Optimal File Distribution in Peer-to-Peer Networks

Kai-Simon Goetzmann[1,⋆], Tobias Harks[1], Max Klimm[1,⋆], and Konstantin Miller[2]

[1] Technische Universität Berlin, Institut für Mathematik
{goetzmann,harks,klimm}@math.tu-berlin.de
[2] Technische Universität Berlin, Telecommunication Networks Group
miller@tkn.tu-berlin.de

Abstract. We study the problem of distributing a file initially located at a server among a set of peers. Peers who downloaded the file can upload it to other peers. The server and the peers are connected to each other via a core network. The upload and download rates to and from the core are constrained by user and server specific upload and download capacities. Our objective is to minimize the makespan. We derive exact polynomial time algorithms for the case when upload and download capacities per peer and among peers are equal. We show that the problem becomes strongly NP-hard for equal upload and download capacities per peer that may differ among peers. For this case we devise a polynomial time $(1 + 2\sqrt{2})$-approximation algorithm. To the best of our knowledge, neither NP-hardness nor approximation algorithms were known before for this problem.

1 Introduction

In the past decade, the concept of data distribution based on peer-to-peer overlay networks has become increasingly popular. A significant fraction of Internet traffic is nowadays generated by peer-to-peer file sharing applications [3]. The key principle underlying peer-to-peer file sharing is that peers who downloaded parts of the file (chunks), start to assist the server in uploading them to other peers. A chunk is the smallest indivisible unit w.r.t. the download source, that is, a peer cannot download parts of a chunk from different sources. It is assumed that the server and the peers are connected via a core network (Internet). The upload and download rates to and from the core are constrained by peer and server specific upload and download capacities. The core itself is usually overprovisioned, thus, there are no capacity constraints present.

While in the past many algorithms and protocols have been studied in the literature for the important problem of optimizing the download process [1,9,10], a systematic performance evaluation of the proposed solutions with respect to *optimal* solutions has not received similar attention. In this paper, we address the fundamental problem of minimizing the total completion time (makespan) of distributing a file among a set of peers in a peer-to-peer network. Due to the intrinsic difficulty of the problem, in this work we restrict ourselves to the case of a single chunk only. To the best of our knowledge, even for this restricted case there are no algorithms with provable performance

⋆ Supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

T. Asano et al. (Eds.): ISAAC 2011, LNCS 7074, pp. 210–219, 2011.

guarantee known. While the case of multiple chunks still eludes us, we see our study as an important first step towards understanding the general peer-to-peer file distribution problem.

The Model. An instance of this problem is described by a tuple $I = (N, c^d, c^u)$, where $N = \{0, \ldots, n\}$ is the set of peers, $c^d = (c_0^d, \ldots, c_n^d) \in \mathbb{Q}_{\geq 0}^{n+1}$ is the vector of download capacities from the core network and $c^u = (c_0^u, \ldots, c_n^u) \in \mathbb{Q}_{\geq 0}^{n+1}$ is the vector of upload capacities to the core network. We will identify the server with peer 0 and assume that only the server initially owns a file of unit size, which is not divided into several chunks. See Figure 1 for an illustration. A feasible solution $S = (s_{i,j})_{i,j \in N}$ is a family of integrable functions $s_{i,j} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, $i, j \in N$, where $s_{i,j}(t)$ denotes the sending rate of peer i to peer j at time t. We require that every peer $i \in N$ receives its data from a unique peer, denoted by $p(i) \neq i$: $s_{k,i}(t) = 0$ for all $k \neq p(i)$ and all $t \in \mathbb{R}_{\geq 0}$. In addition, only peers that possess the file can send with a positive rate: $s_{i,j}(t) = 0$ for all $i, j \in N \setminus \{0\}$, $t \in \mathbb{R}_{\geq 0}$ with $\int_0^t \sum_{k \in N} s_{k,i}(\tau)\, d\tau < 1$. Finally, the sending rates have to obey download and upload capacity constraints: $\sum_{j \in N} s_{i,j}(t) \leq c_i^u$ for all $i \in N$, $t \in \mathbb{R}_{\geq 0}$ and $s_{p(j),j}(t) \leq c_j^d$ for all $j \in N$, $t \in \mathbb{R}_{\geq 0}$. We denote by $x_i(t) = \int_0^t s_{p(i),i}(\tau)\, d\tau$ the proportion of the file owned by peer $i \in N \setminus \{0\}$ at time t. For notational convenience, we set $x_0(t) = 1$ for all $t \in \mathbb{R}_{\geq 0}$. We denote by $C_i = \inf\{t \in \mathbb{R}_{\geq 0} : x_i(t) = 1\}$ the *completion time* of peer i. The makespan of a solution S is then defined as $M = \max_{i \in N \setminus \{0\}} C_i$. If an instance satisfies $c_i^u = c_i^d = c_j^u = c_j^d$ for all $i, j \in N \setminus \{0\}$ we speak of an instance with *homogeneous symmetric capacities*. If an instance satisfies $c_i^u = c_i^d$ for all $i \in N$ (but possibly $c_i^u \neq c_j^u$ for some $i, j \in N \setminus \{0\}$) we speak of *heterogeneous symmetric capacities*. In both cases, we only write $I = (N, c)$.

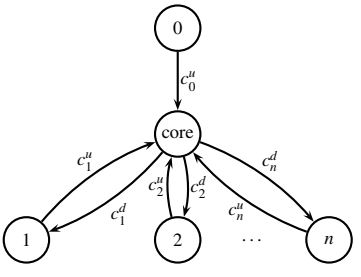

Fig. 1. Graphical representation of the file distribution problem we consider

Previous Work. There is an enormous body of work on the (minimum time) broadcasting problem, multicast problem, and gossiping problem, see [6] for a survey. In the broadcasting problem, the task is to disseminate a message from a source node to the rest of the nodes in a given communication network as fast as possible, see Ravi [15]. When the message needs to be disseminated only to a subset of the nodes, this task is referred to as multicasting, see Bar-Noy et al. [2] for approximation algorithms. In the gossiping problem, several nodes possess different messages and the goal is to transmit every message to every node.

The usual underlying communication model for these problems is known as the telephone model: a node may send a message to at most one other node in each round, and

it takes one unit of time (round) to pass a message. For complete graphs on n nodes, this process terminates in $\lceil \log_2(n+1) \rceil$ rounds. It is known that for arbitrary communication graphs, the problem of computing an optimal broadcast in the telephone model is NP-hard [5], even for 3-regular planar graphs [12].

Khuller et al. [7] study the problem of broadcasting in heterogeneous networks (see Bar-Noy et al. [2] for the corresponding multicast problem). They extend the telephone model by allowing the transmission time of a message to depend on the sender. They prove that it is NP-hard to minimize the makespan and present an approximation scheme.

Note that in contrast to both models, in our model a peer may upload the file to an arbitrary number of peers simultaneously provided the capacity constraints are satisfied. Moreover, the transfer time between any two peers is not given as part of the instance but determined by the sending rates of a feasible solution. In fact, the model by Khuller et al. [7] for broadcasting in heterogeneous networks reduces to a special case of our model. If $c_i^d \geq \max_{j \in N} c_j^u$ for all $i \in N$, there is always an optimal solution in which no peer will send the file to more than one other peer at a time, thus, the time to transfer the file from one peer to another only depends on the sender's upload capacity. In general, however, it may be beneficial to serve multiple peers simultaneously as illustrated in the following example. Peer 0 with capacity $c_0^u = 2$ initially owns the file, and there are two further peers with capacities $c_i^u = c_i^d = 1$, $i = 1, 2$. The optimal solution is $s_{0,1}(t) = s_{0,2}(t) = 1 \ \forall t \in [0, 1)$ with a makespan of $M^* = 1$. On the other hand, restricting peers to upload to at most one other peer at a time results in an optimal makespan of 2.

Mundinger et al. [13] studied a peer-to-peer file distribution problem, where a file is subdivided in multiple parts and the goal is to disseminate the complete file to every peer as fast as possible. The crucial difference to our model is that they assume that the upload capacity of a peer is equally shared among concurrent uploads. Under this fair sharing assumption they prove that for homogeneous capacities a simple greedy algorithm is optimal. For our model we prove a similar result (though for a single indivisible file) *without* the fair-sharing assumption. Our proof is quite involved since dropping the fair sharing assumption considerably complicates matters. For heterogeneous capacities, to the best of our knowledge, neither approximation algorithms nor hardness results were known before.

Also related are works on the so called fluid limit model, where the number of file parts tends to infinity [4,8,11,13,14].

Summary of the Results and Used Techniques. We first study the peer-to-peer file distribution problem for the case of homogeneous symmetric (unit) capacities. We show that a greedy algorithm computes an optimal solution with makespan $\lceil \log_2(n+1) \rceil$. Although similar results for the same algorithm have been obtained before, e.g. by Mundinger et al. [13], or in the broadcast literature, our result holds for a more general model (dropping the fair sharing assumption of [13] and dropping the telephone model). For the case of unit peer capacities and an arbitrary integer server capacity, we propose a polynomial time algorithm (that possibly splits up server capacity) and prove its optimality. We also give a closed-form expression of the minimal makespan. If peer capacities are heterogeneous and symmetric we show that the peer-to-peer file distribution

problem becomes strongly NP-hard. A key ingredient of the reduction (from 3-partition) is the *capacity expansion lemma* (Lemma 3.1) that provides an upper bound on the total amount of data downloaded at any point in time. In light of the hardness, we then study approximation algorithms. We first devise a polynomial time $2\sqrt{2}$-approximation algorithm for instances with heterogeneous, symmetric capacities in which the upload capacity of the server is larger than the download capacity of any other peer. For smaller server capacities a slight modification of our algorithm gives a $(1+2\sqrt{2})$-approximation. Our algorithm proceeds in two phases; in a first phase, we use a time-varying resource augmentation to construct a so called $\sqrt{2}$-*augmented solution* that violates the capacity constraints of the peers at any point in time by a factor of at most $\sqrt{2}$. By exploiting again our capacity expansion lemma, we prove that the makespan of the thus constructed augmented solution is at most a factor of 2 away from the optimal makespan of the original instance. We then rescale the relaxed solution to obtain a feasible solution with makespan less than a factor $2\sqrt{2}$ away from the optimal makespan.

2 Homogeneous Symmetric Capacities

In this section we consider the homogeneous symmetric setting, i.e. $c_i^u = c_i^d = c_i = 1$ for all $i \in N \setminus \{0\}$. The server has a capacity of $c_0^u = c$. For the simplest case, $c = 1$, we propose a Greedy procedure: At each point in time, any peer that already owns the file uploads it to exactly one other peer, which takes one unit of time. Thus in each step, the number of peers owning the file is doubled, resulting in a makespan of $\lceil \log_2(n+1) \rceil$.

Lemma 2.1. *If $c_i = 1$ for all $i \in N$,* Greedy *computes an optimal solution. The optimal makespan is $\lceil \log_2(n+1) \rceil$.*

To prove this, we consider an arbitrary solution and modify it in such a way that in the time interval $[0, 1)$ only one peer is served, without increasing the makespan. Iterating this argument for later points in time and other peers proves that there is an optimal solution with the Greedy structure. More specifically, let peer 1 be a peer that finishes its download first in the given solution. We serve this peer in $[0, 1)$ and change the sending rates from the server to the other peers in such a way that the fraction of the file they own at time C_1 is the same as in the given solution. This can be done without violating the capacities, and obviously the makespan is not increased. A detailed proof will be included in the full version of the paper.

We now consider the case where c is an arbitrary integer, and start with a Lemma stating that in this case there always is an optimal solution that uses *fair-sharing*, i.e. whenever peer 0 serves several peers simultaneously, all of them are served with equal rate. The proof is quite involved, and the integrality condition on c is crucial. To see this, consider an example with $c = 3/2$ and $n = 4$. In an optimal solution, the server serves peer 1 in $[0, 1)$ and peer 2 in $[1, 2)$ with a rate of 1 each and peer 3 with a rate of $1/2$ in $[0, 2)$. Peer 4 is served by peer 1 in $[1, 2)$, resulting in a makespan of $M^* = 2$. The best solution that uses fair-sharing, however, has a makespan of $M = 7/3$: The server serves peers 1 and 2 in $[0, 4/3)$ with a rate of $3/4$ each, and peer 3 and 4 are both served at a rate of 1 in $[4/3, 7/3)$ by the server and peer 1.

Lemma 2.2. *For any instance $I = (N, (c_i)_{i\in N})$ with $c_0 = c \in \mathbb{N}$ and $c_i = 1\ \forall\ i = 1, \ldots, n$, there always exists an optimal solution that uses* fair sharing. *Also, the sending rates $s_{i,j}$ are piecewise constant with discontinuities only at points in time where some peer finishes its download.*

[*Proof will be included in the full version of the paper.*]
In the same setting, we also know that the server always serves groups of c peers jointly, except in the beginning, where a group of at least c and most $2c$ peers is served.

Lemma 2.3. *For any instance $I = (N, (c_i)_{i\in N})$ with $c_0 = c \in \mathbb{N}$ and $c_i = 1\ \forall\ i = 1, \ldots, n$, there always exists an optimal solution where only at the beginning more than c peers are served simultaneously.*

[*Proof will be included in the full version of the paper.*]
Summing up, we know so far that there always is an optimal solution where peer 0 at the beginning serves a set of peers N_1, $c \le |N_1| < 2c$, in $|N_1|\,/\,c$ time units, then some sets $N_2, \ldots, N_{k-1}$ with $|N_j| = c\ \forall j \ge 2$, in one time unit each, and finally a set N_k containing at most c peers, in another unit of time.

We are now ready to present an exact algorithm for this special case and prove its correctness. The algorithm is an extended Greedy procedure: Set $h = \lceil \log_2(n/c + 1)\rceil$. If $n < c(2^h - 1 + 2^{h-1})$, let $|N_1| = \lceil (n - c(2^{h-1} - 1))/2^{h-1}\rceil$ and $k = h$, otherwise let $|N_1| = c$ and $k = h + 1$. Let $|N_j| = c$ for $j = 2, \ldots, k$. The server serves N_1 in $[0, |N_1|/c)$ and N_j in $[|N_1|/c + j - 2, |N_1|/c + j - 2)$. Meanwhile, any peer that finishes its download starts serving other peers greedily.

Theorem 2.4. *For any instance $I = (N, (c_i)_{i\in N})$ with $c_0 = c \in \mathbb{N}$ and $c_i = 1\ \forall\ i = 1, \ldots, n$, the extended* Greedy *yields an optimal solution. The optimal makespan is*

$$M^* = \begin{cases} h - 1 + \frac{1}{c}\left\lceil \frac{n - c(2^{h-1}-1)}{2^{h-1}} \right\rceil & \text{if } n \in [c(2^h - 1), c(2^h - 1 + 2^{h-1})) \\ h + 1 & \text{if } n \in [c(2^h - 1 + 2^{h-1}), c(2^{h+1} - 1)) \,, \end{cases}$$

where $h = \lceil \log_2(n/c + 1)\rceil$.

Proof. For $j = 1, \ldots, k$, we denote by $\tilde{N}_j$ the set of peers that are directly or indirectly served by a peer in N_j:

$$\tilde{N}_j = \{i \in N : p^\ell(i) = i' \text{ for some } i' \in N_j, \ell \in \mathbb{N}_0\}$$

Here $p^\ell(j)$ means that the function p is applied ℓ times to j. W.l.o.g. we can assume that in the solution produced by our algorithm $|\tilde{N}_j| = c \cdot 2^{k-j}$ for all $j = 2, \ldots, k$, i.e. all sets $\tilde{N}_j$ for $j \ge 2$ are as big as possible. If this is not the case we can move peers from $\tilde{N}_1$ to these sets. The lemmas above state that there also is an optimal solution with this structure and $c \le |N_1| < 2c$. In the argumentation below, we therefore restrict to solutions of this structure.

The integer h is chosen such that $c(2^h - 1) \le n < 2^{h+1} - 1$. We first consider the case where $n \ge c(2^h - 1 + 2^{h-1})$. Our algorithm sets $k = h + 1$ and $|N_1| = c$. The maximum number of peers that can be served like this is $c(2^{h+1} - 1) > n$, so indeed all peers are served within a makespan of $h + 1$. Assume that in an optimal solution

only $k = h$ sets are served by the server. The maximum number of peers served in this way is $|N_1| \cdot 2^{h-1} + c(2^{h-1} - 1) < c(2^h + 2^{h-1} - 1) \le n$, so not all peers can be served, contradiction! Therefore in the optimal solution it must hold that $k = h+1$ and $|N_1| \ge c$, proving that the solution produced by our algorithm is optimal.

If $n < c(2^h - 1 + 2^{h-1})$, the size of N_1 in an optimal solution (with $|\tilde{N}_j| = c \cdot 2^{k-j}$ for $j \ge 2$) has to be chosen such that $|\tilde{N}_1| = n - c(2^{h-1} - 1) \le |N_1| \cdot 2^{h-1}$, i.e.

$$|N_1| = \min\left\{\ell \in \mathbb{N} : \ell \ge \frac{n - c(2^{h-1} - 1)}{2^{h-1}}\right\} = \left\lceil \frac{n - c(2^{h-1} - 1)}{2^{h-1}} \right\rceil,$$

which is exactly how the algorithm chooses N_1. It is obvious that choosing $k \le h - 1$ does not lead to a solution where all peers are served, and choosing $k \ge h + 1$ leads to a solution with a makespan of at least $h + 1$. Since the solution produced by our algorithm has a makespan of $t_1 + k - 1 \le h + 1$ this proves the correctness of our algorithm. □

3 Heterogeneous Symmetric Capacities

In this section, we consider the case of heterogenous and symmetric capacities. First, we show that the peer-to-peer file distribution for heterogenous symmetric peers is strongly NP-hard. Then, we devise an algorithm that approximates the optimal makespan by a factor $1 + 2\sqrt{2}$. Our hardness and approximation results rely on a useful lemma that bounds the work done in any feasible solution. For a feasible solution, let $u_i(t_1, t_2)$ and $z_i(t_1, t_2)$ denote the total upload and the idleness of peer i in time interval $[t_1, t_2]$, defined as $u_i(t_1, t_2) = \int_{t_1}^{t_2} \sum_{j \in N} s_{i,j}(\tau)\, d\tau$ and $z_i(t_1, t_2) = \int_{\max\{t_1, C_i\}}^{t_2} (c_i - \sum_{j \in N} s_{i,j}(\tau))\, d\tau$. For $t \ge 0$, we define $X(t) = \sum_{i \in N} x_i(t)$ and $Z(t) = \sum_{i \in N} z_i(0, t)$.

Lemma 3.1 (Capacity Expansion Lemma). *Let $I = (N, (c_i)_{i \in N})$ be an instance of the peer-to-peer file distribution problem with $c = \max_{i \in N} c_i$. Then, for all solutions S of I, the following two hold:*

1. $u_i(t_1, t_2) + z_i(t_1, t_2) \le \max\left\{0, \left(t_2 - t_1 - \frac{1 - x_i(t_1)}{c_i}\right) c_i\right\} = \max\left\{0, (t_2 - t_1)\, c_i - 1 + x_i(t_1)\right\}$ *for all peers $i \in N$ and all times $0 \le t_1 < t_2$.*
2. $X(k/c) + Z(k/c) \le 2^k$ *for all $k \in \mathbb{N}$. This inequality is strict if there is $i \in N$ with $c_i < c$ and $0 < C_i \le k/c$.*

Proof. To see 1., note that for any peer i with $C_i \le t_1$, the upload rate (integrand of the total upload) and the idle rate (integrand of the idleness) sum up to c_i for all $t \in [t_1, t_2]$, thus $u_i(t_1, t_2) + z_i(t_1, t_2) \le (t_2 - t_1)\, c_i$. If $x_i(t_1) < 1$, peer i needs at least $(1 - x_i(t_1))/c_i$ time units to finish its download, thus only a time interval of length $t_2 - t_1 - (1 - x_i(t_1))/c_i$ remains for the upload and the claimed inequality follows.

We prove 2. by induction over k. The inequality is trivial for $k = 0$. So, let us assume that for $k \in \mathbb{N}$, we have $X((k-1)/c) + Z((k-1)/c) \le 2^{k-1}$. Using 1., we obtain the inequality

$$X(k/c) - X((k-1)/c) + Z(k/c) - Z((k-1)/c) = \sum_{i \in N} u_i((k-1)/c), k/c) + z_i((k-1)/c), k/c)$$

$$\le \sum_{i \in N} \max\{0, c_i/c - 1 + x_i((k-1)/c)\} \le X((k-1)/c),$$

where we use $c_i \leq c$. Note that the latter inequality is satisfied strictly if there is a peer i with $C_i \leq k/c$ and $c_i < c$. Rearranging terms and using the induction hypothesis, we obtain $X(k/c) + Z(k/c) \leq 2X((k-1)/c) + Z((k-1)/c) \leq 2^k$, as claimed. □

Hardness. We are now ready to prove strong NP-hardness of the peer-to-peer file distribution problem.

Theorem 3.2. *The peer-to-peer file distribution problem for heterogeneous symmetric peers is strongly NP-hard.*

We reduce from 3-Partition, where a multiset $P = \{k_1, \ldots, k_n\}$ of $n = 3m$ numbers has to be partitioned into sets $P_0, \ldots, P_{m-1}$ such that $\sum_{k_i \in P_j} k_i = B$ for all $j = 0, \ldots, m-1$. W.l.o.g. we can assume $m = 2^\ell$. The idea is to introduce *m subset peers* with capacity B and *n master element peers* with capacity k_i. If we have a yes-instance of 3-Partition, we can first serve all subset peers, and then each subset peer serves those master element peers that correspond to elements in one of the sets P_j with full capacity, resulting in a makespan of $\ell + \ell/B$. To show that for no-instances the makespan is strictly greater than $\ell + \ell/B$, we have to introduce $2\ell k_i - 2$ *element peers* for all $k_i \in P$. This is also the point where the capacity expansion lemma is used. The proof of the hardness result is quite involved and will be included in the full version of the paper.

Approximation. In this section, we devise an algorithm that runs in time $O(n \log n)$ and computes a solution with makespan no larger than $(1 + 2\sqrt{2})M^*$, where M^* is the optimal makespan. We first consider the case $c_0 \geq c_i$ for all $i = 1, \ldots, n$, for which we will show a $2\sqrt{2}$-approximation. Then, we will use this result to obtain a $(1 + 2\sqrt{2})$-approximation for arbitrary server capacities. Before we present details of the algorithm we give a high-level picture of our approach. The intrinsic difficulty in proving any approximation bound is to obtain lower bounds on the optimal makespan. Let us recall the inequality shown in Lemma 3.1: for *any* solution, peer i's contribution to the upload in time interval $[t_1, t_2]$ is bounded by $u_i(t_1, t_2) \leq \max\{0, c_i(t_2 - t_1) - 1 + x_i(t_1)\}$. To exploit this capacity bound, our algorithm should fulfill two properties: it should finish peers with large capacity as soon as possible and it should avoid idle time as long as possible. To achieve exactly this, we use the concept of *time-varying resource augmentation*. We call a possibly infeasible solution $\tilde{S}$ a $\sqrt{2}$*-augmented* solution if at any point in time the original peer capacities are not exceeded by more than a factor of $\sqrt{2}$, i.e. $\sum_{j \in N} s_{i,j}(t) \leq \sqrt{2}c_i^u$ and $s_{p(i),i}(t) \leq \sqrt{2}c_i^d$ for all $i \in N$, $t \in \mathbb{R}_{\geq 0}$. We will show that there is an efficiently computable $\sqrt{2}$*-augmented* solution that satisfies the above mentioned properties. This allows to apply the capacity expansion lemma, and we get that the makespan $\tilde{M}$ of $\tilde{S}$ is not larger than $2M^*$. Rescaling the augmented solution $\tilde{S}$ to obtain a feasible solution S we get an additional factor of $\sqrt{2}$.

Now, we explain our algorithm (termed Scale-Fit) in more detail. Let the peers be labeled such that $c_0 \geq \cdots \geq c_n$. We choose an index k such that

$$c_0/\sqrt{2} \leq \sum_{j=1}^{k} c_j \leq \sqrt{2}c_0 \tag{1}$$

is satisfied, and assign peers $1, \ldots, k$ to peer 0. Note that there always exists such an index k unless peer 0 can serve all peers simultaneously at full download rate. This is

formally proven in the following lemma, which we give for the slightly weaker assumption $c_0 \geq c_1/\sqrt{2}$.

Lemma 3.3. *Let $c_1 \geq \cdots \geq c_n$ and let $c_0 \geq c_1/\sqrt{2}$. If $\sum_{j=1}^{n} c_j > c_0$, then there is $k < n$ such that $c_0/\sqrt{2} \leq \sum_{j=1}^{k} c_j \leq \sqrt{2}c_0$.*

Proof. If $c_0/\sqrt{2} \leq c_1$, there is nothing left to show. Otherwise, let $k' = \max\{\ell \in \{1,\ldots,n\} : \sum_{j=1}^{\ell} c_j < c_0/\sqrt{2}\}$. Since we assume $c_1 < c_0/\sqrt{2}$ and $\sum_{j=1}^{n} c_j > c_0$, we have $1 \leq k' \leq n-1$. We set $k = k'+1$. By definition, $\sum_{j=1}^{k} c_j \geq c_0/\sqrt{2}$. In addition, we have $\sum_{j=1}^{k} c_j \leq 2\sum_{j=1}^{k'} c_j < \sqrt{2}c_0$. □

Given the choice of k according to (1), we now explain the scaling of the capacities. We distinguish two cases. If the total capacity of the downloading peers $1,\ldots,k$ exceeds the capacity of the uploading peer 0, we augment the upload capacity to match the total download capacity. Otherwise the downloaders' capacities are scaled by a common factor in order to fit the uploader's capacity. Formally, the capacities of peers $1,\ldots,k$ are increased by a factor of $\alpha = \max\{1, c_0/\sum_{j=1}^{k} c_j\}$ and that of peer 0 by a factor of $\beta = \sum_{j=1}^{k} \alpha c_j/c_0$. Note that by (1) the scaling factor is not greater than $\sqrt{2}$ in either case.

At time 0, all k peers start downloading from peer 0 with their full (possibly augmented) capacity. Whenever a peer finishes its download, the algorithm rescales the augmented capacity (either that of the downloader or that of the uploader) to the original level. After rescaling, we proceed serving new peers by the respective unused capacities of the parent and the finished peer according to (1), breaking ties arbitrarily. The process stops as soon as the total capacity of the remaining peers is less than the currently available upload capacity. From this point on, all remaining peers are served simultaneously at full download rate (this last step takes at most M^* time). The choice of the augmentation factors and the rescaling procedure ensures the invariant that the augmented capacities do not exceed the original capacities by a factor of $\sqrt{2}$. A formal proof of this fact will be included in the full version of this paper.

Before we formally analyse the performance of Scale-Fit, let us illustrate the algorithm by an example:

Example 3.4. Consider an instance with 6 peers and capacities $c_0 = 5$, $c_1 = 3$, $c_2 = 3$, $c_3 = 5/2$, $c_4 = 2$, and $c_5 = 2$. We first describe how to construct the augmented solution $\tilde{S}$. At time 0, the upload capacity of peer 0 is augmented by a factor of $6/5$ in order to serve peers 1 and 2 at their full download capacities. These downloads are completed at time $1/3$. At that time, the rescaled upload capacities of the parent of peers 1 and 2 (peer 0) can be used separately to serve further peers. The $5/2$ units of capacity (previously assigned to peer 1) are now assigned to peer 3 that downloads with full capacity without any augmentation. The remaining $5/2$ units of capacity are assigned to peer 4, whose download capacity is augmented by a factor of $5/4$. At the same time, peer 1 starts serving peer 5 without any augmentation because peer 5 is the only remaining peer. The highest augmentation factor is $5/4$. Thus, we rescale all rates by $4/5$ and obtain the feasible solution S.

We now turn to the approximation guarantee of our algorithm. For this, let $\tilde{S}$ be the $\sqrt{2}$-augmented solution generated by Scale-Fit, and let T_0 denote the first point in time

in this solution when a peer does not fully use its upload capacity, that is, $T_0 = \min\{t \geq 0 : \exists\, i \in N \text{ with } \tilde{C}_i \leq t \text{ and } \sum_{j \in N} \tilde{s}_{i,j}(t) < c_i\}$. We first show that T_0 is a lower bound on the optimal makespan.

Lemma 3.5. *Let I be an instance of the peer-to-peer file distribution problem with $c_0 \geq c_i\ \forall i = 1, \ldots, n$ and let $\tilde{S}$ be the $\sqrt{2}$-augmented solution generated by* Scale-Fit. *Then $T_0 \leq M^*$, where M^* is the optimal makespan.*

Proof. Let us first consider the case $T_0 < \tilde{M}$. For a contradiction, suppose $M^* < T_0$ and fix an optimal solution S^*. We have $\tilde{X}(T_0) < X^*(T_0) = n$. Let k_0 be the peer with smallest index that has not finished the download at time T_0, that is, $k_0 = \min\{i \in N : \tilde{C}_i > T_0\}$. Note that there is such peer since we assume $T_0 < \tilde{M}$. If $k_0 = 1$, then we obtain the inequality $T_0 < 1/c_1$, which is a lower bound on M^* and there is nothing left to show. So we assume $k_0 > 1$ and consider the point in time $T_1 = T_0 - 1/c_{k_0}$. For a contradiction, let us assume that $\tilde{X}(T_1) \geq X^*(T_1)$.

For the $\sqrt{2}$-augmented solution $\tilde{S}$ returned by Scale-Fit we have $\tilde{x}_{k_0}(T_1) = 0$, and since peers start downloading in order we also get $\tilde{x}_i(T_1) = 0$ for all $i \geq k_0$. By construction, every peer $i \in N$ with $\tilde{x}_i(T_1) > 0$ receives data with download rate $d_i = \alpha c_i \geq c_i$ until $\tilde{C}_i$. Using $\tilde{z}_i(T_1, T_0) = 0$, we obtain

$$\tilde{u}_i(T_1, T_0) \geq \left(\frac{1}{c_{k_0}} - \frac{1 - \tilde{x}_i(T_1)}{d_i}\right) c_i = \frac{c_i}{c_{k_0}} + \frac{c_i}{d_i}(\tilde{x}(T_1) - 1) \geq \frac{c_i}{c_{k_0}} + \tilde{x}_i(T_1) - 1$$

for all $i \leq k_0$ with $\tilde{x}_i(T_1) > 0$ and $u_i(T_1, T_0) \geq 0$ for all other peers. Referring to Lemma 3.1 (1.), we calculate

$$\begin{aligned} X^*(T_0) - X^*(T_1) &\leq \sum_{i \in N} \max\left\{0, \frac{c_i}{c_{k_0}} - 1 + x_i^*(T_1)\right\} = \sum_{i \in N : x_i^*(T_1) > 1 - c_i/c_{k_0}} \left(\frac{c_i}{c_{k_0}} - 1 + x_i^*(T_1)\right) \\ &\leq \sum_{i < k_0 : x^*(T_1) > 1 - c_i/c_{k_0}} \left(\frac{c_i}{c_{k_0}} - 1\right) + X^*(T_1), \end{aligned}$$

where we use $c_i/c_{k_0} \leq 1\ \forall i \geq k_0$. We get $X^*(T_0) - X^*(T_1) \leq \sum_{i<k_0}(c_i\,/\,c_{k_0} - 1) + \tilde{X}(T_1) \leq \tilde{X}(T_0) - \tilde{X}(T_1)$, a contradiction. We conclude $X^*(T_1) > \tilde{X}(T_1)$. Applying the same line of argumentation for T_1 instead of T_0, we derive the existence of $T_2 = T_1 - 1/c_{k_1}$ for some $k_1 \in N$ with $X^*(T_2) > \tilde{X}(T_1)$. We can iterate this argument until we reach T_ℓ, with the property that $X^*(T_\ell) > \tilde{X}(T_\ell)$ and $k_\ell = \min\{i \in N : \tilde{C}_i > T_\ell\} = 1$. This is a contradiction since in the time interval $[0, T_\ell]$ only the server can upload and its upload rate in $\tilde{S}$ is not smaller than that in S^*. We conclude that our initial assumption that $T_0 > M^*$ was wrong, finishing the proof for the case $T_0 < \tilde{M}$. If $T_0 = \tilde{M}$, we can use the same line of argumentation for $T_0 - \epsilon$ instead of T_0, where $\epsilon > 0$ is arbitrary. Thus we obtain $T_0 \leq M^*$ also for that case. □

We are now ready to prove the approximation guarantee of Scale-Fit.

Theorem 3.6. *For the peer-to-peer file distribution problem with heterogeneous symmetric peers, the following holds:*

1. *If* $c_0 \geq c_1$, Scale-Fit *is a* $2\sqrt{2}$*-approximation.*
2. *If* $c_0 < c_1$, *uploading the file to the peer* 1 *and applying* Scale-Fit *is a* $(1 + 2\sqrt{2})$*-approximation.*

Proof. We first show 1. By construction, peer n determines the makespan and starts its download not after T_0. Hence, the makespan of the $\sqrt{2}$-augmented solution $\tilde{S}$ is not larger than $T_0 + 1/c_n \leq 2M^*$ where we use Lemma 3.5 and the fact that $1/c_n$ is a lower bound on the optimal makespan. For rational input, the largest factor with which a capacity constraint is violated is strictly smaller than $\sqrt{2}$. Dividing all sending rates by that factor, we obtain a feasible solution with makespan smaller than $2\sqrt{2}M^*$.

To see 2., note that the server needs $1/c_0 \leq M^*$ time units to transfer the file to peer 1. We then treat the instance as an instance where peer 1 is the server, i.e. we do not let peer 0 upload the file to any other peer except peer 1. Using the same arguments as in the proof of Lemma 3.5, we derive that this takes at most $2\sqrt{2}M^*$ additional time units, implying the claimed approximation factor. □

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surveys 36, 335–371 (2004)
2. Bar-Noy, A., Guha, S., Naor, J., Schieber, B.: Message multicasting in heterogeneous networks. SIAM J. Comput. 30(2), 347–358 (2000)
3. Cho, K., Fukuda, K., Esaki, H., Kato, A.: The impact and implications of the growth in residential user-to-user traffic. In: Proc. ACM SIGCOMM Conf. Applications, Technologies, Architectures and Protocols for Computer Comm. (2006)
4. Ezovski, G., Tang, A., Andrew, L.: Minimizing average finish time in P2P networks. In: Proc. 30th IEEE Internat. Conf. Computer Comm., INFOCOM (2009)
5. Garey, M., Johnson, D.: Computers and Intractability (1979)
6. Hedetniemi, S.T., Hedetniemi, S.M., Liestman, A.: A survey of gossiping and broadcasting in communication networks. Networks 18, 129–134 (1998)
7. Khuller, S., Kim, Y.-A.: Broadcasting in heterogeneous networks. Algorithmica 48(1), 1–21 (2007)
8. Kumar, R., Ross, K.: Peer assisted file distribution: The minimum distribution time. In: Proc. 1st IEEE Workshop on Hot Topics in Web Syst. and Technologies (2006)
9. Li, J.: On peer-to-peer (P2P) content delivery. Peer-to-Peer Netw. Appl. 1, 45–63 (2008)
10. Lua, E., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. IEEE Comm. Surveys and Tutorials 7, 72–93 (2005)
11. Mehyar, M., Gu, W., Low, S., Effros, M., Ho, T.: Optimal strategies for efficient peer-to-peer file sharing. In: Proc. IEEE Internat. Conf. on Acoustics Speech and Signal Proc., ICASSP (2007)
12. Middendorf, M.: Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2. Inf. Process. Lett. 46(6), 281–287 (1993)
13. Mundinger, J., Weber, R., Weiss, G.: Optimal scheduling of peer-to-peer file dissemination. J. of Scheduling 11, 105–120 (2008)
14. Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: Proc. ACM SIGCOMM Conf. Applications, Technologies, Architectures and Protocols for Computer Comm. (2004)
15. Ravi, R.: Rapid rumor ramification: Approximating the minimum broadcast time. In: Proc. 35th Annual IEEE Sympos. Foundations Comput. Sci., pp. 202–213 (1994)