Noname manuscript No.
(will be inserted by the editor)

Modeling max-min fair bandwidth allocation in BitTorrent communities

Elvira Antal · Tamás Vinkó

Received: / Accepted:

Abstract This paper gives an exact mathematical programming model and algorithm of the max-min fairness bandwidth allocation problem in multi-swarm peer-to-peer content sharing community. The proposed iterative method involves solution of LP and MILP problems of large scale. Based on real-world data traces, numerical experiments demonstrate that the new algorithm is computationally faster than an earlier developed one for larger problem sizes, and it provides better numerical stability. Moreover, even if its execution is stopped after some initial steps it still grants feasible solution with good approximation to max-min fairness.

Keywords BitTorrent communities · resource allocation · max-min fairness · MILP

1 Introduction

BitTorrent is one of the most popular content-sharing protocols used by millions of Internet users [16]. It is based on peer-to-peer (P2P) technology which, in contrast with centralized solutions, consists of nodes (peers) acting both as servers and clients at the same time. This decentralized approach can lead to high efficiency and extreme scalability. A BitTorrent network is basically a collection of *swarms*. A swarm is comprised of content and two types of users, namely seeders and leechers. The *content* is the data file to be shared by the participating users. *Seeders* are those users who have the complete copy of the content, being online and willing to share. *Leechers* are

E. Antal
University of Szeged, Institute of Informatics, and
Kecskemét College, Faculty of Mechanical Engineering and Automation
E-mail: antale@inf.u-szeged.hu

T. Vinkó
University of Szeged, Institute of Informatics
H-6701 Szeged, P.O. Box 652, Hungary
Tel.: +36-62-546-193
Fax: +36-62-546-397
E-mail: tvinko@inf.u-szeged.hu

those users who are actively downloading the content. By definition, the BitTorrent protocol splits the content into smaller pieces. During the download the leechers are obtaining the content piece-by-piece, following Rarest First piece selection policy. Once a piece has been downloaded by a leecher, it can upload that to other leechers, hence becoming an uploader.

An important detail of the BitTorrent protocol [5] is the built-in sharing incentive mechanism, realized via the choking algorithm, which is a variant of tit-for-tat [6]. This mechanism ensures that while a peer p is in leeching mode it uploads pieces of the shared content to a set of leeching peers in reciprocation to other pieces they provide for downloading to p. While this scheme works very well during the leeching phase, there is no (widely spread) incentive mechanism in BitTorrent which works for the seeders. Thus, in principle, once the whole content has been downloaded, the peer can simply leave the system without further consequences. Among the possible solutions to this problem, one of the most popular is the so-called private BitTorrent community [25]. Although it breaks the decentralized principle of the P2P system with a dedicated server, the idea is that each user has to register with an individual account and follow some prescribed rules, e.g., sharing ratio enforcement. The *sharing ratio* of a user is defined as the amount of data uploaded divided by the amount of data downloaded. These values are then stored by the server (also called *tracker*) and the users who do not follow the rules, e.g., their sharing ratio is below a certain threshold for long time, are subject to access restrictions or even exclusion from the community. According to measurement studies the private BitTorrent communities provide higher download speed and better availability compared to the open BitTorrent networks [17].

Most of the BitTorrent clients allow users to participate in multiple swarms at the same time, both as seeders and leechers. This fact motivates to investigate the inter-swarm resource allocation problem (RAP) in BitTorrent, which is, in general, a mixed-integer nonlinear optimization problem [4]. A particular instance of RAP is the max-min fair bandwidth allocation problem. In this optimization problem, the goal, essentially, is to find a bandwidth allocation which provides as many users as possible with enough download speed. Although, BitTorrent was originally not designed for P2P video-streaming, many researchers have investigated and proposed modifications of the protocol (see, e.g., in [18,21,26]). In this context, the max-min fair bandwidth allocation targets maximizing the number of users receiving sufficient download speed for streaming, leading to the best possible quality of experience for users. Moreover, it also enables the usage of multiple streaming rates of varying qualities together with the minimization of the number of users experiencing low-quality streams. This problem has been studied in [4], where an intricate iterative algorithm was given. In this paper we revisit this interesting problem instance and aim at giving an exact mathematical programming formalism, and investigating its numerical properties by means of computational tests using real-world measurement data.

The rest of the paper is structured as follows. In Section 2 an overview of related works is given. Section 3 summarizes the definitions and notations of the graph model we use. Then, in Section 4, after giving the formal definition of the max-min fair bandwidth allocation, the proposed algorithmic approach is detailed, including mathematical analysis, re-formalism and proof of the correctness. Section 5 contains the numerical experiments, including comparison with the previously proposed method for the same problem.

2 Related works

Although the concept of fairness and in particular the max-min fairness have been studied in the literature of computer networks in general (e.g., in [2, 13, 14, 20]), one finds considerably less papers in the context of peer-to-peer networks, and in particular in BitTorrent-like systems.

Max-min fairness in P2P networks. Ma *et al.* [15] develop a resource bidding mechanism which provides max-min fairness. An important result of the paper is that the mechanism is incentive based, so two competing nodes with the same value of bidding would not obtain the same amount of resource if their actual contribution to the P2P community differs. Our approach does not involve directly the already mentioned built-in incentive mechanism of BitTorrent (tit-for-tat) because that is the piece-level part of the protocol. We assume, though, that the participating peers are following the rules dictated by the private BitTorrent community. It is worth mentioning here that according to the earlier results in [4], standard BitTorrent provides suboptimal bandwidth allocation compared to max-min fairness. Yan *et al.* [24] present a theoretic framework of optimal resource allocation and admission control for P2P networks. The proposed approach utilizes publicly observable and verifiable information to achieve optimal resource allocation. Our paper differs in many ways. Firstly, we focus on bandwidth allocation, which is the most important resource in content-sharing systems. Secondly, our model works at the inter-swarm level, which is the most complex level due to the behaviour of users, i.e., most of the users are participating in uploading and downloading multiple contents at the same time. Moreover, our experiments are done on real-world measurements at large scale.

BitTorrent-like systems. Fan *et al.* [8] show that there is a fundamental trade-off between keeping fairness and providing good download rate in BitTorrent-like systems. Measuring fairness is done using the so-called fairness index [14] which can express how equal a given assignment is, where the assignment was the peers' sharing ratio, i.e., the uploading amount divided by the download amount of each peer. The paper deals with the max-min fair allocation as rate assignment strategy. It considers such a max-min model in which the *overall* downloading speed of peers are taken into account. Similar approach was taken in the paper of Eger & Killat [7]. Our model gives solution to the problem of optimizing *individually* the downloading sessions of peers. This means, essentially, that we take into account more details motived by measurement facts. The problem of channel-resource imbalance in multi-channel P2P systems, which corresponds to the performance optimization of live streaming, is considered by Wu *et al.* [22]. Our work definitely fits into the same context of multi-channel P2P live streaming. While the provided solution in [22] is heuristic based, we give an exact mathematical model based on theoretical analysis. The paper of Wu *et al.* [23] gives a distributed algorithm to tune the P2P live video system towards the optimal fairness while still maintaining the targeted universal streaming rate. Our paper differs in the underlying model. They consider a dynamic P2P system with video streaming servers, leading to a nonlinear optimization problem. We analyse a static system without central component and more importantly, the flow network model we use allows the usage of efficient linear programming techniques.

3 Notation

For modeling the state of a BitTorrent community at a certain instant, we use the graph-theoretical model introduced in [4], which can be summarized as it follows. A BitTorrent community consists of a set I of users and a set T of torrents. Note that technically speaking there is a difference between torrent (a metafile describing the details of the file subject to download) and swarm (collection of leechers and seeders of the file); we can use these two terms interchangeably. Each user $i \in I$ has upload bandwidth μ_i and download bandwidth δ_i. The flow network representation of a BitTorrent community is $G = (\{U, L, D\}, E, f, c)$, which is a directed, bipartite, weighted graph, where

$U = \{ u_i \mid i \in I\}$: the *upload nodes* of G, where u_i represents the upload (seeding or leeching) potential of user i;

$D = \{ d_i \mid i \in I\}$: the *download nodes* of G, where d_i represents the download (leeching) potential of user i;

$L = \{ l_i^t \mid i \in I,\, t \in T\}$: the *leeching nodes* of G, where the presence of l_i^t, called *leeching session*, denotes that user i leeches actually torrent t;

E: the set of edges $E = E_U \cup E_D$, where $E_U = \bigcup_{i,j,t}(u_i, l_j^t)$ is the set of *upload edges*, and $E_D = \bigcup_{j,t}(l_j^t, d_j)$ is the set of *download edges*;

$c : U \cup L \cup D \to \mathbb{N}$: the *capacity function* represents the bandwidth constraints of the peers:

$$c(u_i) = \mu_i,\; c(d_i) = \delta_i,\; c(l_i^t) = \infty;$$

$f : E \to \mathbb{R}^+$: the *flow function* represents the bandwidth allocation on the edges satisfying the flow conservation property:

$$\sum_{u_i \in U} f(u_i, l_j^t) = f(l_j^t, d_j) \qquad (\forall l_j^t \in L),$$

as well as the *capacity constraints*:

$$\sum_{t,j} f(u_i, l_j^t) \le \mu_i \qquad \forall (u_i, l_j^t) \in E_U,$$

$$\sum_{t} f(l_j^t, d_j) \le \delta_j \qquad \forall (l_j^t, d_j) \in E_D.$$

Figure 1 contains a small example for the bipartite graph representation of a snapshot of a BitTorrent network with two torrents and three users. The rectangles represent two different torrent files. There are two active leeching sessions for torrent t_2. The second user downloads the second file from the third user, and the third user

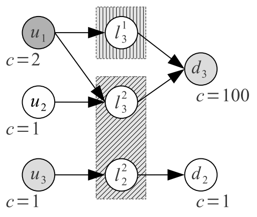


Fig. 1 Bipartite graph of a two-torrent BitTorrent network. Although the graph model has three types of nodes, strictly speaking the BitTorrent community graphs are bipartite.

downloads the first file from the first user and the second file from the first and the second users. Note that every leeching node corresponds to exactly one download edge.

4 Max-min fair bandwidth allocation

4.1 Problem definition

Using the flow network model from Section 3, a bandwidth allocation is *max-min fair* if the flow value $f(l_j^t, d_j)$ on a download edge (l_j^t, d_j) can only be increased by decreasing the flow value $f(l_{j'}^{t'}, d_{j'})$ on another download edge $(l_{j'}^{t'}, d_{j'})$ for which $f(l_{j'}^{t'}, d_{j'}) < f(l_j^t, d_j)$.

A max-min fair allocation assures that the highest possible downloading speed is provided to each user, so that they perceive the best possible quality of experience. Remark that a leecher can have multiple leeching sessions representing multiple file downloads at the same time. The problem formulation we are dealing with in this work requests the max-min fairness for *all* download edges. As it was already stated in [4], the problem is formulated on continuous and convex set, hence the max-min fair allocation uniquely exists [2,19].

4.2 Algorithm outline

Our algorithm is an adapted version of the general Max-Min Programming Algorithm [19], as it computes the max-min fair weights of the download edges with an iterative manner, by fixing the coordinates with the smallest unfixed weight in every iteration. In the subsequent description, sets are denoted by capital letters, the decision variables of the optimization problems by small letters and parameters (and fixed values) by Greek letters. The following steps build up the algorithm:

1. **Initialization.** Let $F := \emptyset$, $k := 1$, $E_1 := E_D$, and $\forall (l_j^t, d_j) \in E_D : \ell_j^t := 0$.

The set F contains the identifiers of the actually fixed flows, k is the iterator. After the last iteration, ℓ_j^t contains the optimal flow values for every download edge $(l_j^t, d_j) \in E_D$.

2. **Lower bound computation for the flows.** Solve the following linear programming (LP) problem, denoted by *MM*$_0$:

$$\begin{aligned} &\max f, \\ \text{s.t. } &f(l_j^t, d_j) \geq f \qquad\qquad \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   Save the minimal flow value. Let $\phi := f$.

Note that *MM*$_0$ is the same problem, as published as a detail of the MaxMin algorithm of Capotă *et al.* [4]. Its optimal solution is calculated only once to achieve a good lower bound for the flows in Step 3. By definition, $\forall (u_i, l_j^t), (l_j^t, d_j) \in E : f(u_i, l_j^t), f(l_j^t, d_j) \in \mathbb{R}^+$, and $f_k \in \mathbb{R}^+$ at every presence.

3. **LP solving.** Solve the following LP problem, denoted by *mMM*$_k^{(1)}$:

$$\begin{aligned} \max\ & f_k + \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t, \\ \text{s.t. } & f(l_j^t, d_j) \geq f_k \qquad\qquad \forall (l_j^t, d_j) \in E_k, \\ & f_k \geq \phi. \end{aligned}$$

   Save the LP optimum. Let $\sigma_k := \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t$, and $\phi_k := f_k$.

The LP problem *mMM*$_k^{(1)}$ combines the maximum flow and the max-min fair objective. It computes the maximum throughput of the network, denoted by σ_k, restricted by the fulfillment of the max-min fairness property. As it will be shown in Lemma 2, this amount of data transfer is guaranteed in every iteration of the proposed algorithm. The max-min fair allocation, which still guarantees the maximum throughput σ_k, will be computed in the next step. The aim of this step is to compute σ_k, ϕ_k, and to offer a good initial (feasible) solution for the following MINLP of special type.

4. **MINLP solving.** Solve the following mixed-integer bilinear programming problem, denoted by *mMM*$_k^{(2)}$:

$$\begin{aligned} \max\ & \sum_{(l_j^t, d_j) \in E_k} x_j^t, \\ \text{s.t. } & \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j)\, x_j^t + \phi_k \cdot \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t = \sigma_k, \\ & f(l_j^t, d_j) \geq \phi_k \qquad\qquad \forall (l_j^t, d_j) \in E_k, \\ & f(l_j^t, d_j) > \phi_k\, x_j^t \qquad\qquad \forall (l_j^t, d_j) \in E_k, \end{aligned}$$

   where $x_j^t \in \{0,\ 1\}$.

The bilinear MINLP problem *mMM*$_k^{(2)}$ guarantees that the max-min flow value ϕ_k will be fixed for the least possible download edges in every iteration. In fact, this is a combinatorial optimization problem. The strict inequality constraint sets the binary x_j^t variable to zero if $f(l_j^t, d_j)$ cannot be increased above ϕ_k in later iterations, so $f(l_j^t, d_j)$ should be fixed to ϕ_k only if $x_j^t = 0$. Lemma 4 of Subsection 4.3 warrants at least one feasible solution for this problem.

In order to solve *mMM*$_k^{(2)}$ efficiently, it will be reformulated using the McCormick envelopes [12]. This results in an equivalent mixed-integer linear programming (MILP) problem, in which the bilinear terms are substituted by new continuous variables

$$p_j^t := f(l_j^t, d_j) \cdot x_j^t,$$

where $\forall (l_j^t, d_j) \in E_k : p_j^t \in \mathbb{R}^+$. Furthermore, the reformulation involves four additional constraints for every new variable p_j^t:

$$\begin{aligned}
&p_j^t \le f(l_j^t, d_j) && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \le \delta_j \cdot x_j^t && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \ge f(l_j^t, d_j) - \delta_j \cdot (1 - x_j^t) && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \ge 0 && \forall (l_j^t, d_j) \in E_k.
\end{aligned}$$

Although, the dimension of the problem is increased, an exact Branch and Bound solver can be applied [3] to find the globally optimal solution of the resulting MILP. We will refer to this problem as the McCormick reformulation of *mMM*$_k^{(2)}$.

5. **Fixing.** Find the binding constraints for ϕ_k, and fix the flow values of the adequate download edges. In other words, collect the download edges, where the optimal flow is equal to ϕ_k to a set Φ_k, add the elements of Φ_k to F, and subtract them from E_k. Formally,

$$\Phi_k := \left\{ (l_j^t, d_j) \in E_k \mid x_j^t = 0 \right\},$$

$$\forall (l_j^t, d_j) \in E_k \text{ for which } x_j^t = 0 \ : \ \ell_j^t := \phi_k,$$

$$F := F \cup \Phi_k, \quad E_{k+1} := E_k \setminus \Phi_k.$$

6. **Stopping criteria.** If $F = E_D$, then stop. Otherwise, $k := k + 1$ and go back to Step 3 (new iteration).

4.3 Correctness

In the following it is proved that the proposed algorithm, called *mMaxMin*, conveys the max-min fair bandwidth allocation.

Denote the optimal objective function value in *mMM*$_k^{(1)}$ as follows:

$$\mathcal{F}_k := \phi_k + \sigma_k.$$

Lemma 1 *Every feasible solution of* mMM$_{k-1}^{(2)}$ *can be mapped to a feasible solution of* mMM$_k^{(1)}$ *for* $k > 1$.

Proof Let $f_k := \phi_{k-1}$ and

$$f_k^{(1)}(l_j^t, d_j) := f_{k-1}^{(2)}(l_j^t, d_j) \qquad \forall (l_j^t, d_j) \in E_k,$$

after the fixing step of *mMaxMin*, where $f_k^{(y)}(l_j^t, d_j)$ denotes the flow values $f(l_j^t, d_j)$ in a feasible solution of *mMM*$_k^{(y)}$. □

The mapping of the optimal solution of *mMM*$_{k-1}^{(2)}$ will be referred to as **the initial solution of *mMM*$_k^{(1)}$**.

Lemma 2 *For every iteration k of* mMaxMin

$$\sigma_k = \sigma$$

holds, where σ denotes a constant, the maximum throughput of the network such that

$$\forall (l_j^t, d_j) \in E_D : f(l_j^t, d_j) \geq \phi.$$

Proof We apply mathematical induction. There are no fixed flow values in the first step, so $\sigma_1 = \sigma$. Now, let us assume that $\sigma_{k-1} = \sigma$. The sum term of the objective function, after Step 3 of *mMaxMin* can be written as

$$\sigma_k = c_1\phi_1 + \cdots + c_{k-1}\phi_{k-1} + c_k\phi_k + R_k,$$

where $\phi_1, \ldots, \phi_{k-1}$ are the fixed flow values (the optimal values of f_k in *mMM*$_1^{(1)}, \ldots,$ *mMM*$_{k-1}^{(1)}$), ϕ_k is the minimal non-fixed flow value, c_i is the multiplicity of ϕ_i (i.e. how many downloading edges has flow value equal to ϕ_i), and R_k is the residual (the sum of the non-fixed flow values minus the ones which will be fixed in the actual iteration). Similarly,

$$\sigma_{k-1} = c_1\phi_1 + \cdots + c_{k-1}\phi_{k-1} + R_{k-1},$$

and

$$\sigma_{k-1} - \sigma_k = R_{k-1} - R_k - c_k\phi_k.$$

If $R_{k-1} - R_k$ would be greater than $c_k\phi_k$, that would mean that the LP solver reduced some non-fixed flow values of the initial solution, defined in the proof of Lemma 1, without redistributing that flow to other edges in *mMM*$_k^{(1)}$. That would be a suboptimal solution, and the solver would not terminate with such a result. Thus, $R_{k-1} - R_k \leq c_k\phi_k$ holds, and accordingly, $\sigma_{k-1} \leq \sigma_k$.

On the other hand, $\sigma_k \leq \sigma$ for any iteration k, as the upload capacities does not change in the network in between the iterations of the algorithm. By assumption, $\sigma_{k-1} = \sigma$, thus $\sigma_k = \sigma$ holds for every k. □

Lemma 3 $\mathcal{F}_k > \mathcal{F}_{k-1}$, *for all iteration $k > 1$ of* mMaxMin.

Proof In the initial solution of *mMM*$_k^{(1)}$, all $f(l_j^t, d_j) \leq \phi_{k-1}$ flow values are fixed, because of Step 5 of *mMaxMin*. Thus $\phi_k > \phi_{k-1}$, and $\sigma_k = \sigma_{k-1} = \sigma$ holds due to Lemma 2. □

Corollary 1 $|\Phi_k| > 0$ *in every* k *iteration of* mMaxMin.

Lemma 4 *The optimal solution of* $\mathrm{mMM}_k^{(1)}$ *can be mapped to a feasible solution of* $\mathrm{mMM}_k^{(2)}$.

Proof Let

$$f_k^{(2)}(l_j^t, d_j) := f_k^{(1)}(l_j^t, d_j),$$

and

$$x_j^t := \begin{cases} 1 & \text{if } f_k^{(1)}(l_j^t, d_j) > \phi_k \text{ and } (l_j^t, d_j) \in E_k, \\ 0 & \text{if } f_k^{(1)}(l_j^t, d_j) = \phi_k \text{ and } (l_j^t, d_j) \in E_k. \end{cases}$$

□

The mapping of the optimal solution of $mMM_k^{(1)}$ will be referred to as **the initial solution of** $\boldsymbol{mMM_k^{(2)}}$.

Lemma 5 *The initial solution of the McCormick reformulation of* $\mathrm{mMM}_k^{(2)}$ *can be constructed from the initial solution of* $\mathrm{mMM}_k^{(2)}$.

Proof The initial solution of $mMM_k^{(2)}$ is extended with initial values for the p variables:

$$p_j^t := \begin{cases} f_k^{(1)}(l_j^t, d_j) & \text{if } x_j^t = 1 \text{ and } (l_j^t, d_j) \in E_k, \\ 0 & \text{if } x_j^t = 0 \text{ and } (l_j^t, d_j) \in E_k. \end{cases}$$

□

Theorem 1 mMaxMin *terminates in finite iterations, and guarantees the max-min fairness property for every download edge.*

Proof The cardinality of set F is increasing in every iteration, provided by Lemma 3 and Corollary 1. As E_D is a finite set, the algorithm will terminate in finite iterations.

Due to Lemma 1 and Lemma 4–5, at least one feasible solution exists for $mMM_k^{(1)}$ and $mMM_k^{(2)}$ in any iteration k of *mMaxMin*. After the last iteration the set $\ell := \{\ell_j^t \mid (l_j^t, d_j) \in E_D\}$ contains the fixed flow values for the download edges. The boundaries of the flow values constrain also the elements of ℓ, thus $0 \leq \ell_j^t \leq \delta_j, \ \forall (l_j^t, d_j) \in E_D$. Therefore, ℓ is a compact set. All the constraints for the flow values are linear, hence ℓ is a convex set. Radunović and Le Boudec [19] proved that there exists a max-min fair bandwidth allocation for convex and compact sets. Let us denote the max-min fair bandwidth allocation for the download edges of the given graph by ω:

$$\omega := \{\omega_j^t \mid (l_j^t, d_j) \in E_D \text{ and } \omega \text{ is max-min fair}\}.$$

We prove by contradiction that *mMaxMin* guarantees the max-min fairness property for every download edge. Suppose $\ell \neq \omega$. Then there exists the smallest index k such that $\exists j \exists t : \left(\ell_j^t \text{ is fixed in iteration } k \text{ and } \ell_j^t \neq \omega_j^t\right)$. It means that $x_j^t = 0$ and $f(l_j^t, d_j) \neq \omega_j^t$ in the optimal solution of $mMM_k^{(2)}$. Remark that $f(l_j^t, d_j) \leq \omega_j^t$, as ω would not be max-min fair otherwise. The construction of $mMM_k^{(2)}$ guarantees that

$\left(x_j^t = 0 \wedge f(l_j^t, d_j) = \phi_k\right) \vee \left(x_j^t = 1 \wedge f(l_j^t, d_j) > \phi_k\right)$ holds for all $(l_j^t, d_j) \in E_D$, and if $f(l_j^t, d_j)$ could be set to a greater value than ϕ_k, then x_j^t will be set to 1. So $x_j^t = 0$ induces that $f(l_j^t, d_j) = \phi_k$. On the other hand, ϕ_k is the max-min non-fixed flow value in iteration k from the solution of *mMM*$_k^{(1)}$. This contradicts the supposition that $\ell_j^t \neq \omega_j^t$. $\square$

4.4 MaxMin-r

Some observations from Subsection 4.3 was made explicit in the implemented version of the algorithm. Furthermore, we have inserted a presolve step, detailed hereinafter. The resulting iterative algorithm, called *MaxMin-r*, is summarized in Algorithm 4.1. *MaxMin-r* was implemented in the AMPL modeling language [10] and some comparative tests were made to investigate its numerical properties – the details of these tests are given in the next section.

The main differences compared to *mMaxMin* are the following:

1. Step 2 was introduced, based on Lemma 2, to determine the constant σ. The LP problem *MM*$_{\text{MaxFlow}}$ is solved only in the first iteration, and the revised *mMM*$_k^{(1)}$ uses σ in the first constraint.
2. In *mMM*$_k^{(1)}$, the lower bound "$\geq (1-\epsilon) \cdot \sigma$" is used instead of a strict equation constraint "$= \sigma$" in regard to possible numerical errors.
3. Step 5 of *MaxMin-r* introduces a presolve phase, based on a standard LP presolve technique, which is implemented also in AMPL [9,11]. During the testing phase of earlier implementation of the algorithm we noticed that the presolving mechanism of AMPL was able to reduce the number of real variables of the MILP problem. Closer investigation revealed that the set

$$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \mid \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{deg_k^-(d_j)} = \phi_k \right\},$$

where $deg_k^-(d_j)$ denotes the number of non-fixed incoming edges of d_j, contains download edges, where the corresponding flow values $f(l_j^t, d_j)$ can be fixed by Step 7 of *MaxMin-r*.

If any flow can be fixed in the presolve phase, MILP solving is skipped. The reason is experimental: for our test cases, in a significant proportion of the iterations, all the necessary fixations were found in this presolve phase. However, in certain cases there are some downloading edges on which the optimal max-min flow value is ϕ_k and they do not become elements of the set E_{k_f}. If this situation occurs then the value of ϕ_k cannot be improved in Step 4 of the next iteration, i.e. $\phi_{k+1} = \phi_k$. Hence, the set E_{k_f} is empty, so in order to find downloading edges on which the flow value must be fixed, the algorithm solves the MILP problem.

Due to this modification, in worst-case, the algorithm takes $2 \cdot |E_D|$ iterations. As it can be seen in Section 5, much less iterations are usually enough in practice.

Figure 2 contains a small illustration for dimension reduction without solving the MILP. User 5 downloads five torrents at the same time, and the maximal flow

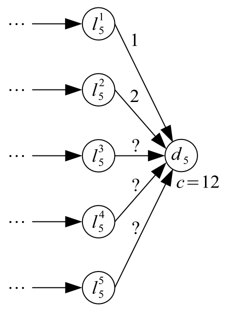


Fig. 2 Example of possible dimension reduction without MILP solving

for downloading the first two torrents was set to 1 and 2 in earlier iterations of *MaxMin-r*. Assume that $\phi_k = 3$. So $f(l_5^t, d_5) \geq 3$ for $t = 3, 4, 5$. The residual download capacity of d_5 in this iteration is $12 - (1 + 2) = 9$, so 3 is also the maximum value for these flow values. Therefore, for $t = 3, 4, 5$ the flow values $f(l_5^t, d_5)$ can all be set to 3 without solving the MILP.

4. Step 6 of *MaxMin-r* uses the McCormick reformulation of *mMM*$_k^{(2)}$.

5 Numerical results

For the numerical tests the post-processed BitTorrent measurement traces of Andrade *et al.* [1] were used. The same dataset was investigated in [4] in which the *MM* algorithm was proposed and empirically tested. The post-processed dataset contains actual statuses of a BitTorrent community called BitSoup.org using the graph format discussed in Section 3. The graphs are implemented in AMPL data format. For our current purposes we selected one graph G randomly and based on that four instances (G_{500}, G_{1000}, G_{1500} and G_{2000}) were derived containing 500, 1000, 1500 and 2000 torrents, respectively. More precisely, these subgraphs contain the corresponding U, L and D nodes of G and their edges. The characteristics of the subgraphs are shown in Table 1. Note that G_{1500} contains less edges than G_{1000}, however, it contains much more nodes, and more edges representing leeching sessions.

Table 1 Characteristics of the graphs used for the numerical tests

Graph	$\lvert U \cup D \cup L \rvert$	$\lvert E \rvert$	$\lvert E_D \rvert$
G_{500}	6 984	43 410	1 411
G_{1000}	14 702	272 231	2 721
G_{1500}	18 333	269 165	3 536
G_{2000}	23 670	524 054	7 326

We compare the AMPL implementations of *MM* and *MaxMin-r*. The results were obtained using MOSEK version 7.0.0.106 for the underlying LPs and Gurobi version 5.6.3 for the underlying MILPs.

Figure 3 shows $f_{\text{MaxMin-r}}(e) - f_{\text{MM}}(e)$, the difference between the optimal flow value of *MaxMin-r* and the optimal flow value of *MM* for download edge $e \in E_D$ in the 1000-torrents instance (related data series are similar for all examples). The values are ordered ascending by the optimal solution of *MM*. Thus positive numbers on the left side of the figure and negative ones on the right side means that *MaxMin-r* provides better flow values than *MM* for some "weak" downloader at the expense of a few "stronger" users. In other words, the new algorithm results in "fairer" allocation than *MM* despite of the similar precision and tolerance settings. How is that possible?

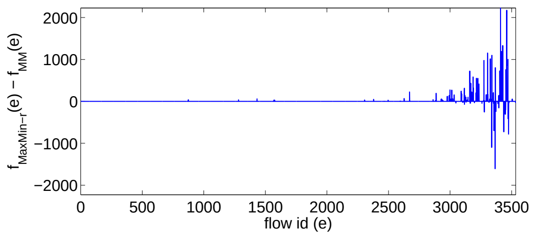


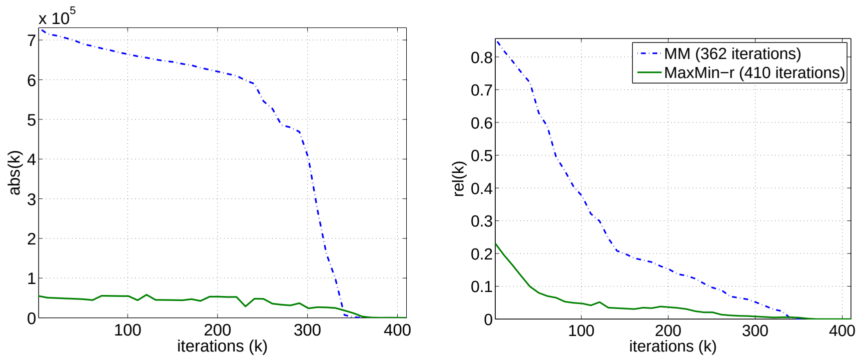
Fig. 3 Effect of the numerical approximation for G_{1000}

Unfortunately, computer implementation turns the continuous optimization problem into a discrete problem, because of the floating point representation of the real variables. The precision and tolerance settings of the numeric solver definitely influence the quality of the produced allocation. Because of that, earlier theoretical results (optimal max-min fair allocation is unique in the continuous case [2]) could not be applied in the numerical tests. Remark that working with symbolic representation would solve this issue, however, for real-world problem instances, even numerical methods are quite slow. On the other hand, the cumulative distribution of the output flow values of *MM* and *MaxMin-r* are identical for the same problems, and more than 85% of the download edges get identical resources from the two algorithm. Therefore we regard the two solutions equally good hereinafter.

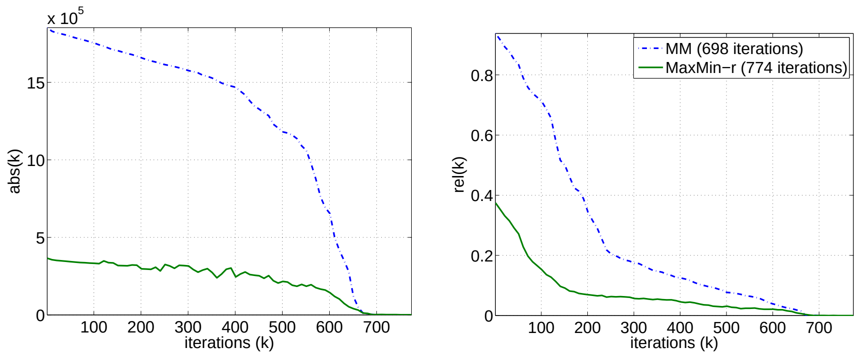
Figure 4 summarizes two aspects of the behaviour of *MM* and *MaxMin-r* for the above introduced 500-torrents, 1000-torrents, 1500-torrents, and 2000-torrents problems. The first column shows the total absolute deviance from the optimal solution:

$$\text{abs}(k) = \sum_{(l_j^t, d_j) \in E_D} \left| f_k(l_j^t, d_j) - f_{\text{opt}}(l_j^t, d_j) \right| ,$$

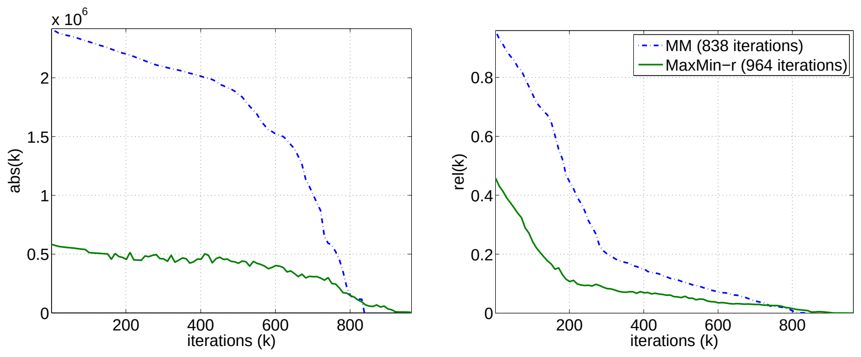
where $f_k(l_j^t, d_j)$ is the flow value on the download edge (l_j^t, d_j) in iteration k, and $f_{\text{opt}}(l_j^t, d_j)$ is the optimal flow value on the same edge, i.e., the result of the last iteration of the relevant algorithm.



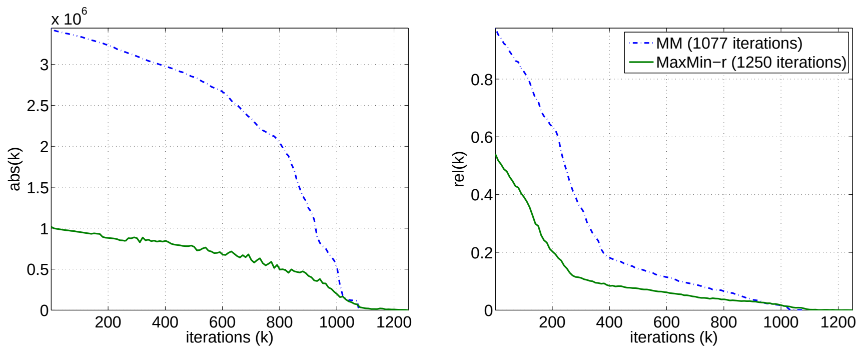
G_{500}



G_{1000}



G_{1500}



G_{2000}

Fig. 4 Quality of results of *MM* (dash-dotted lines), and *MaxMin-r* (solid lines) for the test cases.

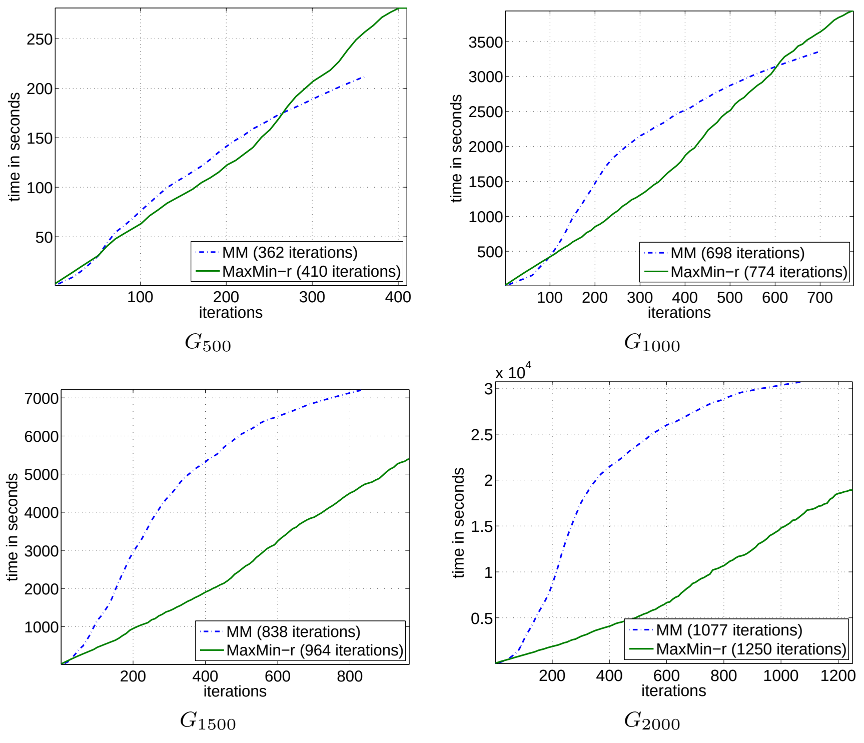


Fig. 5 Running times of *MM* (dash-dotted lines), and *MaxMin-r* (solid lines) for the test cases

The second column shows the proportion of the download edges, for which the relative deviance of the allocated flow values from the optimal solution is less than five percent:

$$\text{rel}(k) = \frac{\sum_{(l_j^t, d_j) \in E_D} r\left(k, (l_j^t, d_j)\right)}{|E_D|},$$

where

$$r\left(k, (l_j^t, d_j)\right) := \begin{cases} 1 & \text{if } \frac{\left|f_k(l_j^t, d_j) - f_{\text{opt}}(l_j^t, d_j)\right|}{f_{\text{opt}}(l_j^t, d_j)} > 0.05, \\ 0 & \text{otherwise.} \end{cases}$$

The running times for the same tests are pictured in Figure 5.

It seems that the exact formulation in *MaxMin-r* produces very good solutions from the first iteration. Comparing total absolute deviance, the output of the new algorithm after the first iteration is the same quality as the output of *MM* after 85% of its iterations. The second column of Figure 4 shows that *MaxMin-r* sets the flow values close to the optimum on much more edges than *MM* does. For example, 46% of the download edges in G_{2000} get almost optimal allocated flow values after the first iteration, compared to the 2.4% near-optimal flow given by *MM*. Furthermore, the first iteration of *MaxMin-r* took 46 seconds for G_{2000} compared to the more than eight-hour running time for the first 910 iterations of *MM*.

Figure 5 shows, that *MaxMin-r* produces shorter running times than *MM* for the bigger test cases, however, it is still impossible to run real-time calculations for complete BitTorrent networks with this technique. Therefore we suggest to stop *MaxMin-r* after the very first iteration to obtain a good feasible approximation for the max-min fair allocation of large problem instances in reasonable time.

6 Conclusions

It was shown by Capotă *et al*. [4] that using the standard BitTorrent protocol's bandwidth allocation, the average performance of a BitTorrent community is suboptimal in terms of max-min fairness. This fairness measure corresponds to the case of video-streaming service – an emerging application of P2P networks. Our motivation here was to give an exact mathematical programming formulation and algorithm which provides details about the particular instance of this interesting optimization problem.

The model involves the McCormick reformulation of the related MINLP. Our observations show that this reformulation, together with presolve techniques, helps the Gurobi solver to achieve shorter running times, and *MaxMin-r* can be faster than the earlier proposed *MM* algorithm on larger problem instances. Moreover, the results from the first iterations of *MaxMin-r* could be used as a very good approximation for the max-min fair allocation. This approximation, which is a feasible solution, can be achieved in fraction of the time of the adequate precession of *MM*.

There are two possible directions for further work. Due to the unavoidable involvement of solving several large scale MILPs to obtain exact solution to the problem including millions of nodes and edges, it is desired to develop very quick heuristics. Furthermore, as the application field of the max-min fairness problem we investigated lies in peer-to-peer systems, a distributed version of the exact algorithm or even distributed heuristics would be preferred. We believe that the results achieved in this paper provide useful insights towards these goals.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). T. Vinkó was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

References

1. Andrade, N., Santos-Neto, E., Brasileiro, F., Ripeanu, M.: Resource demand and supply in bittorrent content-sharing communities. Comput. Netw. **53**(4), 515–527 (2009)
2. Bertsekas, D.P., Gallager, R.G.: Data Networks, 2nd edn. Prentice Hall (1992)
3. Bonami, P., Kilinç, M., Linderoth, J.: Algorithms and Software for Convex Mixed Integer Nonlinear Programs, *The IMA Volumes in Mathematics and its Applications*, vol. 154, pp. 1–39. Springer (2012)
4. Capotă, M., Andrade, N., Vinkó, T., Santos, F., Pouwelse, J., Epema, D.: Inter-swarm resource allocation in BitTorrent communities. In: Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P 2011), pp. 300–309 (2011)

5. Cohen, B.: The BitTorrent protocol specification. `http://bittorrent.org/beps/bep_0003.html`. Accessed: 19-Aug-2014
6. Cohen, B.: Incentives build robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer systems, vol. 6, pp. 68–72 (2003)
7. Eger, K., Killat, U.: Fair resource allocation in peer-to-peer networks (extended version). Comput. Commun. **30**(16), 3046–3054 (2007)
8. Fan, B., Lui, J.S., Chiu, D.M.: The design trade-offs of BitTorrent-like file sharing protocols. IEEE/ACM Transactions on Networking **17**(2), 365–376 (2009)
9. Fourer, R., Gay, D.M.: Experience with a Primal Presolve Algorithm, In: Hager, W.W., Hearn, D.W., and Pardalos, P.M. (eds), Large Scale Optimization: State of the Art, Kluwer Academic Publishers, Dordrecht, p. 135-154 (1994)
10. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL. Boyd & Fraser (1993)
11. Gay, D.M.: Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming, In: Alefeld, G., Rohn, J., Rump, S. and Yamamoto, T. (eds), Symbolic Algebraic Methods and Verification Methods, Springer-Verlag, pp. 99–106, (2001)
12. Gupte, A., Ahmed, S., Cheon, M., Dey, S.: Solving mixed integer bilinear problems using MILP formulations. SIAM J. Optim. **23**(2), 721–744 (2013)
13. Hahne, E.L.: Round-robin scheduling for max-min fairness in data networks, IEEE Journal on Selected Areas in Communications, **9**(7), 1024–1039 (1991)
14. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley–Interscience, New York, NY (1991)
15. Ma, Richard T. B. and Lee, Sam C. M. and Lui, John C. S. and Yau, David K. Y. A game theoretic approach to provide incentive and service differentiation in P2P networks. SIGMETRICS Perform. Eval. Rev., **32** pp. 189–198 (2004)
16. Maier, G., Feldmann, A., Paxson, V., Allman, M.: On dominant characteristics of residential broadband Internet traffic. In Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference, ACM, pp. 90–102 (2009)
17. Meulpolder, M., D'Acunto, L., Capotă, M., Wojciechowski, M., Pouwelse, J.A., Epema, D.H., Sips, H.J.: Public and private Bittorrent communities: a measurement study. In: Proceedings of the 9th International Workshop on Peer-to-Peer Systems (IPTPS) (2010)
18. Mol, J.J.D., Bakker, A., Pouwelse, J., Epema, D., Sips H.: The design and deployment of a Bittorrent live video streaming solution. In: Proceedings of the 11th IEEE International Symposium on Multimedia, pp. 342–349 (2009)
19. Radunović, B., Le Boudec, J.Y.: A unified framework for max-min and min-max fairness with applications. IEEE/ACM Transactions on Networking **15**(5), 1073–1083 (2007)
20. Tschorsch, F., Scheuermann, B.: Tor is unfair – and what to do about it. In: The Proceedings of the IEEE 36th Conference on Local Computer Networks (LCN), 2011, pp. 432–440 (2011)
21. Vlavianos A, Iliofotou M, Faloutsos M.: BiToS: Enhancing BitTorrent for supporting streaming applications. In: Proceedings of the 25th IEEE INFOCOM, pp. 1–6 (2006)
22. Wu, D., Liang, C., Liu, Y., Ross, K.: View-upload decoupling: A redesign of multi-channel p2p video systems. In: Proceedings of the IEEE INFOCOM 2009, pp. 2726 – 2730 (2009)
23. Wu, D., Liang, Y., He, J., Hei, X.: Balancing performance and fairness in p2p live video systems. IEEE Transactions on Circuits and Systems for Video Technology **23**(6), 1029–1039 (2013)
24. Yan, Yonghe and El-Atawy, Adel and Al-Shaer, Ehab, Ranking-based optimal resource allocation in peer-to-peer networks. In Proceedings of the 26th IEEE INFOCOM, pp. 1100–1108 (2007)
25. Zhang, C., Dhungel, P., Wu, D., Liu, Z., Ross, K.: Bittorrent darknets. In: Proceedings of the IEEE INFOCOM 2010, pp. 1–9 (2010)
26. Zhang, X, Liu, J, Li, B, Yum, T.S.: CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In: Proceedings of the IEEE INFOCOM 2005, pp. 2102–2111 (2005)

Algorithm 4.1 *MaxMin-r*

```
1. Lower bound computation for the flows. Solve $MM_0$:
   $$\begin{aligned} &\max f, \\ &\text{s.t. } f(l_j^t, d_j) \geq f \qquad \forall (l_j^t, d_j) \in E_D. \end{aligned}$$
   Save the minimal flow value. Let $\phi := f$.
2. Maximal throughput computation. Solve the following LP problem, denoted by $MM_{\text{MaxFlow}}$:
   $$\begin{aligned} &\max \sum_{(l_j^t, d_j) \in E_D} f(l_j^t, d_j), \\ &\text{s.t. } f(l_j^t, d_j) \geq \phi \qquad \forall (l_j^t, d_j) \in E_D. \end{aligned}$$
   Save the LP optimum. Let $\sigma := \sum_{(l_j^t, d_j) \in E_D} f(l_j^t, d_j)$.
3. Initialization. Let $F := \emptyset$, $k := 1$, $E_1 := E_D, \forall (l_j^t, d_j) \in E_D : \ell_j^t := 0, \phi_0 = 0$.
4. LP solving. Solve the revised version of $mMM_k^{(1)}$:
   $$\begin{aligned} &\max f_k, \\ \text{s.t. } &\sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geq (1-\epsilon) \cdot \sigma \\ &f(l_j^t, d_j) \geq f_k \qquad \forall (l_j^t, d_j) \in E_k, \\ &f_k \geq \phi, \end{aligned}$$
   Save the LP optimum. Let $\phi_k := f_k$.
5. Presolve.
   $$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \mid \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{deg_k^-(d_j)} = \phi_k \right\},$$
   $$x_j^t := 0, \ \forall (l_j^t, d_j) \in E_{k_f}.$$
   If $|E_{k_f}| \neq 0$, go to Step 7.
6. MILP solving. Solve the McCormick reformulation of $mMM_k^{(2)}$:
   $$\begin{aligned} &\max \sum_{(l_j^t, d_j) \in E_k} x_j^t, \\ \text{s.t. } &\sum_{(l_j^t, d_j) \in E_k} p_j^t + \phi_k \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geq (1-\epsilon) \cdot \sigma, \\ &f(l_j^t, d_j) \geq \phi_k \qquad \forall (l_j^t, d_j) \in E_k, \\ &f(l_j^t, d_j) > \phi_k \, x_j^t \qquad \forall (l_j^t, d_j) \in E_k, \\ &\min\left(\delta_j \, x_j^t, \, f(l_j^t, d_j)\right) \geq p_j^t \qquad \forall (l_j^t, d_j) \in E_k, \\ &\max\left(0, \, f(l_j^t, d_j) - \delta_j \, (1 - x_j^t)\right) \leq p_j^t \qquad \forall (l_j^t, d_j) \in E_k, \end{aligned}$$
   where $x_j^t \in \{0, 1\}$ and $p_j^t = f(l_j^t, d_j) \, x_j^t$.
7. Fixing. Find the binding constraints for $\phi_k$, and fix the flow values of the adequate download edges:
   $$\Phi_k := \left\{ (l_j^t, d_j) \in E_k \mid x_j^t = 0 \right\},$$
   $$\ell_j^t := \phi_k, \ \forall (l_j^t, d_j) \in E_k \text{ where } x_j^t = 0,$$
   $$F := F \cup \Phi_k, \quad E_{k+1} := E_k \setminus \Phi_k.$$
8. Stopping criteria. If $F = E_D$, then stop. Otherwise, $k := k + 1$ and go back to Step 4.
```