequalities to derive

$$\begin{aligned}\sum_{e \in p''} l_e(O_e(t)) + O_e(t) l'_e(O_e(t)) &\le \sum_{e \in p'} l_e(O_e(t)) + O_e(t) l'_e(O_e(t)) < \sum_{e \in p'} l_e(O_e(t')) + O_e(t') l'_e(O_e(t')) \\ &\le \sum_{e \in p''} l_e(O_e(t')) + O_e(t') l'_e(O_e(t')) < \sum_{e \in p''} l_e(O_e(t)) + O_e(t) l'_e(O_e(t)),\end{aligned}$$

a contradiction. □

Proof of the Claim We only prove the first part. Suppose that G' is composition of $G'_1 = (V'_1, E'_1)$ and $G'_2 = (V'_2, E'_2)$. If G' is resulted from the series-composition of G'_1 and G'_2, then $\sum_{e \in E'_1} z_e = \sum_{e \in E'_2} z_e < 0$. If G' is resulted from the parallel-composition of G'_1 and G'_2, then either $\sum_{e \in E'_1} z_e < 0$ or $\sum_{e \in E'_2} z_e < 0$. In both cases, we can apply induction to find out a path p' in G'_1 and/or G'_2 so that $z_e < 0$ if $e \in p'$. □

C Proof of Theorem 6

In this section, we assume that the coefficient of all delay functions $\{a_e, b_e\}_{e \in E}$ are rationals. We discuss how to verify whether a network is well-designed and show that in this verifying process, we can also find out the vectors α^i and the ϕ-delay thresholds Φ_i, that are used in Algorithms A and B.

We assume that all rationals are stored in the form of fractions; we also assume that the coefficients $\{a_e, b_e\}_{e \in E}$ are integers. The latter assumption does not cause loss of generality, because we can rescale the flows. In particular, let M be the least common multiple of the denominators of $\{a_e, b_e\}_{e \in E}$ and let $\{Ml_e\}_{e\in E}$ be the set of delay functions. Then an equilibrium flow f (of which the optimal flow is a special case) with respect to $\{Ml_e\}_{e\in E}$ implies an equilibrium flow f/M with respect to $\{l_e\}_{e\in E}$ and vice versa. Also, after the rescaling of the delay functions, the new input size is still bounded by a polynomial of the original input size. In the remaining, let $a_{\max} := \max\{a_e\}_{e\in E}$, $b_{\max} := \max\{b_e\}_{e\in E}$, $|V| = n$ and $|E| = m$.

Suppose that the network is well-designed "up to t_{h-1}.": that is, $O(t)$ is monotonically non-decreasing in $(0, t_{h-1}]$ on all edges E and we have found out the ϕ_i-delay thresholds Ψ_i and the value t_i by which the optimal flow changes the set of paths for $0 \le i \le h-1$ (see Lemma 12). In the h-th stage, the main tasks include

- find out the next set of paths $\mathcal{P}_h$ used by the optimal flow when $t \in (t_{h-1}, t_h]$, where t_h is the largest value by which the optimal flow uses the set of paths $\mathcal{P}_h$; (in case that $\mathcal{P}_h$ happens to be the last set of paths used by the optimal flow, assume t_h to be some arbitrarily large value).
- check that when $t \in (t_{h-1}, t_h]$, the optimal flow is monotonically non-decreasing in all edges and if it is, calculate the exact values in the vector α^h,
- find out the flow value t_h by which the optimal flow shifts again to another set of paths different from $\mathcal{P}_h$, when $\mathcal{P}_h$ is not the last set of paths used by the optimal flow.

The following three sections deal with these issues separately. In the last section, we argue that all operations can be done in polynomial time and the bit sizes of all values used in our algorithm are also polynomially bounded.

C.1 Finding out $\mathcal{P}_h$

It is natural to try to use convex programming to find out $\mathcal{P}_h$ used by the optimal flow $O(t)$ when $t \in (t_{h-1}, t_h]$. However, there is no prior knowledge of t_h; and even if it is given, it may happen that $O_e(t_h)$ is very small for some $e \in E$, then the approximate integral solution returned by convex programming may not correctly give the true set of paths used by $O(t)$.

To get around these issues, we trim the graph G so that if G is really well-designed, in the trimmed graph of G, the optimal flow would stick to the same set of paths $\mathcal{P}_h$ for all $t > t_{h-1}$ and then we can choose an arbitrary large $t > t_{h-1}$ to apply convex programming.

Define a subgraph $G(V, E')$ whose s-d paths form a superset of $\mathcal{P}_h$ Suppose that the given network is well-designed. Then by Lemma 12(iii), all paths in $\mathcal{P}_h$ have the same minimum 2-delay Ψ_{h-1} in $O(t_{h-1})$ among all paths in $\mathcal{P}$. We intend to find out the subgraph whose edges consist of the union of all these s-d paths whose 2-delay equal Ψ_{h-1}. To find out such union, we make use of the shortest path algorithm as follows.

Define the cost on edge $e \in E$ as $L_e(O(t_{h-1}), 2)$. Build a shortest path tree T rooted at s and find out the distance labels $c(v)$ for all $v \in V$. For each edge $e = (u, v) \notin T$, if e is on any s-d path in the original graph G and $c(u) + L_e(O(t_{h-1}), 2) = c(v)$, add e into T. Let the resultant set of edges be E'.

Lemma 23. *Suppose that the given network G is well-designed. Then all paths in $G(V, E')$ have the same 2-delay Ψ_{h-1} in $O(t_{h-1})$ and these paths form a superset of $\mathcal{P}_h$.*

Proof. This follows from Lemma 12(iii) and the shortest path algorithm. □

The following lemma states that if G is well-designed, then so is $G(V, E')$ and $\mathcal{P}_h$ will be the last set of paths used by the optimal flow in it (so we can choose an arbitrarily large t to apply convex programming). Conversely, if $G(V, E')$ is well-designed, then G is well-designed "at least up to t_h".

Lemma 24. *(i) Suppose that the given network G is well-designed. Then the optimal flow is also monotonically non-decreasing for all $t > t_{h-1}$ in the network $G(V, E')$ and it can be expressed as $O(t) = O(t_{h-1}) + \alpha^h(t - t_{h-1})$.*
(ii) Conversely, if the optimal flow in $G(V, E')$ is monotonically non-decreasing for all $t > t_{h-1}$ and it can be expressed as $O(t) = O(t_{h-1}) + \alpha^h(t - t_{h-1})$ for some vector α^h, then $O(t)$ is also optimal in G for $t \in (t_{h-1}, t_h]$ for some $t_h > t_{h-1}$.

Proof. To prove (i), because of Lemma 23, it needs to be shown that there is no s-d path q in $G(V, E')$ and $q \notin \mathcal{P}_h$, so that when $t > t_{h-1}$, $L_q(O(t), 2)) < L_p(O(t), 2)$ for some $p \in \mathcal{P}_h$.

If $q \notin \mathcal{P}_h$, then given $\epsilon > 0$ so that when $t = t_{h-1} + \epsilon \leq t_h$, in the original graph G,

$$L_q(O(t), 2) = \Psi_{h-1} + \epsilon \sum_{e \in q} a_e \alpha_e^h \geq L_p(O(t), 2) = \Psi_{h-1} + \epsilon \sum_{e \in p} a_e \alpha_e^h, \forall p \in \mathcal{P}_h.$$

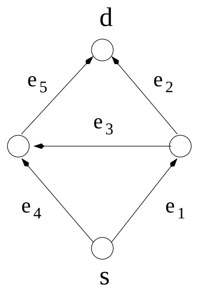
So $\sum_{e \in q} a_e \alpha_e^h \geq \sum_{e \in p} a_e \alpha_e^h$, implying that $L_q(O(t), 2) \geq L_p(O(t), 2)$ for all $t > t_{h-1}$ in $G(V, E')$.

For (ii), we need to argue that $O(t) = O(t_{h-1}) + \alpha^h(t - t_{h-1})$ is also an optimal flow in the original graph G for $t \in (t_{h-1}, t_h]$.

First note that if path $q \notin \mathcal{P}_h$ but q is in $G(V, E')$, then by the same argument as above, it is easy to show that $t_{h-1} < t \leq t_h$, $L_q(O(t), 2) \geq L_p(O(t), 2)$ in the original graph G.

For those paths q not contained in $G(V, E')$, in G, $L_q(O(t_{h-1}), 2) > \Psi_{h-1} = L_{p \in \mathcal{P}_h}(O(t_{h-1}), 2)$ If $\sum_{e \in q} a_e \alpha_e^h < \sum_{e \in p \in \mathcal{P}_h} a_e \alpha_e^h$, there exists a value $t > t_{h-1}$ so that $L_q(O(t), 2) = L_p(O(t), 2)$. Choose the smallest such value t and we claim that it is t_h. (In the case that there is no such path q, let t_h be infinity). To see this, note that by this choice of t, we have that for each path $q \notin \mathcal{P}_h$ and no matter q is contained in $G(V, E')$ or not, $L_q(O(t), 2) \geq L_p(O(t), 2)$ in $(t_{h-1}, t_h]$. Hence, by Lemmas 8 and 9, when $t \in (t_{h-1}, t_h]$, $O(t)$ is the optimal flow in the original graph G. □

We emphasize that $\mathcal{P}_h$ can be a *strict* subset of all s-d paths in $G(V, E')$. The network in the figure below is a simple example. It is well-designed and when $t > 0$, the optimal flow $O(t)$ uses the two paths $e_1 - e_2$ and $e_4 - e_5$ evenly, though when $t = 0$, the path $e_1 - e_3 - e_5$ has the same 2-delay $\Psi_0 = 0$ as the other two paths.



Edge	Delay function
e_1	$2x$
e_2	x
e_3	$0.1x$
e_4	x
e_5	$2x$

Fig. 5. A counter-example to show that $\mathcal{P}_h$ can be a strict subset of all s-d paths in $G(V, E')$.

We now use convex programming to find out $\mathcal{P}_h$. Consider the following convex program.

$$\min \sum_{e\in E'} f_e(a_e f_e + b_e)$$

$$\sum_{v:(u,v)\in E'} f_{uv} - \sum_{v:(v,u)\in E'} f_{vu} = 0 \quad \forall u \notin \{s,d\} \tag{21}$$

$$\sum_{v:(s,v)\in E'} f_{sv} = t \tag{22}$$

$$f_e \geq 0 \quad \forall e \in E' \tag{23}$$

By Lemma 24(i), if G is well-designed, then we can choose an arbitrary $t > t_{h-1}$ and the optimal solution $O(t)$ for the above convex program gives the set of paths in $\mathcal{P}_h$. The integral optimal solution of this program can be found using min-cost algorithm [1, Chap. 14.5]. However, the integral optimal is only an approximate; and if the exact solution $O(t)$ on edge e has $0 < O_e(t) < 1$, we may miss this edge in the integral solution. To avoid this, we need a suitably large t to guarantee that if edge e is used by $O(t)$, then $O_e(t) \geq 1$.

Lemma 25. *Suppose that G is well-designed. Then there exists a value M_h so that α_e^h is a multiple of $1/M_h$ for each $e \in \mathcal{P}_h$, and $M_h \leq m!a_{\max}^m$.*

The proof of this lemma is deferred to the next section. By this lemma and Lemma 24(i), if G is well-designed and we choose t to be $\lceil t_{h-1} + m!a_{\max}^m \rceil$, then $O_e(t) \geq 1$ for all $e \in E'$. Thus the returned approximate integral solution correctly returns the set of edges that are used in $\mathcal{P}_h$.

To use the min-cost flow technique in [1, Chap. 14.5], one needs to set a capacity constraint on each edge $e \in E'$. In our case, we can let the flow value t to be the capacity of all edges. Then the complexity of finding an integral solution is $O((m + n\log n)\log t)$, where the expression $(m + n\log n)$ comes from the shortest path calculation and we will discuss how large t can be in Section C.4.

C.2 Calculating the exact values in α^h

Suppose that $\mathcal{P}_h$ is known. Let $G_h = (V_h, E_h)$ be the subgraph of $G(V, E')$, where E_h consists of all edges used in $\mathcal{P}_h$ while V_h are vertices on any s-d path using only edges in E_h.

If the network G is well-designed, by Lemma 12(ii), α^h itself is a flow of value 1 and along every two s-d paths p,q in G_h, $\sum_{e\in p} a_e\alpha_e^h = \sum_{e\in q} a_e\alpha_e^h$. Also, if $e \notin E_h$, α_e^h must be 0. For simplicity, in the following discussion, we assume that α^h is a vector indexed by edges in E_h, instead of E.

By the above discussion, if the network is well-designed, the unique solution to (24)-(26) gives α^h.

$$\sum_{e\in p} a_e\alpha_e - \sum_{e\in q} a_e\alpha_e = 0 \quad \forall p,\, q \text{ that are } s\text{-}d \text{ paths in } G_h \tag{24}$$

$$\sum_{v:(u,v)\in E_h} \alpha_{uv} - \sum_{v:(v,u)\in E_h} \alpha_{vu} = 0 \quad \forall u \notin \{s,d\} \tag{25}$$

$$\sum_{v:(s,v)\in E_h} \alpha_{sv} = 1 \tag{26}$$

However, the number of equations in (24) can be exponential. Below we use the idea of *cycle basis* [9, 17] to reduce the size of the linear system.

Let the collection of all undirected cycles in the underlying graph $G_h = (V_h, E_h)$ be $\mathcal{C}$. Orient each cycle $C \in \mathcal{C}$ arbitrarily. If in G_h, e agrees with the orientation of C, we write $e \hookrightarrow C$, otherwise, we write $e \not\hookrightarrow C$. Observe that for every two s-d simple paths p and q in G_h, $p \cap q$ is composed of a subset of undirected cycles in $\mathcal{C}$. This follows from the fact that an optimal flow cannot send its flow along a directed cycle. Thus, (24) for such pair of paths can be written as $\sum_{C \in (p\cap q)\cap\mathcal{C}} \sum_{e\hookrightarrow C} a_e\alpha_e - \sum_{e\not\hookrightarrow C} a_e\alpha_e = 0$. By linear algebra, (24) can be expressed as

$$\sum_{e\hookrightarrow C} a_e\alpha_e - \sum_{e\not\hookrightarrow C} a_e\alpha_e = 0, \quad \forall C \in \mathcal{C}.$$

The above set of equations can still be of exponential size. Now define $\gamma_e = a_e\alpha_e$ for all $e \in E_h$. The the above set of equations can be re-written as

$$\sum_{e\hookrightarrow C} \gamma_e - \sum_{e\not\hookrightarrow C} \gamma_e = 0 \quad \forall C \in \mathcal{C},$$

and we can write the above set of equations in the matrix form $A\gamma = 0$, where A is a $\{-1, 0, 1\}$ matrix and its row space is exactly the cycle space of the graph $G_h = (V_h, E_h)$. Thus, any cycle basis would guarantee that every row can be generated by a linear combination of vectors in it. It is well-known that a cycle basis can be found in $O(m^2)$ time and it has $O(m)$ vectors [9]. Let a cycle basis in $G_h = (V_k, E_k)$ be $\mathcal{CB}$ and consider the following system.

$$\sum_{e\hookrightarrow C} a_e\alpha_e - \sum_{e\not\hookrightarrow C} a_e\alpha_e = 0 \quad \forall C \in \mathcal{CB} \tag{27}$$

Lemma 26. *If the network G is well-designed, then α^h is the unique solution to the system (25)-(27) and the size of this system is bounded by $O(m + n)$.*

Proof. It follows from the preceding discussion. □

By the above lemma, if the system (25)-(27) does not have a unique and non-negative solution, we can report the network is not well-designed. By Lemma 24(ii), we can also test whether the optimal flow is monotonically non-decreasing when $t \in (t_{h-1}, t_h]$.

We now give the proof of Lemma 25 that has been promised earlier.

Proof of Lemma 25. We write the system (25)-(27) in the matrix form $A\alpha = b$. By Kramer's rule, the solution to (25)-(27) can be expressed as $\frac{\mathrm{Det}(A_e)}{\mathrm{Det}(A)}$, where A_e is derived from A by replacing the column corresponding to edge e with vector b. Since $|\mathrm{Det}(A)| \le m!a_{\max}^m$ and $\mathrm{Det}(A_e)$ is integer, we prove the lemma. □

C.3 Discovering Ψ_h and t_h

Suppose that α^h is known and $O(t)$ is monotonically non-decreasing in $(t_{h-1}, t_h]$. We can first check whether $\mathcal{P}_h$ is the last set of paths used by the optimal flow by the following lemma.

Lemma 27. *$\mathcal{P}_h$ is the last set of paths used by the optimal flow in the original graph G if and only if for all paths $q \in \mathcal{P}\backslash\mathcal{P}_h$, $\sum_{e\in p} a_e\alpha_e^h \le \sum_{e\in q} a_e\alpha_e^h$ for some $p \in \mathcal{P}_1$.*

Proof. ($\rightarrow$) If $\mathcal{P}_h$ is the last set of paths and $\sum_{e\in p} a_e\alpha_e^h > \sum_{e\in q} a_e\alpha_e^h$ for some path $q \in \mathcal{P}\backslash\mathcal{P}_h$, then there exists a large enough $t > t_{h-1}$ such that $L_q(O(t), 2) < L_p(O(t), 2)$, contradicting Lemma 8.

($\leftarrow$) We claim that $O(t) = O(t_{h-1}) + \alpha^h(t - t_{h-1})$ for all $t > t_{h-1}$ in G. If the claim holds, then Lemma 8 implies that $\mathcal{P}_h$ is the last set of paths. To see this claim, note that $\sum_{e\in p} a_e\alpha_e^h \leq \sum_{e\in q} a_e\alpha_e^h$ for all $q \in \mathcal{P}\backslash\mathcal{P}_h$ and $L_q(O(t_{h-1}), 2) \geq L_p(O(t_{h-1}), 2)$ in G. Thus, all paths in $\mathcal{P}_h$ always have the smallest 2-delay for all $t > t_{h-1}$. □

From now on we assume that $\mathcal{P}_h$ is not the last set of paths used by the optimal flow in G.

By Lemma 12(iii), when t is increased to t_h, there exists a path $q \in \mathcal{P}_{h+1}\backslash\mathcal{P}_h$ so that $L_q(O(t_h), 2) = L_p(O(t_h), 2) = \Psi_h$, for some path $p \in \mathcal{P}_1$. The difficulty lies in the fact that there may exist an exponential number of paths in $\mathcal{P}\backslash\mathcal{P}_h$. The next lemma implies that it suffices to keep track of a polynomial number of 2-delays in these paths.

Lemma 28. *Suppose that G is well-designed and $\mathcal{P}_h$ not the last set of paths used by the optimal flow. Then there exists a path $q \in \mathcal{P}_{h+1}\backslash\mathcal{P}_h$ so that*

(i) $q = q_1 \cup q_2 \cup q_3$, where q_1 links s and u, using only edges in E_h, q_2 links u and v, using only edges in $E\backslash E_h$, and q_3 links v and d, using only edges in E_h.
(ii) Let the cost of all edges in $E\backslash E_h$ be b_e. Then q_2 is a shortest path from u to v in the subgraph $G(V, E\backslash E_h)$;
(iii) Let q_v be a path from s to v using only edges in E_h. Then

$$\sum_{e\in q_1} L_e(O(t_{h-1}), 2) + 2(t_h - t_{h-1})a_e\alpha_e^h + \sum_{e\in q_2} b_e = \sum_{e\in q_v} L_e(O(t_{h-1}), 2) + 2(t_h - t_{h-1})a_e\alpha_e^h. \quad (28)$$

Proof. Let $\tilde{q}$ be a path in $\mathcal{P}_{h+1}\backslash\mathcal{P}_h$. Then $\tilde{q}$ uses at least an edge $E\backslash E_h$. Let the first subpath of $\tilde{q}$ using edges in $E\backslash E_h$ be q_2 and u and v its starting and ending vertices. Then $v \in V_h$ and the 2-delay of the path from v to d using only edges in E_h is the same as the 2-delay of the subpath of q from v to d when $t = t_h$. (This holds because both paths are subpaths of some paths in $\mathcal{P}_{h+1}$). Now let q_1 be a path from s to u and q_3 a path from v to d, both using only edges in E_h. (i) then follows.

(ii) follows from Lemma 12(iii) and (iii) from Lemma 12(i). □

Lemma 28 suggests a straightforward algorithm to discover t_h. Pick every two vertices u and v in V_h. Calculate the amount $t_h - t_{h-1}$ based on the formula in (28). The pair that gives the smallest positive amount of $t_h - t_{h-1}$ is the right pair and yields the correct value of t_h. Finally, Ψ_h can be calculated based on t_h using Lemma 12(iiia).

C.4 The Complexity of the Algorithm and the Bit Size of All Values

All the computations discussed, except the convex programming in Section C.1, can be easily done in polynomial time.

For the convex programming we use, recall that we need to use a large enough value $t = \lceil t_{h-1} + m!a_{\max}^m \rceil$ as a capacity on each edge and the complexity of would be $O((m + n\log n)\log t)$. Below we give a (rather loose) bound of t_h. Let

$$\Delta_i = \max_{v\in V} \max_{\forall p,q \text{ from } s \text{ to } v} |L_p(O(t_i), 2) - L_q(O(t_i), 2)|.$$

Note that $\Delta_0 \leq mb_{\max}$. Now observe that (28) can be re-written as

$$\begin{aligned} t_h - t_{h-1} &= \frac{\sum_{e\in q_1} L_e(O(t_{h-1}),2) + \sum_{e\in q_2} b_e - \sum_{e\in q_v} L_e(O(t_{h-1}),2)}{2\sum_{e\in q_v} a_e\alpha_e^h - \sum_{e\in q_1} a_e\alpha_e^h} \\ &\leq M_h(\sum_{e\in q_1\cup q_2} L_e(O(t_{h-1}),2) - \sum_{e\in q_v} L_e(O(t_{h-1}),2)) \\ &\leq m!a_{\max}^m \Delta_h, \end{aligned}$$

where the inequalities are due to Lemma 25. Furthermore, we have

$$\Delta_h \leq \Delta_{h-1} + 2ma_{\max}(t_h - t_{h-1}) \leq \Delta_{h-1}(1 + 2ma_{\max}m!a_{\max}^m),$$

where the first inequality follows from the fact that in the extreme case that, given two directed paths linking s to some vertex v, one path does not increase its 2-delay when t goes from t_{h-1} to t_h while the other has m edges and $\alpha_e^h = 1$ for all such edges. Now by simple calculation, we have

$$t_h \leq m!a_{\max}^m(\sum_{i=0}^{h-1}\Delta_i) = \Delta_0 m!a_{\max}^m(\sum_{i=0}^{h-1}(1+2ma_{\max}m!a_{\max}^m)^i) < mb_{\max}m!a_{\max}^m(1+2ma_{\max}m!a_{\max}^m)^h.$$

As $h \leq m$. We conclude that the value $\log t = \log\lceil t_{h-1} + m!a_{\max}^m\rceil$ is polynomially bounded.

Finally, we show that the bit sizes of all values used in our computation are polynomially bounded.

Lemma 29. *Suppose that the network G is well-designed. Then all values in $O_e(t_h)$, can be expressed as a multiple of $\Pi_{i=1}^h M_i \leq (a_{\max}^m m!)^h$.*

Proof. This can be proved by induction and equation (28). $\square$

By the above lemma, the bound on t_h, and Lemma 12(iiia), we can conclude that the bit sizes of the ϕ-delay thresholds Ψ_h are also polynomially bounded.