Unsplittable max-min demand allocation - a routing problem

Nilsson, Pål; Pioro, Michal

2005

Unsplittable max-min demand allocation – a routing problem

Pål Nilsson[1] and Michał Pióro[1,2]
[1]Department of Communication Systems, LTH, Lund University
Box 118 SE-221 00, Lund, Sweden
[2]Institute of Telecommunications, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warszawa, Poland
Email: {paln,mpp}@telecom.lth.se

Abstract The end-to-end assignment of bandwidth to node-pairs (demands) in a communication network can be considered fair if it is distributed according to the max-min fair (MMF) principle. This paper investigates the problem of obtaining an MMF allocation if each demand is required to use exactly one path (i.e., to use unsplittable flows). First it is shown that the problem is $\mathcal{NP}$-hard, both if each demand may use an arbitrary path and also if each demand is restricted to use a path from a small, predefined (demand-specific) path-list. Then, a number of mixed integer programming models are presented for the problem. These models constitute a basis for resolution techniques and are therefore examined in terms of computation times on a set of randomly generated problem instances.

1 Introduction

In this paper a routing problem involving max-min fair allocation of bandwidth to demands in a communication network is dealt with. The considered problem has been studied in its simplified form already in [1], and has also, as a convex optimization problem, been solved for bifurcated (splittable) flows [8, 12, 11]. This paper addresses a more difficult version when only non-bifurcated (unsplittable) flows are allowed. Such an assumption, also called requirement of single-path flows, results in non-convex problem formulations which are inherently hard [5] (the hardness was pointed out already in [7]). Unsplittable flows is often a realistic restriction due to the used routing protocol, or simply an explicit management requirement, stipulating avoidance of packet resequencing in receiving nodes. With the network given in terms of topology and link capacities, the following traffic engineering problem is identified: the demand between each source-destination (S-D) node-pair must be associated with a single path such that a sufficient volume of flow can be routed on demands' single paths simultaneously, without exceeding the link capacities. By "sufficient volume" a volume that is equitable among different S-D pairs is addressed, i.e., a fair sharing of resources. Particularly, we study the problem of assigning a single-path flow to each demand, such that the flow distribution is Max-Min Fair (MMF). Consequently, once each S-D pair is assigned a single path, the problem is reduced to max-min fair sharing of corresponding link capacities, for which an efficient polynomial time algorithm exists [1]. Thus essentially, the considered problem amounts to the very hard task of appropriate path selection.

1.1 Notation

The used notions will be defined in terms of graph theory, where a graph, $G = (V, E)$, represents a network, with the vertex set, V, modeling nodes and the set of edges, E, modeling links. A set of vertex-pairs called demands, D, is assumed to be given. A demand is a requirement for communication between a node-pair in the network. For each demand d, $d \in D$, it will sometimes be convenient to predefine a set of paths, P_d, where each element p, $p \in P_d$, defines a cycle-free set of edges that connects vertex-pair d. Let elements of the index sets $\{1, \ldots, |E|\}$, $\{1, \ldots, |D|\}$ and $\{1, \ldots, |P_d|\}$ uniquely identify elements of the sets E, D and P_d respectively. As there will be no risk of ambiguity we will slightly abuse notation, indexing on e, d, and p, as if they were elements of these index sets. Each edge e, $e \in E$, is associated with a capacity, c_e. A binary indicator, δ_{edp}, is used for the path-edge incidence relation; let $\delta_{edp} = 1$ if $e \in p$ and $p \in P_d$, and $\delta_{edp} = 0$ otherwise. The flow allocated to demand d will be identified by X_d, and x_{dp} is used to denote its part that is allocated to path p, i.e., $\sum_{p \in P_d} x_{dp} = X_d$. Let $\boldsymbol{x} = (X_1, X_2, \ldots, X_{|D|})$ denote the vector of all flows (also denoted the allocation

vector), and $\vec{\boldsymbol{x}}$ be the allocation vector sorted in non-decreasing order. The sorted allocation vector $\vec{\boldsymbol{x}}$ is said to be lexicographically greater than the sorted allocation vector $\vec{\boldsymbol{y}}$, $\vec{\boldsymbol{x}} \succ \vec{\boldsymbol{y}}$, if the first non-zero entry of $\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}}$ is positive. Consequently, $\vec{\boldsymbol{x}} \succeq \vec{\boldsymbol{y}}$ means $\vec{\boldsymbol{x}} \succ \vec{\boldsymbol{y}}$ or $\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}} = \mathbf{0}$, and $\vec{\boldsymbol{x}}^* = \text{lex}\max\{\vec{\boldsymbol{x}} : \boldsymbol{x} \in Q\}$ that $\vec{\boldsymbol{x}}^* \succeq \vec{\boldsymbol{x}}$ for all $\boldsymbol{x} \in Q$.

1.2 Problem Description

This study concerns the problem of obtaining MMF single-path (i.e., unsplittable) flows in a capacitated network. In an MMF allocation of flows each demand is assigned a flow and a routing such that it holds for the sorted allocation vector that an entry can be increased only at the cost of decreasing a previous entry, or by making the allocation vector infeasible. It can be shown that obtaining an allocation vector, $\boldsymbol{x} = (X_1, X_2, \ldots, X_{|D|})$, with this characteristic is equivalent to solving

$$\text{lex max } \vec{\boldsymbol{x}};\ \boldsymbol{x} \in Q, \tag{1}$$

where Q is the set of feasible solutions [11]. A solution $\boldsymbol{x} \geq \mathbf{0}$ is considered feasible, $\boldsymbol{x} \in Q$, if

(i) $\sum_{d \in D} \sum_{p \in P_d} \delta_{edp} x_{dp} \leq c_e$, for all $e \in E$ and

(ii) for all $d \in D$, $x_{dp'} = X_d$ for some path p', $p' \in P_d$, and $x_{dp} = 0$ for all other paths p, $p \in P_d,\ p \neq p'$,

i.e., if the sum of flows on a link does not exceed the link's capacity, and no demand has flows on more than one path.

2 Computational complexity

In this section it will be shown that solving (1) is $\mathcal{NP}$-hard. This is accomplished by proving that the corresponding decision problem is $\mathcal{NP}$-complete. It will be convenient to divide the problem into two cases: the case when a demand may use any one path to connect its node-pair – the *unlimited path-sets case*, and the case when a demand may select a path only from a limited, predefined set of paths – the *limited path-sets case*. A somewhat different version of the former decision problem was indeed proven $\mathcal{NP}$-complete in [6].

2.1 Unlimited path-sets

When a demand may use any (simple) path that connects its node-pair, there is clearly no reason to predefine its path-set, as it is directly implied by $G = (V, E)$. This is the case for all demands if path-sets are unlimited. We will refer to the decision problem corresponding to the optimization problem of obtaining a maximized minimal flow on single paths with unlimited path-sets as MAXIMIZING MINIMAL SINGLE-PATH FLOW (MMSPF). If the set of feasible solutions, Q, admits usage of any path for each demand, such an optimization problem is indeed equivalent to finding a maximal first entry of $\vec{\boldsymbol{x}}$, $\boldsymbol{x} \in Q$, i.e., certainly a subproblem of (1).

MAXIMIZING MINIMAL SINGLE-PATH FLOW
INSTANCE: Graph $G = (V, E)$, an edge capacity c_e for each e, $e \in E$, a set of demands (vertex-pairs) D, and a number $K > 0$.
QUESTION: Is there a set of simple paths, containing exactly one path for each demand d, $d \in D$, such that for each demand d it is possible to assign a flow X_d, with $X_d \geq K$, on the path corresponding to demand d, without violating any edge capacity c_e, $e \in E$?

It is a straightforward exercise to show $\mathcal{NP}$-completeness of MMSPF, e.g. by transforming it to EDGE-DISJOINT PATHS [9]. The main problem – lexicographically maximizing $\vec{\boldsymbol{x}};\ \boldsymbol{x} \in Q$ (not only considering the MMSPF subproblem), corresponds to the following decision problem denoted LEX MAX SINGLE PATH FLOWS (LMSPF):

LEX MAX SINGLE PATH FLOWS
INSTANCE: Graph $G = (V, E)$, an edge capacity c_e for each e, $e \in E$, a set of demands (vertex-pairs) D, and

a target vector, $\boldsymbol{y}$, of length $|D|$.
QUESTION: Is there a set of simple paths, containing exactly one path for each demand d, $d \in D$, such that for each demand d it is possible to assign a flow X_d on the path corresponding to demand d, without violating any edge capacity c_e, $e \in E$, and such that if $\boldsymbol{x} = [X_d]_{d \in D}$, then $\vec{\boldsymbol{x}} \succeq \vec{\boldsymbol{y}}$?

Proposition 1 *LMSPF is $\mathcal{NP}$-complete.*

Proof. As MMSPF is a subproblem of LMSPF, the same transformation is valid here. □

It is possible to prove this fact exploiting another involved difficulty, namely packing. This can be done with a transformation from PARTITION, as will be shown in the following section.

2.2 Limited path-sets

The previous section had the assumption that the selection of a single path could be made from all possible paths for a demand. In many cases such an assumption might be too generous and not practical. For these cases a more realistic assumption would be that each demand is given with a predefined, bounded (in terms of cardinality) path-set. We will assume such a setting in this section. Clearly, if the sizes of the predefined path-sets are very large, this will differ only academically from the unlimited path-sets case. However, in practice the number of admissible paths will be substantially smaller than the number of all possible paths. We will show that LMSPF with predefined path-sets is $\mathcal{NP}$-complete. The cardinality of the predefined path-sets will be limited to 2, resulting in the decision problem LEX MAX SINGLE PATH FLOWS-2 (LMFSP2):

LEX MAX SINGLE PATH FLOWS-2
INSTANCE: Graph $G = (V, E)$, an edge capacity c_e for each e, $e \in E$, a set of demands (vertex-pairs) D, two simple paths for each demand, and a target vector, $\boldsymbol{y}$, of length $|D|$.
QUESTION: Is there a set of paths, containing exactly one of the two admissible paths for each demand d, $d \in D$, such that for each demand d it is possible to assign a flow X_d on the path corresponding to demand d, without violating any edge capacity c_e, $e \in E$, and such that if $\boldsymbol{x} = [X_d]_{d \in D}$, then $\vec{\boldsymbol{x}} \succeq \vec{\boldsymbol{y}}$?

Proposition 2 *LMSPF2 is $\mathcal{NP}$-complete.*

Proof. LMSPF2 belongs to $\mathcal{NP}$ since a nondeterministic algorithm needs only to guess one of the two paths for each demand and apply the well-known polynomial time algorithm (e.g. see [1]) to obtain the allocation vector $\boldsymbol{x}$, and check if $\vec{\boldsymbol{x}} \succeq \vec{\boldsymbol{y}}$. We will transform PARTITION into a single-source multiple-sinks instance of LMSPF2, essentially using an idea from [4].

PARTITION
INSTANCE: Finite set A of items, and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$.
QUESTION: Is there a subset $A' \subset A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$?

Consider an arbitrary instance of PARTITION and construct a graph as follows: let two "core vertices", κ_l and κ_r, be connected by 2 "core edges", e and e', both of which has capacity $\frac{1}{2}\sum_{a \in A} s(a)$. For each element $a \in A$, connect by an edge of capacity $s(a)$ a vertex t_a, to the right core vertex, κ_r. Let $\boldsymbol{y} = [s(a)]_{a \in A}$. Now consider each (κ_l, t_a)-pair, $a \in A$, in the resulting graph as a demand (see Figure 1). Each (κ_l, t_a)-pair has exactly two admissible paths; one traversing e and one traversing e'. This construction, which is apparently done in polynomial time, is indeed a valid instance of LMSPF2 (with $|A|$ demands). Now assume that there is a positive answer to PARTITION. This implies existence of a subset $A' \subset A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$. Assign to each (κ_l, t_a)-pair a flow of $X_a = s(a)$. If $a \in A'$ let the demand use edge e', and if $a \in A \setminus A'$ edge e, together with its dedicated edge connecting κ_r and t_a. As $c_e = c_{e'} = \frac{1}{2}\sum_{a \in A} s(a)$, no link capacities are exceeded. We have thus an allocation $\boldsymbol{x}$, for which $X_a = s(a)$ for all $a \in A$, and consequently that $\vec{\boldsymbol{x}} = \vec{\boldsymbol{y}}$, implying a positive answer to LMSPF2. Conversely, suppose that there is a positive answer for the constructed instance of LMSPF2, i.e., that there exists a set of single paths for which an allocation vector $\boldsymbol{x}$, with $\vec{\boldsymbol{x}} \succeq \vec{\boldsymbol{y}}$, is obtainable. In particular, this means that there exist flows X_a, such that $X_a = s(a)$ for all $a \in A$. Thus the total flow between κ_l and κ_r is $\sum_{a \in A} s(a)$. But $c_e = c_{e'} = \frac{1}{2}\sum_{a \in A} s(a)$ and flows are unsplittable so there must

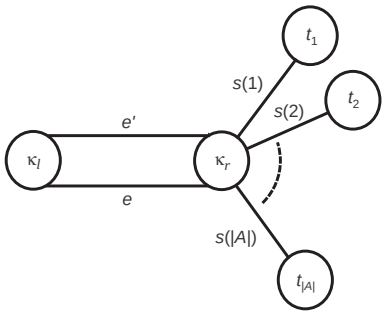

Figure 1: The resulting graph.

exist a subset $A' \subset A$, such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a) = \frac{1}{2}\sum_{a\in A} s(a)$, which answers the PARTITION question positively. Therefore, since PARTITION is $\mathcal{NP}$-complete [2], LMSPF2 must also be $\mathcal{NP}$-complete. □

In the unlimited path-sets case it was possible to prove $\mathcal{NP}$-completeness even of the subproblem of maximizing the minimal single path-flow. A natural question to ask is if this is possible also if path-sets are limited. Indeed, this is the case, as is shown by a transformation from 3-SATISFIABILITY in [9].

3 MIP Models

Because of the underlying application, we will from now on concentrate on the limited path-sets case. It will thus be assumed that for each instance, each demand d, $d \in D$, is given with a predefined (limited) set of paths, P_d. In this section, focus is put on representing subproblems of (1) as Mixed-Integer Programming (MIP) problems. Particularly, each MIP is designed to compute a specific entry, say entry j, of $\vec{x}$. We will sometimes denote this entry by the j:th *level*. It will become clear that the full problem can be solved (in some different ways) essentially by resolving a sequence of these MIPs.

3.1 The first level

The first level, i.e, the first element of $\vec{x}$ (called h_1), can be obtained by solving

$$\textbf{\textit{maximize}} \quad h_1 \tag{2}$$

$$\textbf{\textit{subject to}} \quad \sum_{p\in P_d} u_{dp} = 1 \qquad d \in D \tag{3}$$

$$\sum_{d\in D}\sum_{p\in P_d} \delta_{edp} u_{dp} h_1 \le c_e \qquad e \in E \tag{4}$$

$$h_1 \ge 0, \;\; u_{dp} \in \{0,1\}\ . \tag{5}$$

Apparently, this formulation has a difficulty of being nonlinear, containing multiplication of two variables (constraint (4)). However, by defining a new variable, $\mu = \frac{1}{h_1}$, this problem can be avoided. Of course, it is then necessary to presume that $h_1 > 0$, which is reasonable. Clearly, μ will be minimal when (and only when) h_1 is maximal. Further, if we restate the problem in variable μ, the nonlinearity is eliminated, making use of that neither h_1 nor μ are demand-dependent;

$$\textbf{\textit{minimize}} \quad \mu \tag{6}$$

$$\textbf{\textit{subject to}} \quad \sum_{p=1}^{P_d} u_{dp} = 1 \qquad d \in D \tag{7}$$

$$\sum_{d\in D}\sum_{p\in P_d} \delta_{edp} u_{dp} \le \mu c_e \qquad e \in E \tag{8}$$

$$\mu \ge 0, \;\; u_{dp} \in \{0,1\}. \tag{9}$$

The last formulation is a (linear) mixed 0-1 program. The advantages of the "inverted approach" taken in (6)-(9) has been exploited in [8], although for the splittable paths case.

3.2 Finding all levels – an explicit approach

The formulation presented in this section can be used as a quite straightforward way of expressing all the higher MMF-levels, h_k^*, $k = 2, \ldots, |D|$. Given the first level (computed e.g. by (6)-(9)), resolution of the presented MIP provides a solution to the studied problem. Defining $z_k = h_k - \sum_{i=2}^{k} h_{i-1}^*$, and letting M be a sufficiently large number, e.g., $M = \max_e\{c_e\}$, we may solve

$$\textbf{\textit{maximize}} \quad z_k \tag{10}$$

$$\textbf{\textit{subject to}} \quad \sum_{d\in D}\sum_{p\in P_d} x_{dp} = (|D| - (k-1))z_k \tag{11}$$

$$\sum_{p\in P_d} x_{dp} \le z_k \qquad d \in D \tag{12}$$

$$x_{dp} \le M w_{dp}^{(k)} \qquad p \in P_d,\ d \in D \tag{13}$$

$$\sum_{d\in D}\sum_{p\in P_d} w_{dp}^{(k)} = |D| - (k-1) \tag{14}$$

$$w_{dp}^{(2)} \le u_{dp} \qquad p \in P_d,\ d \in D \tag{15}$$

$$w_{dp}^{(r+1)} \le w_{dp}^{(r)} \qquad p \in P_d,\ d \in D,\ r \in \{2, \ldots, k-1\} \tag{16}$$

$$\sum_{d\in D}\sum_{p\in P_d} w_{dp}^{(r)} = |D| - (r-1) \qquad r \in \{2, \ldots, k-1\} \tag{17}$$

$$\sum_{p\in P_d} u_{dp} = 1 \qquad d \in D \tag{18}$$

$$\sum_{d\in D}\sum_{p\in P_d} \delta_{edp}(u_{dp}h_1^* + \sum_{r=2}^{k-1} w_{dp}^{(r)} z_r^* + x_{dp}) \le c_e \qquad e \in E \tag{19}$$

$$u_{dp} \in \{0,1\},\ w_{dp}^{(r)} \ge 0,\ x_{dp} \ge 0,\ z_k \ge 0. \tag{20}$$

It should be noted that the correctness of (10)-(20) relies entirely on that $w_{dp}^{(i)}$ is forced binary for all $i = 2, \ldots, k$, although there is no such explicit constraint.

Property 1 *If h_1^* and all z_i^*, $2 = 1, \ldots, k-1$, are optimal, then for any feasible solution to (10)-(20) it holds that $w_{dp}^{(i)} \in \{0,1\}$, for all $i = 2, \ldots, k$.*

Proof. Suppose that $w_{dp}^{(r-1)} \in \{0,1\}$, $p \in P_d,\ d \in D$, for some r, $3 \le r \le k$. As $w_{dp}^{(r-1)} \in \{0,1\}$, and since $\sum_{d\in D}\sum_{p\in P_d} w_{dp}^{(r-1)} = |D| - r + 2$, there exist exactly $|D| - r + 2$ non-zero variables $w_{dp}^{(r-1)}$, all of which must be equal to 1. By (16), $w_{dp}^{(r-1)} \ge w_{dp}^{(r)}$, so $w_{dp}^{(r-1)} = 0$ implies $w_{dp}^{(r)} = 0$. Now for z_{r-1}^* to be optimal there must exist a (d,p), $p \in P_d,\ d \in D$, such that $w_{dp}^{(r-1)} = 1$ and $w_{dp}^{(r)} = 0$ (otherwise z_{r-1}^* could be increased). Combined with the constraint $\sum_{p\in P_d} w_{dp}^{(r)} = |D| - r + 1$, this implies that there must exist $|D| - r + 1$ non-zero variables $w_{dp}^{(r)}$, that sum up to $|D| - r + 1$. Since $0 \le w_{dp}^{(r)} \le 1$, it follows that $w_{dp}^{(r)} \in \{0,1\}$. An identical argument can be used showing the same relation beween $w_{dp}^{(2)}$ and u_{dp}, implying that $w_{dp}^{(2)} \in \{0,1\}$ as $u_{dp} \in \{0,1\}$ by definition. Thus by induction over r the general result follows. □

A procedure to solve (1) is thus to solve (6)-(9) for the first level, h_1^*, and then to iteratively solve (10)-(20), for $z_i^*,\ i = 2, 3, \ldots, |D|$, and compute corresponding levels, $h_i^* = h_1^* + \sum_{j=2}^{i} z_j^*$.

3.3 Cumulated values – an implicit method

The MIP model outlined in this section is based on the method of describing a non-convex MMF problem presented in [10], under the notion of *conditional means*. It is referred to as an *implicit method*, since its original form constitutes a general way of expressing a wide range of non-convex MMF problems. Let $(h_1, h_2, \ldots, h_{|D|}) = \vec{\boldsymbol{x}}$, $\boldsymbol{x} = (X_1, X_2, \ldots, X_{|D|})$, where $X_d = \sum_{p \in P_d} x_{dp},\ d \in D$, is the total flow assigned to demand d. We are interested in the lexicographic maximization of this vector, i.e.,

$$\text{lex max } (h_1, h_2, \ldots, h_{|D|}) \,, \tag{21}$$

such that $\boldsymbol{x} \in Q$. Define the k:th *cumulated ordered value*, H_k, $k \leq |D|$, as $H_k = \sum_{j=1}^{k} h_j$. Then, for a given outcome vector $\boldsymbol{x}$, this entity can be computed as

$$\begin{aligned} H_k \;=\; & \min \sum_{d \in D} X_d a_{kd} && (22)\\ & \text{s.t. } \sum_{d \in D} a_{kd} = k && (23)\\ & 0 \leq a_{kd} \leq 1,\ d \in D. && (24) \end{aligned}$$

The optimization problem that is dual to (22)-(24) may be directly stated as

$$\begin{aligned} H_k \;=\; & \max kr_k - \sum_{d \in D} \lambda_{kd} && (25)\\ & \text{s.t. } r_k - X_d \leq \lambda_{kd},\ d \in D && (26)\\ & \lambda_{kd} \geq 0,\ d \in D, && (27) \end{aligned}$$

which is linear even if $\boldsymbol{x}$ is a variable. In terms of the solution, lexicographic maximization of all the cumulated ordered values, H_k, $k \in \{1, \ldots, |D|\}$, with $\boldsymbol{x} \in Q$ ($\boldsymbol{x}$ is a variable), can in fact be shown equivalent to lexicographic maximization of the sorted allocation vector, $\vec{\boldsymbol{x}}$, with $\boldsymbol{x} \in Q$ [10]. This yields the following representation of the studied problem:

$$\begin{aligned} & \text{lex max } \left(H_1, H_2, \ldots, H_{|D|} \right) && (28)\\ & \text{s.t. } H_k = kr_k - \sum_{d \in D} \lambda_{kd},\ k \in \{1, \ldots, |D|\} && (29)\\ & r_k - X_d \leq \lambda_{kd},\ k \in \{1, \ldots, |D|\},\ d \in D && (30)\\ & \lambda_{kd} \geq 0,\ k \in \{1, \ldots, |D|\},\ d \in D && (31)\\ & \boldsymbol{x} \in Q. && (32) \end{aligned}$$

For the purpose of solving problem (28)-(32), with Q constituted by the single-path and the link-load constraints, it is possible to solve the following MIP iteratively for $k = 2, \ldots, |D|$:

$$\begin{aligned} & \textbf{\textit{maximize}} && H_k && && (33)\\ & \textbf{\textit{subject to}} && H_k \leq kr_k - \sum_{d \in D} \lambda_{kd} && && (34)\\ & && H_l^* \leq lr_l - \sum_{d \in D} \lambda_{ld} && l \in \{1, \ldots, k-1\} && (35)\\ & && r_l - \sum_{p \in P_d} x_{dp} \leq \lambda_{ld} && l \in \{1, \ldots, k\},\ d \in D && (36)\\ & && x_{dp} \leq u_{dp} M && p \in P_d,\ d \in D && (37)\\ & && \sum_{p \in P_d} u_{dp} = 1 && d \in D && (38)\\ & && \sum_{d \in D} \sum_{p \in P_d} \delta_{edp} x_{dp} \leq c_e && e \in E && (39)\\ & && x_{dp} \geq 0\,,\ \ \lambda_{id} \geq 0\,,\ \ u_{dp} \in \{0, 1\}, && && (40) \end{aligned}$$

where M is a sufficiently large number, e.g. $M = \max_e\{c_e\}$, and $H_l^*,\ l \in \{1,\ldots,k-1\}$, are the cumulated ordered values obtained in previous iterations. On purpose, this MIP does not cover computation of the first level, $H_1 = h_1$, which may be more efficiently computed by (6)-(9).

3.4 The distribution approach – finite outcome set

In many cases, especially in the communication network context, it is very reasonable to assume that the flow volume allocated to demand d may take on only a limited set of discrete values. Such a restriction results in essentially the same problem, but with a *finite outcome set*.

Property 2 *If the flow volumes X_d, for all demands $d \in D$, only can assume values from a finite set of levels, $v_1, v_2, \ldots, v_r$, then*

$$\textit{lex max } \vec{\boldsymbol{x}};\ \boldsymbol{x} \in Q, \tag{41}$$

is equivalent to

$$\textit{lex min } \left(\sum_{d\in D} t_{2d}, \sum_{d\in D} t_{3d}, \ldots, \sum_{d\in D} t_{rd} \right) \tag{42}$$

$$\textit{s.t. } v_k - X_d \le t_{kd},\ k \in \{2,\ldots,r\},\ d \in D \tag{43}$$

$$t_{kd} \ge 0,\ k \in \{2,\ldots,r\},\ d \in D \tag{44}$$

$$\boldsymbol{x} \in Q. \tag{45}$$

This property is derived formally in [10]. Intuitively, it is not hard to see that (42)-(45) first assures that as many demands as possible are raised to the first non-zero level, v_2 ($v_1 = 0$). Keeping this many demand volumes greater or equal than v_2, it is then assured that as many demands as possible are raised to v_3, and so on. The obvious benefit is that if the demand volumes are bound to be modular, then (42)-(45) may be solved to get a solution to the studied problem. This can be done by solving the following MIP iteratively.

$$\textbf{\textit{minimize}} \quad \sum_{d\in D} t_{kd} \tag{46}$$

$$\textbf{\textit{subject to}} \quad v_l - \sum_{p\in P_d} x_{dp} \le t_{ld} \qquad d \in D,\ l \in \{2,\ldots,k-1\} \tag{47}$$

$$\sum_{d\in D} t_{ld} \le \tau_l^* \qquad l \in \{2,\ldots,k-1\} \tag{48}$$

$$v_k - \sum_{p\in P_d} x_{dp} \le t_{kd} \qquad d \in D \tag{49}$$

$$x_{dp} \le u_{dp} M \qquad p \in P_d,\ d \in D \tag{50}$$

$$\sum_{p\in P_d} u_{dp} = 1 \qquad d \in D \tag{51}$$

$$\sum_{d\in D} \sum_{p\in P_d} \delta_{edp} x_{dp} \le c_e \qquad e \in E \tag{52}$$

$$x_{dp} \ge 0\ ,\ \ t_{ld} \ge 0\ ,\ \ u_{dp} \in \{0,1\}, \tag{53}$$

where M is a sufficiently large number, e.g. $M = v_k$.

3.5 Upper and Lower bounds

In a communication network, it is often a reasonable assumption that each demand $d \in D$ is given with acceptation limits, q_d and Q_d, where it is required that the allocation vector satisfies $q_d \le X_d \le Q_d$, for all $d \in D$. The upper limit, Q_d, can be incorporated in a natural fashion in all of the presented MIP models. This is just a matter of substituting the constraints of the type $x_{dp} \le u_{dp}M$, with $x_{dp} \le u_{dp}Q_d$ for all $p \in P_d$, $d \in D$. The lower limit, q_d, has to be added as explicit constraints, $q_d \le X_d$ for all $d \in D$. Note that the acceptation limits have to be carefully a priori determined, since otherwise there is a large risk that they make the MIP model infeasible.

4 Some numerical experiments

As has been shown, the studied problem suffers from heavy computational complexity. This is mainly due to the single-paths requirement and the fact that it embodies a multi-criteria optimization problem. Therefore, it is indeed interesting to study and compare computation times for different instances and for different resolution techniques.

4.1 Exact methods

In this section we give a flavour of the computation times associated with solving

$$\text{lex max } \vec{\boldsymbol{x}};\ \boldsymbol{x} \in Q,$$

with the two different approaches described in sections 3.2 and 3.3, respectively. Both these approaches offer exact methods for solving the considered problem. Particularly, we present two algorithms, called the *explicit* and the *implicit* methods. The explicit method (Algorithm 1) invokes the MIP formulated by (10)-(20), whereas

Algorithm 1 *The explicit method*

1: solve (6)-(9) for h_1^*
2: **for** $k := 2$ to $|D|$ **do**
3: solve (10)-(20) for z_k^*
4: **end for**
5: Compute h_k^* from z_k^*, $k = 2, \ldots, |D|$

the implicit method (Algorithm 2) invokes the MIP of (33)-(40). Both methods require resolution of a sequence

Algorithm 2 *The implicit method*

1: solve (6)-(9) for $h_1^* = H_1^*$
2: **for** $k := 2$ to $|D|$ **do**
3: solve (33)-(40) for H_k^*
4: **end for**
5: compute $\vec{\boldsymbol{x}}$ from H_k^*, $k = 1, \ldots, |D|$

of MIPs. The frameworks of the algorithms are implemented in MATLAB6.5, and the MIPs are solved using a MATLAB interface to CPLEX 9 (mex-function), called CPLEXINT (downloadable freeware [3]). This means that the generic CPLEX 9 MIP-solver is used for resolving MIPs (6)-(9), (10)-(20), and (33)-(40). The computations were carried out on a PC with an Intel PIII-1GHz CPU, RAM of 256 MB, and Windows 2000 OS. Table 1 contains computation times for the two different methods, for a number of small instances. The instances are characterized by that there is a demand between every node-pair and that link capacities are uniformly distributed over $\{10, 20, 30, 40, 50\}$. For the first 7 instances there are two paths per demand, and for the 5 last instances there are three paths per demand. The results of Table 1 suggest that the implicit method is slightly faster than that of the explicit method. If we sum up the usage of variables and constraints for the MIP corresponding to iteration k in the two different approaches, this is reasonable to expect.

4.2 The distribution approach

In this section we investigate numerically the method described in Section 3.4. This method can be viewed as a resolution technique for the studied problem if demand volumes only can assume some predefined, discrete levels. Furthermore, it can also be regarded as an approximate method for solving the studied problem without restrictions on demand volumes. From the latter viewpoint it is interesting how much faster the problem can be solved if this approximate method is used, i.e., how much computation time that can be saved if we predefine a grid, $\{v_1, v_2, \ldots, v_r\}$, and require that demand volumes only can assume values from this, and to what cost, in terms of deviation from optimum, this may be done. Algorithm 3 is based on the distribution approach. The hardware and software constellations are identical to those of Section 4.1. In Table 2, the distribution approach is applied to a number of different instances. Again, link capacities are uniformly distributed over

Algorithm 3 *The distribution approach*

1: $l := 2$, $A := true$, $x^1_{dp} := 0$ for all $p \in P_d,\ d \in D$
2: **while** $l \leq r$ and $A = true$ **do**
3: solve (46)-(53) for τ^*_l, x^l_{dp} and u_{dp} for all $p \in P_d,\ d \in D$
4: **if** $\sum_{p \in P_d} x^l_{dp} - \sum_{p \in P_d} x^{l-1}_{dp} = 0$ for all $d \in D$ **then**
5: $A := false$
6: **end if**
7: $l := l + 1$
8: **end while**

$\{10, 20, 30, 40, 50\}$, and there is a demand between every node-pair in the network. The elements of the predefined grid, $v_1, v_2, \ldots, v_r$, are all the numbers v_i that can be written as $v_i = m \cdot z$, $z \in \mathbb{Z}$, $0 \leq v_i \leq 50$, where m is called the module size. The computation times are given for Algorithm 3 with $m = 2$ (dist-2) and $m = 5$ (dist-5). Application of the best exact method (Algorithm 2), did not finish in less than half an hour for any of the instances in Table 2. Comparison of the exact methods and the distribution approach, by the

#nodes	#links	#paths/ demand	computation times (s)	
			explicit method	implicit method
5	8	2	0.8130	0.4710
6	12	2	1.1230	1.3730
7	12	2	10.098	4.7190
8	13	2	16.420	21.139
9	18	2	$1.62{\cdot}10^3$	327.92
10	18	2	$1.61{\cdot}10^3$	327.11
11	19	2	$1.92{\cdot}10^3$	107.41
4	6	3	0.4220	0.1080
5	8	3	1.1560	0.7020
6	12	3	8.8550	13.077
7	17	3	26.738	28.999
8	12	3	236.49	46.704

Table 1: The explicit and implicit methods.

#nodes	#links	#paths/ demand	computation times (s)	
			dist-2	dist-5
10	22	3	58.518	5.3940
11	23	3	180.81	2.5340
12	24	3	50.954	1.5780
13	26	3	51.781	2.5780
14	24	3	16.640	1.2200
15	25	3	47.485	2.4700
16	33	3	59.623	5.6410
16	34	3	29.400	3.1260
17	36	3	99.595	12.333
18	40	3	358.72	5.5310
19	38	3	$1.838{\cdot}10^3$	4.9680
20	36	3	108.31	8.0770

Table 2: The distribution approach.

computation times given in Table 1 and Table 2, strongly suggests that the distribution approach should be used whenever its deviation from the true optimum is acceptable. Certainly, such an error tolerance will depend on the details of the application. Even though it is possible to give an artificial example for where the deviation is larger, we observe that an entry of the sorted distribution approach solution vector practically always differs by at most the module size from an entry of the sorted optimal allocation vector. Besides, if the demand volumes are restricted to be modular, the distribution approach is actually an exact method.

5 Conclusion

This paper investigates a problem of obtaining a max-min fair demand bandwidth assignment in a communication network, when only unsplittable (single-path) flows are allowed. First, we show that if for each demand (node-pair), any single path can be used, the problem is obviously $\mathcal{NP}$-hard. Since in a communication network it is often quite unreasonable to assume that a demand may use any path connecting its end nodes, this result is not sufficient in showing the associated computational complexity. Therefore, it is then assumed that each demand is given a predefined (limited) path-set. We show that this problem, with cardinalities of path-sets limited to as little as 2, is also $\mathcal{NP}$-hard.

Having established the computational complexity for the considered problem, we proceed to resolution techniques. This is done by representing subproblems as mixed-integer programs (MIP). Particularly, each MIP is constructed to compute a consecutive entry of the optimal MMF allocation vector. In essence, the full problem is treated by resolving a sequence of such MIPs. In this spirit, a problem-specific resolution technique is developed. This technique is then compared with a general resolution technique for non-convex lexicographic maximization. In its current status the problem-specific resolution technique turns out to be somewhat slower than the general approach. Further developments may however change this observation. Finally, we examine a resolution technique called the distribution approach, which assumes that flow allocated to a demand may

take on only a limited set of discrete values, i.e., we assume a finite outcome set. This models the modularity of flow in a communication network. Moreover, for the continuous flow case, this approach may serve as an approximation technique. It is shown that even with very small (relative to link capacities) module sizes, the distribution approach is substantially faster than its counterparts that assume non-modularity. Consequently, we conclude that the distribution approach is advantageous in addressing the studied problem, both because its approximation ability, and as an exact solver in the case of demand volumes being multiples of a module.

References

[1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[2] M. Garey and D. Johnson. *Computers and intractability – a guide to the theory of NP-completeness*. Freeman, 1979.

[3] http://control.ee.ethz.ch/∼hybrid/cplexint.php.

[4] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, MIT, 1996.

[5] J. Kleinberg. Single-source unsplittable flow. In *IEEE Symposium on Foundations of Computer Science*, pages 68–77, 1996.

[6] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001.

[7] N. Megiddo. Optimal flows in networks with sources and sinks. *Mathematical Programming*, 7, 1974.

[8] D. Nace. A linear programming based approach for computing optimal fair splittable routing. In *IEEE International Symposium on Computers and Communications*, pages 468–474, 2002.

[9] P. Nilsson and M. Pioro. Max-min fairness of unsplittable network flows. Technical report, Dept. of Communication Sytems, Lund University, June 2005. CODEN: LUTEDX(TETS-7209)/1-21/(2005)&local 25.

[10] W. Ogryczak, M. Pioro, and A. Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of telecommunications and information technology*, 3, 2005.

[11] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann (Elsevier), 2004.

[12] M. Pióro, P. Nilsson, E. Kubilinskas, and G. Fodor. On efficient max-min fair routing algorithms. In *IEEE International Symposium on Computers and Communications*, pages 465–472, 2003.