Maximum flow under proportional delay constraint

Pierre Bonami, Dorian Mazauric, Yann Vaxès

Maximum flow under proportional delay constraint

Pierre Bonami[a,b], Dorian Mazauric[a,c], Yann Vaxès[a]

[a] *Aix-Marseille Université, CNRS, LIF UMR 7279, 13000, Marseille, France.*
[b] *IBM ILOG CPLEX, Madrid, Spain.*
[c] *Inria Sophia Antipolis - Méditerranée*

Abstract

Given a network and a set of source destination pairs (connections), we consider the problem of maximizing the sum of the flow under proportional delay constraints. In this paper, the delay for crossing a link is proportional to the total flow crossing this link. If a connection supports non-zero flow, then the sum of the delays along any path corresponding to that connection must be lower than a given bound. The constraints of delay are on-off constraints because if a connection carries zero flow, then there is no constraint for that connection. The difficulty of the problem comes from the choice of the connections supporting non-zero flow. We first prove a general approximation ratio using linear programming for a variant of the problem. We then prove a linear time 2-approximation algorithm when the network is a path. We finally show a Polynomial Time Approximation Scheme when the graph of intersections of the paths has bounded treewidth.

Keywords: Maximum flow, on-off delay constraints, polynomial time approximation scheme (PTAS), dynamic programming, bounded treewidth, linear programming.

1. Introduction

The multi-commodity flow problem is a classical network flow problem with multiple commodities (flow demands) between different source and sink nodes. This problem has been widely studied in the literature. Given a network, a set of capacities on edges, and a set of demands (commodities), the problem consists in finding a network flow satisfying all the demands and respecting capacities and flow conservation constraints. The integer version of the problem is NP-complete [6], even for only two commodities and unit capacities (making the problem Strongly NP-complete in this case). In that version, the problem consists of producing an integer flow satisfying all demands and respecting the previously mentioned constraints. However, if fractional flows are allowed, the problem can be solved in polynomial time through linear programming [1, 4, 8].

Network operators must satisfy some Quality of Service requirements for their clients. One of the most important parameters in telecommunications networks is the end-to-end delay of a unit of flow between a source node and a destination node. This requirement is not taken into account in the multi-commodity flow problem. The delay through a link depends on the amount of flow supported by this link; classically it is modeled by a convex function. The end-to-end delay for a demand and a path associated with this demand, is the sum of the delays through all links of this path. Some papers focus on minimizing the mean end-to-end delay. This problem consists of minimizing a convex function under linear constraints [3, 9] and can be solved using semidefinite programming [12]. Other papers focus on finding a multi-commodity flow that satisfies the demands and that minimizes the maximum end-to-end delay [5].

As the authors of [2], we think that a more realistic problem consists of adding strict end-to-end delay constraints for all connections. Indeed, in communication networks, there are multiple classes of services, and for each of them, it is crucial to respect a certain level of Quality of Service, i.e. respecting a threshold for the end-to-end delay. It is why we study a multicommodity flow problem in which all demands have to respect an end-to-end delay constraint.

In this paper, we focus on multicommodity network flow problems in which each edge possesses a proportional latency function that describes the common delay experienced by the flow on that edge as a function of the flow rate. These problems model congestion effects that appear in a variety of applications such as communication networks, vehicular traffic, supply chain management, or evacuation planning. In such applications, the latency function need

not be proportional to the flow rate. We investigate the problem of maximizing the sum of the flow under proportional delay constraints. We allow fractional flows for all connections. As mentioned before, we have an end-to-end delay constraint for each demand. But, if a path associated with one connection carries zero flow, then the constraint is not active. These on-off constraints make the problem more difficult to solve than just a simple linear program [7].

In [2], it is proved that this problem is NP-hard even if there are only two paths per connection. In their proof, delay and capacity constraints are used. The complexity of the problem is unknown when only considering delay constraints. We formally present our problem in Section 1.1. We then describe a simple example in Section 1.2 and preliminary results in Section 1.3. We present our contributions in Section 1.4.

1.1. Problem formulation and model

Let $G = (V,E)$ be a connected undirected graph (that represents a network) with a coefficient α_e for each edge $e \in E$. Let $\{(s_1,t_1),\ldots,(s_m,t_m)\}$ be a set of m source destination pairs (connections). Let $\mathcal{P} = \{P_1,\ldots,P_m\}$ be a set of m paths in G. For all $i = 1,\ldots,m$, P_i is the path, between the source s_i and the destination t_i, allocated to connection (s_i,t_i). Without loss of generality, we suppose that there is a unique path for each connection. Indeed, we can have several connections involving the same pair of nodes but with different associated paths. We denote by x_i the flow through the path P_i for all $i = 1,\ldots,m$. We suppose that the delay τ_e for crossing an edge $e \in E$ is proportional to the total flow $\sum_{i:e\in E(P_i)} x_i$ crossing the edge e, i.e. $\tau_e = \alpha_e \sum_{i:e\in E(P_i)} x_i$. Let $\lambda > 0$. For all $i = 1,\ldots,m$, we require for path P_i that, if $x_i > 0$, then the end-to-end delay $\sum_{e\in P_i} \tau_e$ is at most λ. By scaling the coefficients of the α_e, we can make this bound λ equal to one. Using the notation $\beta_{i,j} := \sum_{e\in E(P_i)\cap E(P_j)} \alpha_e$, these constraints can be written as follows: $\sum_{i=1}^{m} \beta_{i,j} x_i \leq 1$. A multicommodity flow $x = (x_1,\ldots,x_m)$ is *admissible* if the following latency requirement is satisfied: for all $j = 1,\ldots,m$ such that $x_j > 0$, then $\sum_{e\in E(P_j)} \tau_e \leq 1$. Furthermore, for all $i = 1,\ldots,m$, if $x_i > 0$, then we say that connection (s_i,t_i) and path P_i are *active* (*inactive* otherwise). The Maximum Flow under Delay Constraint problem (FDC) consists in finding among the solutions satisfying these constraints a solution of maximum value $\sum_{i=1}^{m} x_i$, i.e.:

$$\left\{\begin{array}{lll} \text{Max} & \sum_{i=1}^{m} x_i & \\ & \sum_{i=1}^{m} \beta_{i,j} x_i \leq 1 & j = 1,\ldots,m \quad x_j > 0 \\ & x_i \geq 0 & i = 1,\ldots,m. \end{array}\right.$$

The hardness of this problem comes from the choice of the active paths. Indeed, if we are given a set of active paths $\mathcal{P}^* \subseteq \mathcal{P}$ in some optimal solution, then the problem becomes polynomial since it reduces to solving the linear program $LP(\mathcal{P}^*)$. Without loss of generality let $\mathcal{P}^* = \{P_1,\ldots,P_{m^*}\}$ with $m^* \leq m$.

$$LP(\mathcal{P}^*)\left\{\begin{array}{lll} \text{Max} & \sum_{i=1}^{m^*} x_i & \\ & \sum_{i=1}^{m^*} \beta_{i,j} x_i \leq 1 & j = 1,\ldots,m^* \\ & x_i \geq 0 & i = 1,\ldots,m^*. \end{array}\right.$$

The dual of this linear program is:

$$DLP(\mathcal{P}^*)\left\{\begin{array}{lll} \text{Min} & \sum_{j=1}^{m^*} y_j & \\ & \sum_{j=1}^{m^*} \beta_{i,j} y_j \geq 1 & i = 1,\ldots,m^* \\ & y_j \geq 0 & j = 1,\ldots,m^*. \end{array}\right.$$

Note that $LP(\mathcal{P}^*)$ and its dual $DLP(\mathcal{P}^*)$ differ only by the sense of inequalities and the direction of optimization. In particular, if the system of equations $\sum_{i=1}^{m^*} \beta_{i,j} x_i = 1, j = 1,\ldots,m^*$, has a solution then this solution is optimal for the primal and for the dual since it satisfies both primal and dual complementary slackness conditions.

1.2. Example

Consider the path $G = (V,E)$ described in Figure 1 and the three source destination pairs (s_1,t_1), (s_2,t_2), and (s_3,t_3). We set $\alpha_e = 1$ for all $e \in E$. FDC consists of solving:

$$\left\{\begin{array}{ll} \text{Max} & x_1 + x_2 + x_3 \\ & x_1(3x_1 + 2x_2) \leq x_1 \\ & x_2(2x_1 + 4x_2 + x_3) \leq x_2 \\ & x_3(x_2 + 2x_3) \leq x_3 \\ & x_1, x_2, x_3 \geq 0 \end{array}\right.$$

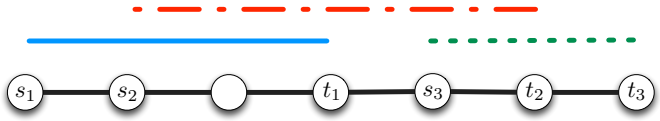

Figure 1: Instance of FDC: path $G = (V,E)$ with $\alpha_e = 1$ for all $e \in E$, and three source destination pairs (s_1,t_1), (s_2,t_2), and (s_3,t_3).

The first constraint means that, if the path P_1 is active, then the end-to-end delay for connection (s_1,t_1) must be at most one. The delay for crossing the leftmost edge of P_1 is x_1. The delay is $x_1 + x_2$ for each of the two other edges. Thus, if $x_1 > 0$, then $3x_1 + 2x_2 \leq 1$. Note that if $x_1 = 0$, the first constraint is always satisfied. The second and the third inequalities are constructed similarly. One can observe that $x^* = (1/3, 0, 1/2)$ is the optimal solution. Note that connection (s_2,t_2) is inactive, i.e. $x_2 = 0$, and so the second constraint is satisfied. That explains why the end-to-end delay $2x_1 + 4x_2 + x_3 = 7/6 > 1$ for connection (s_2,t_2) is not satisfied.

1.3. Preliminaries

We first prove in Lemma 1 that we can consider instances such that the path of a connection is not included in a path of another connection (but the paths may intersect).

Lemma 1. *Let I be an instance of FDC. If there are two different connections P_j and P_k such that $E(P_k) \subseteq E(P_j)$, then there is an optimal solution x^* such that $x^*_j = 0$.*

Proof. Consider any admissible flow x for I such that $x_j > 0$. We construct another admissible flow x' from x in which we only change the amount of flow for paths P_j and P_k. More precisely, we set $x'_k := x_k + x_j$, $x'_j := 0$, and $x'_i := x_i$ for all $i \in \{1,\ldots,m\} \setminus \{j,k\}$. Clearly, the total amount of flow is unchanged, i.e. $\sum_{i=1}^{m} x_i = \sum_{i=1}^{m} x'_i$. Let τ and τ' denote the vector of delay for x and x', respectively. By construction of x', we get $\tau'_e \leq \tau_e$ for all $e \in E$ because $E(P_k) \subseteq E(P_j)$. As x is an admissible flow, then it follows that $\sum_{e \in E(P_i)} \tau'_e \leq 1$, for all $i = 1,\ldots,m$. Thus, the vector of flow x' is admissible. To conclude, if x is an optimal solution, then x' is an optimal solution such that $x'_j = 0$. □

By Lemma 1, we only consider instances such that $E(P_k) \not\subseteq E(P_j)$ for all $j,k \in \{1,\ldots,m\}$, $j \neq k$.

Let us define the graph H of intersections of the paths of an instance of FDC. The set of nodes $V(H) = \{h_1,\ldots,h_m\}$ corresponds to the set of paths $\mathcal{P} = \{P_1,\ldots,P_m\}$. For $i,j \in \{1,\ldots,m\}$, $i \neq j$, there is an edge $\{h_i,h_j\} \in E(H)$ between two nodes $h_i \in V(H)$ and $h_j \in V(H)$ if, and only if, there exists $e \in E$ such that $e \in E(P_i) \cap E(P_j)$, i.e. when P_i and P_j share at least one edge. The graph H of intersections of the paths of the instance depicted in Figure 1 is a path composed of three nodes. Without loss of generality, we assume that the graph H is connected. Indeed, if H is not connected, we can consider the maximal connected components of H as independent instances of FDC.

1.4. Contributions

The complexity of FDC is still unknown but we conjecture that it is very hard to solve even for simple classes of instances. In this context, we make the following contributions. In Section 2, we prove a general polynomial approximation algorithm and a polynomial L-approximation algorithm, where L is the size of a longest path of $\mathcal{P}$. In Section 3, we prove a linear time 2-approximation algorithm when G is a path. In Section 4, we prove a Polynomial Time Approximation Scheme (PTAS) when the graph H of intersections of the paths has bounded treewidth.

2. Approximation algorithms using linear programming

This section is dedicated to prove approximation algorithms for FDC based on a variant of the problem, called Maximum Flow under Strong Delay Constraint problem (FSDC), for which all the end-to-end delay constraints must be satisfied even for inactive connections. This problem is polynomial since it reduces to solving the linear program $LP(\mathcal{P})$, i.e. $LP(\mathcal{P}^*)$ previously described when replacing $\mathcal{P}^*$ by $\mathcal{P}$ and m^* by m. Note that an admissible solution for FSDC is an admissible solution for FDC because the end-to-end delay constraints are satisfied for all (active and inactive) connections in FSDC. Consider the example of Figure 1. One can observe that $x = (1/4, 0, 1/2)$ is an optimal

solution for this variant of the problem. Such a solution for FSDC obtained by solving $LP(\mathcal{P})$ is close to the optimal solution x^* for FDC. Indeed, $(x_1^* + x_2^* + x_3^*)/(x_1 + x_2 + x_3) = 10/9$.

We first analyze the ratio between the value of an optimal solution for FSDC and the value of an optimal solution for FDC. We then deduce some polynomial approximation algorithms for FDC based on the solution of the linear program $LP(\mathcal{P})$ for FSDC (Theorem 3 and Corollary 1). Let us first define some parameters and prove Lemma 2.

Definition 1. For all $k = 1, \ldots, m$, let $S_k^* \subseteq \{1, \ldots, m\} \setminus \{k\}$ be a maximum cardinality set of indices such that for all $i \in S_k^*$, there exists $e \in E(P_k) \cap E(P_i)$ such that for all $j \in S_k^* \setminus \{i\}$, $e \notin E(P_j)$. Let $|S^*| = \max_{i=1,\ldots,m} |S_i^*|$.

Intuitively, S_k^* is a maximal set of (indices of) paths that each have a "private" edge in common with P_k (i.e. an edge not shared with another path).

Lemma 2. *Let I be any instance. Let x^* be an optimal solution for FDC and let x be an optimal solution for FSDC. Then $\sum_{i=1}^m x_i \geq \sum_{i=1}^m x_i^*/|S^*|$.*

Proof. Recall that if $x_k^* > 0$, then $\sum_{e \in E(P_k)} \tau_e^* \leq 1$. We first prove that if $x_k^* = 0$, then $\sum_{e \in E(P_k)} \tau_e^* \leq |S_k^*|$. Let P_k be a path such that $x_k^* = 0$. Consider a minimal (by inclusion) set of active paths $S \subseteq \mathcal{P}$ that covers all edges of P_k supporting non-zero flow. By definition of S_k^*, $|S| \leq |S_k^*|$. Assign to each edge $e \in E(P_k)$ supporting non-zero flow, a path P_i, $i \in S$ such that $e \in P_i$. For each $i \in S$, let E_i be the set of edges assigned to the path P_i. Since $x_i^* > 0$, $\sum_{e \in E_i} \tau_e^* \leq 1$, i.e. the sum of the delays on the edges of E_i is at most 1. There are $|S|$ groups of edges, and thus $\sum_{e \in E(P_k)} \tau_e^* \leq |S| \leq |S_k^*|$, i.e. the total delays on the edges of P_k is at most $|S_k^*|$.

For all $k \in \{1, \ldots, m\}$ such that $x_k^* = 0$, then $\sum_{e \in E(P_k)} \tau_e^* \leq |S_k^*|$. Consider the solution x' obtained by dividing all the flows of x^* by $|S^*|$. Formally, set $x_i' := x_i^*/|S^*|$ for all $i = 1, \ldots, m$. By construction, $\sum_{i=1}^m x_i' = \sum_{i=1}^m x_i^*/|S^*|$. By previous claims, x' is an admissible solution for FSDC. Finally, an optimal solution x for FSDC satisfies $\sum_{i=1}^m x_i \geq \sum_{i=1}^m x_i'$. □

Because FSDC is polynomial and by Lemma 2, we deduce the following theorem:

Theorem 3. *There exists a polynomial $|S^*|$-approximation algorithm for FDC.*

Let $k \in \{1, \ldots, m\}$ such that $|E(P_i)| = L$, where $L = \max_{i=1,\ldots,m} |E(P_i)|$ be the size of a longest path. By definition, S_k^* is a maximal set of paths for which there exists at least one edge that is not shared with another path (of S_k^*). We get that $|S^*| \leq L$. Indeed, the number of "private" edges cannot be greater than the number of edges of the path. We deduce the following corollary of Theorem 3:

Corollary 1. *There exists a polynomial L-approximation algorithm for FDC, where $L = \max_{i=1,\ldots,m} |E(P_i)|$.*

Finally, when the graph G is a path, one can observe that $|S^*| \leq 2$. Indeed, by Lemma 1, we assume that any path (of a connection) is not included in another path, and so S_k^* is composed of at most two paths ("located" in both extremities) for every $k \in \{1, \ldots, m\}$. Thus, there is a polynomial 2-approximation algorithm. For such class of instances, we prove in Section 3 a stronger result (in terms of complexity), namely a linear 2-approximation algorithm for FDC.

3. Linear time 2-approximation algorithm when G is a path

This section is dedicating to proving another polynomial 2-approximation algorithm when the graph is a path. The interest of this algorithm is its linear time complexity. A solution x of FDC is said to be *independent* if its support $\{P_k : x_k > 0\}$ is a family of edge-disjoint paths. In other words, x is independent if the set of vertices $\{h_k : x_k > 0\} \subset V(H)$ forms an independent set of H. Recall that the graph H is the graph of intersections of the paths. We prove in Lemma 4 that a best independent solution has a value which is at least half the total flow of an optimal solution if G is a path. Note that $x = (1/3, 0, 1/2)$ is the best independent solution for the instance depicted in Figure 1 (that coincides in that case to the optimal solution of FDC). In the following, we consider that H is node-weighted. For all $i = 1, \ldots, m$, the weight w_{h_i} (or simply denoted w_i) of the node $h_i \in V(H)$ represents the maximum amount of flow that can support the path P_i if the paths sharing an edge with P_i do not carry any flow. Formally, $w_i := 1/\sum_{e \in E(P_i)} \alpha_e$.

Lemma 4. *Let I be any instance and let G be any path. Let x^* be an optimal solution and let x be a best independent solution for FDC. Then $\sum_{i=1}^{m} x_i \geq \frac{1}{2}\sum_{i=1}^{m} x_i^*$. There exists a linear time 2-approximation algorithm for FDC when G is a path.*

Proof. Consider an optimal solution x^* of an instance I and the instance I' obtained from I by removing all source-destination pairs that do not belong to the support of x^*. Since instances I and I' have the same optimum and an independent solution for I' is also an independent solution for I, it suffices to prove the claim of lemma 4 for I'. For the instance I', there is an optimal solution in which all delay constraints are active. This solution is therefore an optimal solution of the linear program $LP(\mathcal{P}^*)$ where $\mathcal{P}^* := \{P_i : x_i^* > 0\}$ and its value is at most the value of any feasible dual solution y of $DLP(\mathcal{P}^*)$. Without loss of generality let $\mathcal{P}^* = \{P_1, \ldots, P_{m^*}\}$ with $m^* \leq m$. Let $x_{I'}$ be an optimal independent solution of I', we will show that there exists a feasible solution $y_{I'}$ of $DLP(\mathcal{P}^*)$ whose value is at most twice the value of $x_{I'}$.

Since G is a path, the graph H of intersections of the paths is an interval graph. Let $\mathcal{C}$ be the set of maximal cliques of H. Since H is perfect, from the theory of perfect graphs (see for instance Chapter 65 of [11]), the caracteristic vector z of a maximum weighted independent set is an optimal solution of the following linear program $LPQ(\mathcal{P}^*)$:

$$(LPQ(\mathcal{P}^*)) \begin{cases} \text{Max} & \sum_{i=1}^{m^*} w_i z_i \\ & \sum_{i:h_i \in C} z_i \leq 1 \quad \forall C \in \mathcal{C} \\ & z_i \geq 0 \qquad\qquad i = 1, \ldots, m^*. \end{cases}$$

Consider the dual $(DLPQ(\mathcal{P}^*))$ of this linear program :

$$(DLPQ(\mathcal{P}^*)) \begin{cases} \text{Min} & \sum_{C \in \mathcal{C}} t_C \\ & \sum_{C:h_i \in C} t_C \geq w_i \quad i = 1, \ldots, m^* \\ & t_C \geq 0 \qquad\qquad \forall C \in \mathcal{C}. \end{cases}$$

Let t be an optimal solution of $DLPQ(\mathcal{P}^*)$. The following algorithm computes from t, a feasible solution y of $DLP(\mathcal{P}^*)$. Let $C \in \mathcal{C}$, we denote $l(C)$ (resp. $r(C)$) the index of the leftmost (resp. rightmost) path such that its corresponding node in H belongs to the clique C. For all $i = 1, \ldots, m^*$ do $y_i := 0$, and for all $C \in \mathcal{C}$ do $y_{l(C)} := y_{l(C)} + t_C$ and $y_{r(C)} := y_{r(C)} + t_C$. Since the value of each variable t_C for $C \in \mathcal{C}$ is added to exactly two variables y_j, $j = 1, \ldots, m^*$, the value of y is at most twice the value of t.

For all $i = 1, \ldots, m^*$, if $e \in E(P_i)$ the following inequality holds: $\sum_{j:e \in E(P_j)} y_j \geq \sum_{C:h_i \in C} t_C \geq w_i$. Since t is a feasible solution of $DLPQ(\mathcal{P}^*)$, the second inequality is trivially true. In order to verify the first inequality, note that if the node h_i corresponding to the path P_i belongs to the clique C then $e \in E(P_i)$ implies that $e \in E(P_{l(C)})$ or $e \in E(P_{r(C)})$. Thus, if a clique C contributes t_C for the second sum then $y_{l(C)}$ or $y_{r(C)}$ contribubes t_C for the first one. We conclude that, for all paths $i = 1, \ldots, m^*$, $\sum_{e \in E(P_i)} \alpha_e (\sum_{j:e \in E(P_j)} y_j) \geq \sum_{e \in E(P_i)} \alpha_e w_i = w_i \sum_{e \in E(P_i)} \alpha_e = 1$.

This shows that y is a feasible solution of $DLP(\mathcal{P}^*)$. By the definition of y, its value is at most twice the weight of a maximum weight stable set S of H. The independent solution corresponding to S is thus a 2-approximation for the instance I'.

To conclude the proof, computing an optimal independent solution when G is a path reduces to compute a maximum weighted independent set of H and so we get the linear time complexity because H is an interval graph. □

Note that we cannot find a similar result when the graph G is a tree. Indeed, a best independent solution may be arbitrarily far from an optimal solution for this class of graphs. As an example, consider the instance depicted in Figure 2. The graph G is a tree composed of m edge disjoint paths between v and the leaves t_i, $1 \leq i \leq m$, each composed of m edges, and there is an edge between s and v. There are $m \geq 1$ source destination pairs $(s,t_1), \ldots, (s,t_m)$. For every $i \in \{1, \ldots, m\}$, we denote by P_i the path corresponding to connection (s,t_i). For every edge $e \in E \setminus \{s,v\}$, then $\alpha_e = 1$, and $\alpha_{s,v} = 1/m$. Since $\{s,v\} \in E(P_i)$ for every $i \in \{1, \ldots, m\}$, then any independent solution contains at most one active path and the total amount of flow is at most $m/(m^2+1)$. Furthermore, if $x_i = 1/(1+m)$ for every $i \in \{1, \ldots, m\}$, then x is an admissible solution for FDC such that the total amount of flow is $m/(1+m)$.

Conjecture 1. *Let $k \geq 1$ be any constant integer and let H^k be the node-weighted graph constructed as follows. The set of vertices is the family of all the subsets of at most k paths (connections), and there is an edge between two nodes u and v if, and only if, there is at least one edge of G that belongs to one path of u and to one path of v. The weight*

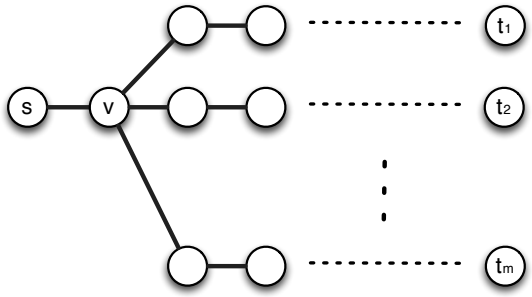

Figure 2: Instance of FDC such that a best independent solution is arbitrarily far from an optimal solution. See details in the text.

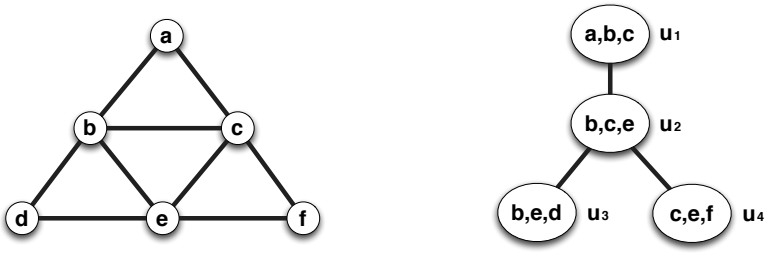

Figure 3: A graph H of intersections of the paths (left) and an optimal tree decomposition T of H with $tw(T) = 2$ (right).

of a node u is the amount of flow of an optimal solution of FDC reduced to the set of paths corresponding to u. We conjecture that computing a maximum weighted independent set of H^k gives a $((k+1)/k)$-approximation algorithm when G is a path. Note that $k = 1$ corresponds to the polynomial 2-approximation algorithm proved in Lemma 4.

4. Polynomial Time Approximation Scheme for some classes

This section is dedicated to prove a Polynomial Time Approximation Scheme (PTAS) for FDC when the graph H of intersections of the paths has bounded treewidth.

We first define a variant of the problem, called Maximum Discrete Flow under Delay Constraint problem (DFDC), as follows. The only change is that the possible amount of flow for all connections is taken among a finite set of non negative values containing zero. More formally, given a finite set X of non negative values containing zero, $x_i \in X$ for all $i = 1,\ldots,m$. Consider the instance described in Figure 1 with $X = \{0,1/3,2/3,1\}$. One can observe that $x = (1/3,0,1/3)$ is an optimal solution for this variant of the problem.

In this section, we prove an exact dynamic programming algorithm for DFDC based on a tree decomposition of H (Lemma 5). We then deduce a PTAS for FDC when the graph H of intersections of the paths has bounded treewidth (Theorem 6).

A tree decomposition T of H, with set of nodes $V(T) = \{Y_1,\ldots,Y_n\}$, where each Y_i is a subset of $V(H)$, satisfies the following properties. i) The union $\cup_{i=1}^{n} Y_i$ of all sets Y_i equals $V(H)$. ii) If Y_i and Y_j both contain a vertex $h \in V(H)$, then all nodes Y_k of T in the (unique) path between Y_i and Y_j contain h as well. iii) For every edge $\{h,h'\} \in E(H)$, there is a subset Y_i that contains both h and h'. The width of a decomposition is $\max_{i=1,\ldots,n} |Y_i| - 1$ and the treewidth of H is the minimum width among all its possible tree decompositions.

Before formally proving Lemma 5, we give the intuition of the dynamic programming algorithm described in it. Consider an instance of FDC for which the graph H of intersections of the paths is depicted in Figure 3 (left). The tree T depicted in Figure 3 (right) is an optimal tree decomposition of H with $tw(H) = 2$. Consider any set X of positive values containing 0. We now explain the computation of an optimal solution for DFDC based on the tree decomposition.

1) Consider u_3. The set of corresponding paths (connections) is $\{b,e,d\}$. The parent of u_3 is u_2. Paths b and e are

also in u_2 while d is not in it. For every possible admissible vector of flows for $\{b,e\}$, we compute an optimal flow for d. We obtain $O(|X|^2)$ vectors.

2) Consider u_4. The set of corresponding paths is $\{c,e,f\}$. The parent of u_4 is u_2. For every possible admissible vector of flows for $\{c,e\}$, we compute an optimal flow for f. There are $O(|X|^2)$ vectors.

3) Consider u_2. The set of corresponding paths is $\{b,c,e\}$. The parent of u_2 is u_1. Paths b and c are also in u_1 while e is not in it. For every possible admissible vector of flows for $\{b,c\}$, we aim at computing an optimal flow for $\{d,e,f\}$. To do that, for every possible admissible vector of flows for $\{b,c,e\}$, we compute an optimal flow for $\{d,f\}$ by using the computations done in 1) and 2).

4) Consider the root u_1. The set of corresponding paths is $\{a,b,c\}$. For every possible admissible vector of flows for $\{a,b,c\}$, we compute an optimal flow for $\{d,e,f\}$ by using the computation done in 3).

5) We finally compute an optimal solution for DFDC by chosing an optimal vector among the set of vectors computed in 4).

Lemma 5. *Let I be any instance and let X be a finite set of non negative values containing zero. There exists an exact $O(m|X|^{tw(H)+1})$-time complexity algorithm for DFDC where $tw(H)$ is the treewidth of H.*

Proof. Consider a tree decomposition T of H such that $tw(H) = \max_{i=1,\dots,n} |Y_i| - 1$. Let $t = tw(H)+1$. Let $r \in V(T)$ be the root (arbitrarily chosen) of the tree T. The set $N(u)$ represents the children of u for all $u \in V(T)$. Let $d_u = |N(u)|$. The subtree T_u is the connected component of T containing u when removing the edge $\{u,u'\}$ where u' is the parent of u, for all $u \in V(T) \setminus \{r\}$. Recall that the set of nodes $V(H) = \{h_1,\dots,h_m\}$ corresponds to the set of source destination pairs, and so corresponds to the set of paths $\mathcal{P} = \{P_1,\dots,P_m\}$. Let us define $P_u = \{P_i : h_i \notin u', \exists v \in V(T_u) : h_i \in v, i = 1,\dots,m\}$ and let $Q_u = \{P_i : h_i \in u', \exists v \in V(T_u) : h_i \in v, i = 1,\dots,m\}$ where u' is the parent of u in T, for all $u \in V(T) \setminus \{r\}$. We set $P_r = \{P_1,\dots,P_m\}$ and $Q_r = \emptyset$. Note that the set $P_u \cup Q_u$ represents all the paths that correspond to nodes of subtree T_u. Let $u \in V(T)$. Let $\{q_u^1, q_u^2, \dots, q_u^{|X|^{|Q_u|}}\}$ be the set of all possible vectors of flows for the set of paths Q_u. For all $i = 1,\dots,|X|^{|Q_u|}$ and for all $P \subseteq P_u \cup Q_u$, we define $f_{q_u^i}(P)$ as the total amount of flow of an optimal solution for DFDC reduced to the set of paths P when the vector of flows for paths of Q_u is q_u^i. Without loss of generality, we suppose that the vector q_u^i is admissible for all $u \in V(T)$ and for all $i = 1,\dots,|X|^{|Q_u|}$.

We aim at computing $f_{q_u^i}(Q_u \cup P_u)$ for all $u \in V(T)$ and for all $i = 1,\dots,|X|^{|Q_u|}$. As $|Q_u| \leq t$, then there are $O(|X|^t)$ vectors to compute for each u. We proceed by induction. Consider any leaf $u \in V(T)$. Then $|P_u \cup Q_u| \leq t$ by construction of T. Thus, we compute $f_{q_u^i}(Q_u \cup P_u)$ for all $i = 1,\dots,|X|^{|Q_u|}$ by enumerating all the possible vectors for paths of Q_u. This can be done in $O(|X|^{|Q_u \cup P_u|}) = O(|X|^t)$-time.

Consider a non-leaf node $u \in V(T)$. Let $N(u) = \{u_1,\dots,u_{d_u}\}$ be the set of children of u. Suppose we have computed $f_{q_{u_j}^i}(Q_{u_j} \cup P_{u_j})$ for all $j = 1,\dots,d_u$ and for all $i = 1,\dots,|X|^{|Q_{u_j}|}$. We now compute $f_{q_u^i}(Q_u \cup P_u)$ for all $i = 1,\dots,|X|^{|Q_u|}$. To do that, let $R_u = P_u \setminus \cup_{j=1}^{d_u} P_{u_j}$. Let $\{r_u^1,\dots,r_u^{|Q_u \cup R_u|}\}$ be the set of all possible vectors of flow for the set of paths R_u. We define $f_{q_u^i, r_u^{i'}}(P)$ as the total amount of flow of an optimal solution for DFDC reduced to the set of paths $P \subseteq P_u \cup Q_u$ when the vector of flows for paths of Q_u is q_u^i and when the vector of flows for paths of R_u is $r_u^{i'}$. By definition of $f_{q_u^i}(Q_u \cup P_u)$, we get that

$$f_{q_u^i}(Q_u \cup P_u) = \max_{i'=1,\dots,|X|^{|R_u|}} f_{q_u^i, r_u^{i'}}(Q_u \cup P_u).$$

By construction of T and by definition of P and Q, if a path $P \in P_{u_{j'}}$, for some $j' \in \{1,\dots,d_u\}$, then $P \notin P_{u_j}$ for all $j \in \{1,\dots,d_u\} \setminus \{j'\}$. We deduce that

$$f_{q_u^i, r_u^{i'}}(Q_u \cup P_u) = \sum_{j=1}^{d_u} f_{q_u^i, r_u^{i'}}(P_{u_j}) + f_{q_u^i, r_u^{i'}}(Q_u \cup R_u) = \sum_{j=1}^{d_u} f_{q_u^i, r_u^{i'}}(P_{u_j}) + f_{q_u^i}(Q_u) + f_{r_u^{i'}}(R_u).$$

Indeed, if two sets P and P' are disjoint, then $f_{q_{u^i}}(P \cup P') = f_{q_{u^i}}(P) + f_{q_{u^i}}(P')$. Note that some values of vectors q_u^i and $r_u^{i'}$ can be useless for the set of paths P_{u_j} but it is not necessary to introduce other notations. The values of $f_{q_u^i}(Q_u)$ and

$f_{r_u^i}(R_u)$ are directly deduced from q_u^i and $r_u^{i'}$, respectively (sum of the elements of the vector). Finally,

$$f_{q_u^i, r_u^{i'}}(P_{u_j}) = f_{q_{u_j}}(Q_{u_j} \cup P_{u_j}) - f_{q_{u_j}}(Q_{u_j}),$$

where q_{u_j} is the union of q_u^i and $r_u^{i'}$ from which we remove the elements that do not correspond to paths of $Q_{u_j} \cup P_{u_j}$. The computation of $f_{q_u^i}(Q_u \cup P_u)$ for all $i = 1, \ldots, |X|^{|Q_u|}$ can be done in $O(|X|^{|Q_u|}|X|^{|R_u|}d_u)$-time because the computation of $f_{q_u^i, r_u^{i'}}(P_{u_j})$ can be done independently for each j. Since $Q_u \cup R_u$ is the set of paths (connections) that corresponds to the nodes of H in u in the tree decomposition T of H, then $|Q_u \cup R_u| \leq t$. We deduce that such a computation can be done in $O(d_u|X|^t)$-time.

Finally, we have computed $f_{q_u^i}(Q_u \cup P_u)$ for all $i = 1, \ldots, |X|^{|Q_u|}$ and for all $u \in V(T)$. As $Q_r = \emptyset$, we have computed $f_\emptyset(Q_r \cup P_r)$ for the root r. We deduce that $f_\emptyset(Q_r \cup P_r)$ is the total amount of flow of an optimal solution for DFDC. Note that the computation of the vector of flow x of such an optimal solution is similar to the previous equations (e.g. by replacing max by argmax). The time complexity of the algorithm is $O(|X|^t \sum_{u \in V(T)} d_u) = O(|E(T)||X|^t) = O(m|X|^{tw(H)+1})$. □

Let $x_{max} = \max_{i=1,\ldots,m} 1/\sum_{e \in E(P_i)} \alpha_e$ and $x_{min} = \min_{i=1,\ldots,m} 1/\sum_{e \in E(P_i)} \alpha_e$. In the rest of the paper, we assume that $\log(x_{max}/x_{min})$ is bounded by a polynomial in the size of the instance (in other words, x_{max}/x_{min} can be exponential). We prove in Theorem 6 a PTAS for FDC when the graph H has bounded treewidth.

Theorem 6. *Let $t \geq 1$ be any constant integer. For any $\varepsilon > 0$, there is a polynomial $(1+\varepsilon)$-approximation algorithm for FDC when $tw(H) \leq t$.*

Proof. For every $\varepsilon > 0$, we prove that there exists an instance of DFDC such that the optimal solution x^ε satisfies $\sum_{i=1}^m x_i^\varepsilon \geq (\sum_{i=1}^m x_i^*)/(1+\varepsilon)$, where x^* is an optimal solution for FDC.

We first prove that there exists an independent set $IS(H)$ of H such that $|IS(H)| \geq m/(t+1)$. Recall that $m = |V(H)|$. A graph is t-degenerate if every induced subgraph has a vertex of degree at most t, and the degeneracy of a graph is the smallest t for which it is t-degenerate. As $tw(H) \leq t$ and as the degeneracy is upper-bounded by the treewidth [10], then H has degeneracy at most t. We then construct $IS(H)$ as follows: we add one node of degree at most t in $IS(H)$, we remove this node and its neighbors, and we repeat this process on the remaining graph (of degeneracy at most t) until obtaining the empty graph. By construction, $IS(H)$ is an independent set of H. Furthermore, since the number of steps of this process is at least $m/(t+1)$, we get that $|IS(H)| \geq m/(t+1)$.

Let ε' be such that $0 < \varepsilon' < \varepsilon$. Let $X_0 = \{x_{max}/(1+\varepsilon')^i : i = 0, 1, \ldots, p\}$ where p is such that $x_{max}/(1+\varepsilon')^p \leq (\varepsilon - \varepsilon')x_{min}/(t+1)$ and $X = \{0\} \cup X_0 \cup \{x_{min}\}$. Thus, $(1+\varepsilon')^p \geq x_{max}(t+1)/(x_{min}(\varepsilon - \varepsilon'))$, and so $p \geq \log_{1+\varepsilon'}(x_{max}(t+1)/(x_{min}(\varepsilon - \varepsilon')))$. One can check that p is bounded by a polynomial in the size of the instance because t is a constant and ε' only depends on ε, that is also a constant. Choosing $\varepsilon' = \varepsilon/2$, we get that $|X| = 3 + \lceil \log_{1+\varepsilon/2}(2x_{max}(t+1)/(x_{min}\varepsilon)) \rceil$ is bounded by a polynomial in the size of the instance (even if x_{max}/x_{min} is exponential). By Lemma 5, we compute in polynomial time an optimal solution x^ε for DFDC with X as set of possible flow values. Recall that t and $|X|$ are constant. Let us define $f^+(x) = \sum_{i=1}^m x_i \mathbb{1}_{x_i \geq x_{max}/(1+\varepsilon')^p}$ and $f^-(x) = \sum_{i=1}^m x_i \mathbb{1}_{x_i < x_{max}/(1+\varepsilon')^p}$ for any vector x. We set $f(x) = f^+(x) + f^-(x)$. Consider an optimal solution x^* for FDC. We get $f(x^*) < f^+(x^*) + mx_{max}/(1+\varepsilon')^p$. We construct an auxiliary vector x' from x^* as follows. For all $i = 1, \ldots, m$, if there exists j, $1 \leq j \leq p$, such that $x_{max}/(1+\varepsilon')^j \leq x_i^* \leq x_{max}/(1+\varepsilon')^{j-1}$, then set $x_i' = x_{max}/(1+\varepsilon')^j$. For all $i = 1, \ldots, m$, if $x_i^* < x_{max}/(1+\varepsilon')^p$, then set $x_i' = 0$. We get $f^+(x^*)/f(x') \leq 1+\varepsilon'$ by construction of x'. As x' takes flow values in X, then $f(x^\varepsilon) \geq f(x')$ because x^ε is an optimal solution for DFDC. Thus, $f^+(x^*)/f(x^\varepsilon) \leq 1+\varepsilon'$.

We now prove that $f(x^*)/f(x^\varepsilon) \leq 1+\varepsilon$. By previous claims, it is sufficient to prove that $f^+(x^*)/f(x^\varepsilon) + (mx_{max})/((1+\varepsilon')^p f(x^\varepsilon)) \leq 1+\varepsilon$. First, recall that $f^+(x^*)/f(x^\varepsilon) \leq 1+\varepsilon'$. It remains to prove that $(x_{max}m)/((1+\varepsilon')^p f(x^\varepsilon)) \leq \varepsilon - \varepsilon'$. The polynomial time algorithm described in Lemma 5 finds, at least, a set of $m/(t+1)$ disjoint paths (connections) with amount of flow (at least) x_{min} (other connections can be zero). Thus, as $t+1 \geq m/|IS(H)|$ and $x_{min} \in X$, then $f(x^\varepsilon) \geq mx_{min}/(t+1)$. The corresponding nodes forms an independent set of H. Recall that p is such that $x_{max}/(1+\varepsilon')^p \leq (\varepsilon - \varepsilon')x_{min}/(t+1)$, and so $x_{min} \geq (x_{max}(t+1))/((1+\varepsilon')^p(\varepsilon - \varepsilon'))$. Thus, $f(x^\varepsilon) \geq (x_{max}m)/((1+\varepsilon')^p(\varepsilon - \varepsilon'))$, and so $(x_{max}m)/((1+\varepsilon')^p f(x^\varepsilon)) \leq \varepsilon - \varepsilon'$. We conclude that $f(x^*)/f(x^\varepsilon) \leq 1+\varepsilon$, and so the polynomial time algorithm of Lemma 5 returning x^ε is a $(1+\varepsilon)$-approximation algorithm for FDC. □

We deduce a PTAS for two other classes of instances. Let us first define $\chi_e = |\{i : E(P_i) \cap e \neq \emptyset, i = 1, \ldots, m\}|$ for all $e \in E$, as the number of paths that contain edge e. Let $\chi_G = \max_{e \in E} \chi_e$ be the maximum number of paths that share an edge. Let Δ_G be the maximum degree of G.

If G is a tree, then the treewidth $tw(H)$ of the graph H is bounded by $\Delta_G \, \chi_G$. We deduce Corollary 2.

Corollary 2. *Let $d, t \geq 1$ be any constant values. For any $\varepsilon > 0$, there is a polynomial $(1+\varepsilon)$-approximation algorithm for FDC when the graph $G = (V, E)$ is a tree, $\Delta_G \leq d$, and $\chi_G \leq t$.*

If G is a path, then the treewidth $tw(H)$ of the graph H is bounded by χ_G. We deduce Corollary 3.

Corollary 3. *Let $t \geq 1$ be any constant integer. For any $\varepsilon > 0$, there is a polynomial $(1+\varepsilon)$-approximation algorithm for FDC when the graph G is a path and $\chi_G \leq t$.*

However, the complexity of FDC remains open when the graph G is a path if χ_G is not bounded. More precisely, we do not know if the problem is NP-hard and if there exists a polynomial approximation algorithm with approximation ratio less than 2.

5. Conclusion and future work

In this paper, we introduced the FDC and we proposed polynomial time approximation scheme and constant factor approximation algorithms for some classes of instances. However, the complexity of FDC is still unknown while the problem has been proved to be NP-hard in presence of capacity constraints [2]. The two main open questions are the following. First, we conjecture that FDC is NP-complete in general. Thus, an important problem is to characterize the classes of instances for which FDC is NP-complete and to develop polynomial time exact algorithms for the others. Second, an interesting problem consists in obtaining better approximation ratios for the classes of instances investigated in this paper and to develop approximation algorithms for other ones (considering here classes for which we do not know exact polynomial time algorithms). To show the difficulty of such questions, when the graph G is a path, we do not know if FDC is in P while the best known polynomial time approximation algorithm gives an approximation ratio equal to two.

References

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., 1993. Network flows: theory, algorithms, and applications. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[2] Ben-Ameur, W., Ouorou, A., 2006. Mathematical models of the delay constrained routing problem. Algorithmic Operations Research 1 (2).

[3] Bertsekas, D., Gallager, R., 1992. Data networks (2nd ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[4] Cormen, T. H., Stein, C., Rivest, R. L., Leiserson, C. E., 2001. Introduction to Algorithms, 2nd Edition. McGraw-Hill Higher Education.

[5] Correa, J. R., Schulz, A. S., Moses, N. E. S., 2007. Fast, fair, and efficient flows in networks. Operations Research 55 (2), 215–225.

[6] Even, S., Itai, A., Shamir, A., 1976. On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. 5 (4), 691–703.

[7] Hijazi, H., Bonami, P., Cornuéjols, G., Ouorou, A., 2012. Mixed-integer nonlinear programs featuring "on/off" constraints. Comp. Opt. and Appl. 52 (2), 537–558.

[8] Minoux, M., 1989. Networks synthesis and optimum network design problems: Models, solution methods and applications. Networks 19 (3), 313–360.

[9] Ouorou, A., Mahey, P., Vial, J.-P., 2000. A survey of algorithms for convex multicommodity flow problems. Management Science 46 (1), 126–147.

[10] Robertson, N., Seymour, P., 1984. Graph minors. III. planar tree-width. Journal of Combinatorial Theory, Series B 36 (1), 49 – 64.

[11] Schrijver, A., 2003. Combinatorial optimization. Polyhedra and efficiency. Algorithms and Combinatorics. Vol. 24. Springer-Verlag, Berlin.

[12] Touati, C., Altman, E., Galtier, J., 2003. Semi-definite programming approach for bandwidth allocation and routing in networks. Game Theory and Applications 9, 169–179.