Max-Min Fairness in multi-commodity flows

Dritan Nace[1], Linh Nhat Doan[1], Olivier Klopfenstein[2]
and Alfred Bashllari[1]
[1] Université de Technologie de Compiègne,
Laboratoire Heudiasyc UMR CNRS 6599,
60205 Compiègne Cedex, France
E-mail: {nace,doannhat,abashlla}@hds.utc.fr
[2] France Télécom R&D, 38-40 rue du Général-Leclerc
92794 Issy-les-Moulineaux Cedex 9, France.
E-mail: olivier.klopfenstein@francetelecom.com

February 10, 2006

Abstract

In this paper, we provide a study of Max-Min Fair (MMF) multi-commodity flows and focus on some of their applications to multi-commodity networks. We first present the theoretical background for the problem of MMF and recall its relations with lexicographic optimization as well as a polynomial approach for achieving leximin maximization. We next describe two applications to telecommunication networks, one on routing and the second on load-balancing. We provide some deeper theoretical analysis of MMF multi-commodity flows, show how to solve the lexicographically minimum load network problem for the link load functions most frequently used in telecommunication networks. Some computational results illustrate the behavior of the obtained solutions and the required CPU time for a range of random and well-dimensioned networks.

1 Introduction

In this paper, we provide a study of Max-Min Fair (MMF) multi-commodity flows and focus on their applications to networks. More generally, MMF finds applications in numerous areas where it is desirable to achieve an equitable distribution or sharing of resources and is therefore closely related to maxmin or minmax optimization problems, widely studied in the literature. Recently in [21], the lexicographically minimum and maximum load problems are introduced and studied; a polynomial approach is proposed to solve them, and some applications are cited. We show how the proposed approach can be applied to the max-min fair multicommodity flow problem. We then focus on two network applications and exhibit some new theoretical properties followed by computational results.

From the telecommunication point of view, the models proposed address medium to long-term network planning issues. For this reason, routing paths

are computed *off-line*, based on a static view of the network taking into account, for instance, estimated average or worst-case traffic demands. Obviously, network managers are interested in designing a routing scheme capable of withstanding changes and instabilities in the network, (*i.e.,* a "robust" routing scheme). Max-min fair approaches are shown to address, to a certain extent, this problem. Additionally, in economical studies, we are sometimes faced with two opposing behaviors or goals: network-aware (*i.e.,* maximizing the *overall throughput* of the network) and user-aware (*i.e.,* maximizing the *individual throughput* for each particular customer). Obviously max-min fairness goes in line with the user-aware goal as it tries to put some equity in resource utilization, and the approaches proposed here could be useful tools to measure its impact in terms of network resources. Clearly, classical quality of service issues (delay, jitter, packet loss rates, etc.) cannot be directly tackled with our static models. Nevertheless, optimizing the bandwidth utilization in network planning is a classical and relevant, although approximate, way to make quality of service easier to achieve, since most QoS criteria are related to the available bandwidth.

The major contributions of this work are providing a deeper theoretical analysis of MMF multi-commodity flows and solving the lexicographically minimum load problem in networks for the link load functions most frequently used in telecommunication networks. Furthermore, numerical results obtained with realistic instances could be useful for network managers when designing the best suited routing in the network as well when deciding the most convenient link load function to be used.

The paper is organized as follows. We present in Section 2 the theoretical background of MMF, recall its relations with lexicographic optimization, and the approach proposed in [21] for solving the lexicographically optimum load problems. In Section 3 and 4, after summarizing some related work, we present in greater detail its application to routing as applied in telecommunication networks, along with some computational results. In Section 5 we recall the lexicographically minimum load network problem, and present some theoretical results that allow us to deal with a large range of link load functions frequently used in telecommunication networks. Some numerical results are also reported at the end of this section.

2 Preliminaries

Let us begin by introducing formally the notion of max-min fairness. Bertsekas and Gallager [3] were among the first to study MMF with respect to rate allocation in a given network. They define an allocation of bandwidth to a set of connections to be *Max-Min Fair* if it is not possible to increase the allotted rate of any connection exclusively at the expense of connections whose rates are greater. Mathematically, this can be expressed as follows: let P be the set of connections routed in a given network and $\{\gamma_p : p \in P\}$ the associated rates. The allocation γ is said to be *feasible* if $\gamma \geq 0$ and all capacity constraints are satisfied. Consequently, the allocation γ is termed max-min fair if it is feasible and for any other feasible allocation γ', the existence of $p \in P$ such that $\gamma'_p > \gamma_p$ implies: there exists $s \in P : \gamma'_s < \gamma_s \leq \gamma_p$. This definition was introduced for the case of fixed single-path routing. The max-min fairness definition can then

be generalized to a set of vectors on a set $\Gamma \subseteq R^n$ as follows:

Definition 1. *A vector γ is max-min fair on set Γ if and only if* $(\forall \gamma' \in \Gamma)(\exists s \in \{1, ..., n\})(\gamma'_s > \gamma_s) \Rightarrow (\exists t \in \{1, ..., n\})(\gamma'_t < \gamma_t \leq \gamma_s)$.

Another generalization of max-min fairness is given in [15] where the authors define the state of *approximated equilibrium* for an allocation vector. With respect to max-min fairness this can be interpreted as follows: for a constant c, an allocation vector γ is *c-approximated max-min fair* if it is not possible to raise the value of a component γ_i without decreasing some other component γ_j such that $\gamma_j \leq c\gamma_i$. The max-min fair vector defined above is thus the 1-approximated max-min fair vector.

Max-min fairness is closely related to lexicographic optimization as remarked explicitly or not in numerous studies, [15, 22, 27, 28], etc. Let us recall some definitions on lexicographic ordering.

Definition 2. *A vector γ is lexicographically greater than γ' if there exists $s \in \{1, ..., n\}$ such that $\gamma_p = {\gamma'}_p$, for all $p \in \{1, ..., s-1\}$ and $\gamma_s > {\gamma'}_s$.*

Definition 3. *A vector γ is lexicographically maximal in Γ if for every vector $\gamma' \in \Gamma$, γ is lexicographically greater than or equal to γ'.*

Let $\overrightarrow{\gamma}$ be the vector γ with its indices reordered so that the components are in non-decreasing order. A feasible vector is defined as leximin maximal [28], as follows:

Definition 4. *A vector $\gamma \in \Gamma$ is leximin maximal if for every vector $\gamma' \in \Gamma$, $\overrightarrow{\gamma}$ is lexicographically greater than or equal to $\overrightarrow{\gamma'}$.*

Let us now see how the max-min fairness concept can be transposed to real valued functions. For a given vector γ, consider the following system:

$$\{f_i(x) \geq \gamma_i; \quad i \in 1, ..., m, x \in X \subseteq R^n_+\} \tag{1}$$

We have focused on the feasible vectors γ that are lexicographically maximal, leximin maximal or max-min fair. Although computing a lexicographically maximal vector for the system (1) is relatively easy, this is not always the case when computing a max-min fair vector. Fortunately, in many cases it turns out to be as simple as achieving leximin maximization, by using the following theoretical results.

Max-min fairness versus leximin maximization. The relationship between max-min fairness and leximin maximization has already been studied in [28] as we shall now recall. Let Γ denote the set of vectors γ. Radunovic and Le Boudec show that if Γ is convex and compact, it possesses a unique max-min fair vector. Furthermore, if a max-min fair vector exists on a set Γ, then it is the unique leximin maximal vector on Γ. In other words, for a range of applications the problem of computing an MMF vector is reduced to a leximin maximal computing problem[1]. Let us consider in the following paragraph the lexicographic optimization problem related to the system (1) defined above for the linear case.

[1]The definitions of min-max fairness, lexicographically mimimum vectors and leximax minimal vectors are analogous. The same relations exist between min-max fairness and leximax minimization.

Linear functions. In this paragraph we recall briefly a polynomial approach for computing leximin maximal vectors for system (1) in the linear case, (see [21] for further details). Let each function $f_i(x)$ be given by $\sum_{j=1}^{n} a_{ij}x_j$. A vector γ is termed *feasible* if each of its values is a real valued number and if there is a feasible solution to the linear system (1). Let us now see how to compute a leximin maximal vector among the feasible vectors for system (1) by solving a sequence of linear programs.

Algorithm 1. Computing the leximin maximal vector for a given linear system.

Step 1.

Find a maximum scalar value α^* such that $\gamma_i = \alpha^*$ for all i is a feasible vector, and identify all binding constraints. If $\alpha^* = +\infty$, then every variable γ_i gets the value of infinity and the algorithm ends here; otherwise go to Step 2.

Step 2.

Given a strictly complementary optimal solution[2], let L denote the indices of the binding constraints and let $\gamma_k = \alpha^*$. Set $\sum_{j=1}^{n} a_{kj}x_j = \alpha^*$ for all $k \in L$ and fix them to equalities. Next, iterate between Steps 1 and 2 until no inequality constraint remains in the system.

Typically, for step 1 at some iteration k, we solve the following linear program:

$$Maximize \quad \alpha$$

$$(\pi_i) \quad \sum_{j=1}^{n} a_{i,j}x_j \geq \alpha b_i \quad \forall i \in \{1..m\} \setminus L_{k-1} \tag{2}$$

$$(\pi_i) \quad \sum_{j=1}^{n} a_{i,j}x_j = \gamma_i b_i \quad \forall i \in L_{k-1} \tag{3}$$

$$x \in X. \tag{4}$$

where L_{k-1} gives the set of indices of the binding constraints encountered during the first $k-1$ iterations, while x is restricted to a feasible set $X \subseteq R_+^n$ and the respective dual variables are in parentheses.

Finding a strictly complementary optimal solution may be achieved by using an IPM (Interior Point Method) method based on the central trajectory [11]. Binding constraints are then simply identified among constraints (2) as these with positive dual values (*i.e.,* $\pi_i > 0; i \in \{1..m\} \setminus L_{k-1}$). However, it is possible to achieve strictly complementary solutions by any LP solver using the method given in [12]. Finally, the vector γ obtained at the end of Algorithm 1 is leximin maximal and it can be obtained in polynomial time [21].

3 Max-min fairness applied to routing

With respect to applications of MMF to telecommunication networks, a lot of work related to rate adaptation and congestion control in TCP networks (see [20, 6, 5, 14, 19] etc.) has already been done, while its relations with routing and network design have been the subject of several works over the last decade

[2] An optimal solution is termed strictly complementary if there is only one zero value for each complementary pair of slack variables.

[9, 15, 4, 7, 17, 22, 24, 27, 26], etc. Among them, some work has been devoted to the static routing (connections and corresponding routing paths are given) case, where source rates are subject to change (see for instance [3, 9]). In [3] the authors present the *progressive filling* algorithm for achieving a max-min fair distribution of resources to connections for the fixed single path routing case (where each connection is associated with a particular fixed path). The main idea behind the algorithm is to uniformly increase the individual allocations of connections until one or more link becomes congested. Then the connections that cannot be improved are removed from the network together with the capacities they occupy; the process continues until all connections are removed. In [9] the authors propose an LP model for the static case and an extension of their algorithm with a heuristic for computing variants of fair routing. Other works [4, 7, 17] examine and compare performances of representative routing algorithms. In these three studies the authors have tried to find an algorithm with fair properties and/or maximizing the overall throughput. They are all online routing algorithms: that is, for each new connection, the source computes in real-time the appropriate path according to local or collected information. In [22, 23], the problem of max-min fair bandwidth-sharing among elastic traffic connections when routing is not fixed has been considered in an off-line context. The proposed algorithm can be seen as an extension of the progressive filling algorithm given in [3] except that the routing is not fixed and at each iteration a new routing is computed while the previously saturated links and the corresponding fair sharing remain fixed until the end of the algorithm. However, this work has led to a heavy computationally LP model. Finally, two similar simpler approaches (polynomial) enriched by theoretical analysis have been given in [24, 26]. The mathematical model presented in the next section is strongly inspired from these two works.

3.1 Max-min fair multi-commodity flows

In several application settings, for instance long-term planning and network design, the routing problem is frequently formulated through multi-commodity flows. In the following we will focus on the max-min fair multi-commodity flows, which has an obvious relationship to max-min fair routing. Flow problems in networks have been extensively studied since the seminal work of Ford and Fulkerson [10]. A number of variants of the flow problem in a graph with edge capacities frequently arise in operations research and other settings [1]. A widely studied problem is the maximum flow problem. In [10], the authors have shown that multiple-source multiple-sink networks can be reduced to single-source single-sink networks by the adjunction of a super-source, a super-sink and some additional arcs. As a consequence, the problem can be solved by the well-known labeling method for searching augmenting paths. However, this is not applicable when, instead of maximizing the flow globally, one needs to distribute it fairly. Generally speaking, it might be desirable to maximize, with respect to network resources, the minimum amount which is supplied to sinks or commodities (source-sink couples). Furthermore, it is sometimes a requirement to extend the expected fairness beyond this basic level to a more globally fair distribution of resources. We consider here the max-min fair multi-commodity flow problem which, as we will show, is equivalent to the leximin maximization of the vector of individual flows in multi-commodity networks. We should point

out that we are interested here in multi path routing (*i.e.,* splittable) case, while in the single path routing case even the fair single-source unsplittable flow problem is shown to be NP-complete [15]. Let us define formally in the following a multi-commodity flow, a flow vector and the max-min fair multi-commodity flow problem.

Let a capacitated network be given by a triplet $N = (V, A, C)$, where V is a nonempty finite set whose elements are the nodes (or routers). A is the set of ordered pairs (called arcs or links) of nodes, and C is a function from A to nonnegative reals, called the capacity function. For each arc (i, j), the associated capacity is denoted by C_{ij}. Let D be the set of commodities d labeled in $\{1, 2, ..., |D|\}$. Each commodity has a source node s^d and a sink node t^d, and a traffic value to be satisfied. A flow f^d (of value $b \geq 0$) in network N with respect to commodity d is a function from A to $\mathbb{R}^+$ such that:

$$f_{ij}^d \leq C_{ij} \quad \forall (i,j) \in A \tag{5}$$

$$\sum_{j:(i,j)\in A} f_{ij}^d - \sum_{j:(j,i)\in A} f_{ji}^d = \left\{ \begin{array}{ll} b, & i = s^d \\ -b, & i = t^d \\ = 0, & otherwise \end{array} \right\} \quad \forall i \in V \tag{6}$$

where f_{ij}^d give the arc-flow values. Any flow satisfying constraints (5) and (6) is called feasible. A multi-commodity flow f is composed of feasible flows f^d with respect to commodities $d \in D$ such that: for all $(i, j) \in A$, $\sum_{d\in D} f_{ij}^d \leq C_{ij}$. This multi-commodity flow is called feasible. To each multi-commodity flow one can associate some vector, called *flow vector*, composed of source-to-destination commodity flow values corresponding to the component flows. More formally, the flow vector can be defined as follows:

Definition 5. *Given a multi-commodity flow f, the vector $\lambda = \{\lambda[d], d \in D\}$ whose components give the flow value with respect to commodities is called the flow vector of multi-commodity flow f.*

Obviously, the flow vector is defined on $X \subset R^{|D|}$. Let us denote as F the set of all the feasible multi-commodity flows in network N with respect to a set of commodities D and as X the set of feasible flow vectors with respect to F. Henceforward the term *MMF multi-commodity flow* will always refer to a multi-commodity flow whose flow vector is max-min fair. Similarly, *MMF flow vector* will refer to a max-min fair flow vector. Now, the MMF multi-commodity flow problem can be stated: *given a capacitated network $N = (V, A, C)$, find an MMF multi-commodity flow f with respect to a given set of commodities D.*

A simpler statement of the MMF multi-commodity flow problem can be deduced from the following proposition.

Proposition 1. *In a given capacitated network $N = (V, A, C)$ and a set of commodities D, there exists a max-min fair flow vector and it is the unique leximin maximal flow vector.*

According to some results presented in [28] and recalled in Section 2, the proof becomes straightforward if we can show that the set of feasible flow vectors is convex and compact. Let us first consider the convexity of X: for any $x, y \in X$

with multi-commodity flows χ and ψ (in F), $\alpha x + (1-\alpha)y$ with $\alpha \in [0,1]$ is also in X. This statement can be seen to be true since $\alpha\chi + (1-\alpha)\psi$ gives a feasible multi-commodity flow and its flow vector is quite simply $\alpha x + (1-\alpha)y$, which proves that X is convex. Second, let us recall that a subset of $R^{|D|}$ is compact if and only if it is closed and bounded; $|D|$ gives the number of commodities. Notice that the set of feasible vectors is defined in $X \subset R^{|D|}$, $(|D| \leq |V| * (|V|-1))$. Further, it can easily be seen that such a set is closed and bounded since the set of feasible multi-commodity flows contains all its limits and is bounded. This concludes the proof of the above proposition. □

In this way, we can restrict ourselves in searching for leximin maximization instead of max-min fairness and the problem can be restated as follows: *given a capacitated network $N = (V, A, C)$, find a multi-commodity flow such that the its flow vector with respect to a given set of commodities D is leximin maximal.*

3.1.1 Mathematical modeling and resolution approach

Our problem is computing the leximin maximal flow vector with respect to a given network and a demand traffic matrix, which can be accomplished using the approach proposed in [21] (recalled in Section 2). In the following, we show in greater detail how this approach is applied for solving the MMF multi-commodity flow problem. Intuitively, the main idea behind the algorithm is that the first lowest value ($\overrightarrow{\lambda}[1]$) has to be maximized before the second lowest value ($\overrightarrow{\lambda}[2]$) is maximized, and inductively, the maximization for a component is carried out after components whose values are less good than the given value have been maximized. A commodity whose flow value has already been lexicographically maximized in the sense of Algorithm 1, is called *saturated*. In other terms, the corresponding traffic constraint was a binding constraint during the last executed step, and it has been fixed at equality. To this end we run a program similar to (2-4), referred to below as problem Pk, at each step. The algorithm terminates when all commodities are saturated. Henceforward an arc will be termed saturated when there is no more capacity left on it, that is, all its capacity has already been assigned to flows. The algorithm can then be stated as follows:

Algorithm MMF (Max-Min Fair multi-commodity flow)

Input: A capacitated network $N = (V, A, C)$ and a set of commodities $d \in D$.

Output: Find the max-min fair multi-commodity flow and the associated flow vector.

(I) Set $k := 0$ and $L_0 = \phi$;

(II) While $L_k \neq D$ do:
- Set $k = k + 1$. Solve the LP problem P_k, compute α;
- Identify the set D_k of saturated demands, set $\lambda[d] = \alpha$ for all $d \in D_k$ and $L_k = L_{k-1} \cup D_k$.

(III) The flow vector obtained at the last step is leximin maximal, and the obtained multi-commodity flow is thus a max-min fair one.

where D_k denotes the set of commodities saturated at the (same) value obtained at step k of the algorithm and L_k denotes the set of demands saturated during

the first k steps of the algorithm and given by $\bigcup_{1 \leq p \leq k} D_p$. More precisely, D_k is the set of demands whose flow values cannot by any means be further increased above α. The precise LP formulation of problem P_k and some methods for identifying the saturated demands are presented below. Obviously, each step of the algorithm is intended to maximize the current α, that is the next component of the sorted flow vector. It terminates when all commodity flows are saturated.

Formulating and solving the problem P_k:
The problem P_k can be modeled using a classic arc-node LP formulation as follows:

$$Maximize \quad \alpha$$

$$(\omega_{i,j}) \qquad \sum_{d \in D} f_{ij}^d \leq C_{ij} \qquad \forall (i,j) \in A \tag{7}$$

$$(\pi_{i,d}) \qquad \sum_{j:(i,j)\in A} f_{ij}^d - \sum_{j:(j,i)\in A} f_{ji}^d = 0 \qquad \forall d \in D, \forall i \notin \{s^d, t^d\} \tag{8}$$

$$(\pi_{i,d}) \qquad \sum_{j:(i,j)\in A} f_{ij}^d - \sum_{j:(j,i)\in A} f_{ji}^d = \begin{cases} \lambda[d], i = s^d \\ -\lambda[d], i = t^d \end{cases} \qquad \forall d \in L_{k-1}, \forall i \in V \tag{9}$$

$$(\pi_{s^d,d}) \qquad \sum_{j:(j,i)\in A} f_{ji}^d - \sum_{j:(i,j)\in A} f_{ij}^d + \alpha \leq 0 \qquad \forall d \in D \setminus L_{k-1}, i = s^d \tag{10}$$

$$(\pi_{t^d,d}) \qquad \sum_{j:(i,j)\in A} f_{ij}^d - \sum_{j:(j,i)\in A} f_{ji}^d + \alpha \leq 0 \qquad \forall d \in D \setminus L_{k-1}, i = t^d \tag{11}$$

$$f_{ij}^d \geq 0 \qquad \forall d \in D, \forall (i,j) \in A \tag{12}$$

where constraints (7) are capacity constraints, constraints (8) are flow conservation constraints and (9, 10, 11) are traffic constraints. Finally, constraints (12) indicate that arc-flow values must be non-negative while in parentheses we give the respective dual variables. At the end of the solving procedure we obtain a multi-commodity flow, the optimal solution value α, and the dual values associated with constraints. In practice, several commodity flows can be simultaneously saturated at the end of the k^{th} step. For this, different methods can be used to identify the binding constraints allowing the set D_k to be computed. A direct method is to find a strictly complementary optimal solution at each step of the Algorithm MMF; that is, in each complementary pair of slack variables, one should be strictly positive (no pair should consist of two zero values). This can be achieved by IPM methods [11]. In the set D_k one can put all demands such that one of the dual variables $\pi_{s^d,d}$ (or $\pi_{t^d,d}$) is strictly positive. Where this is not possible, we can achieve this by any LP solving method, although this involves solving an auxiliary LP problem as shown in [12, 21]. Another way to handle this is by making use of some fundamental results of linear programming theory (complementary slackness property), that is, the constraints whose dual coefficient values are strictly positive are necessarily satisfied at equality (for the standard LP formulation), and showing that we obtain necessarily at least one binding constraint each time we solve the problem P_k. It can easily be seen that with respect to the dual formulation of P_k [26], there is at least one positive

value among the dual coefficients since:

$$\sum_{d \in D \setminus L_{k-1}} (\pi_{s^d,d} + \pi_{t^d,d}) \geq 1 \tag{13}$$

$$\pi_{s^d,d}, \pi_{t^d,d} \geq 0 \quad \forall d \in D. \tag{14}$$

We are thus sure that there must be at least one new demand saturated at each step. A straightforward method should be fixing the value of the last saturated demands to α for the upcoming computations. Afterward, one can classify demands according to the obtained value of max-min fairness in the right D_k. A final method is to test if demands satisfied at equality are saturated or not. Such tests add a single slack variable for each demand and test if it can take a (strictly) positive value or not, [23, 27]. One may remark that all the above methods place in D_k only those demands such that the corresponding traffic constraints have positive dual variables.

3.1.2 Some theoretical properties

In this section we study the optimality and polynomiality of our approach, and we highlight some properties of max-min fair multi-commodity flows and the max-min fair flow vector. We notice that the existence and the uniqueness of the MMF flow vector can be easily deduced from Proposition 1. We now consider the correctness of our iterative algorithm, that is, we prove (Theorem 1) the optimality (with respect to max-min fairness) of the multi-commodity flow computed by our Algorithm MMF and the polynomiality of the method:

Theorem 1. *The multi-commodity flow obtained at the end of the Algorithm MMF is max-min fair and it can be obtained in polynomial time.*

According to Proposition 1, we need to prove that the flow vector obtained at the end of the algorithm is leximin maximal. Then, since the Algorithm MMF is an application of Algorithm 1, the proof is straightforward in the light of the results presented in [21] regarding the leximin maximality of the vector achieved by Algorithm 1 and the polynomiality of the approach. □

We intend now to go further and to exhibit some particular properties related to max-min fair multi-commodity flows, beginning with some preliminary results. First, let us denote with S_k the set of saturated links at the end of the step k. Let us consider the set of multi-commodity flows satisfying all constraints associated with the problem P_k (step k). Obviously, for each multi-commodity flow there is at least one newly-saturated commodity flow and/or arc. Let us denote as D_k^* (resp. S_k^*) the intersection of all these sets of newly saturated commodity flows (resp. arcs) with respect to all possible multi-commodity flows obtained as solution of problem P_k. Then, we can state:

Lemma 1. *All sets D_k^* and S_k^* are not empty.*

Let us begin by considering the set D_k^* and suppose by absurdity that it is empty. Indeed, without loss of generality, let us suppose that there exist two multi-commodity flows Φ_1 and Φ_2 satisfying all constraints with respect to P_k (*i.e.,* at the end of step k), and having distinct sets of saturated flows at the same value α (denoted respectively Q_1 and Q_2 with $Q_1 \cap Q_2 = \emptyset$).

Clearly both Q_1 and Q_2 are included in the set of demands non-saturated at the end of step $k-1$ and given by $D \setminus L_{k-1}$. Let $\Phi = \frac{(\Phi_1+\Phi_2)}{2}$. It is obvious that the multi-commodity flow Φ satisfies capacity and flow conservation constraints associated with the problem P_k. Furthermore, it is not saturated in any commodity flow contained in $D \setminus L_{k-1}$. In these conditions, we increase at the same rate the flow value to each f_{ij}^d with respect to non-saturated commodities $(D \setminus L_{k-1})$ until the network becomes saturated. Obviously, α will be increased and will have a value strictly superior to that obtained by Φ_1 and Φ_2, which contradicts the fact that Φ_1 and Φ_2 are obtained as solutions of problem P_k during the k^{th} step. We have consequently proved the first part of the above lemma. The second part can be proved in a similar way. It will be noted that each multi-commodity flow solution of problem P_k necessarily saturates some arcs separating the extremities of the newly saturated demands in the graph. Using the same logic, that is, assuming the existence of two multi-commodity flows Φ_1 and Φ_2 satisfying all constraints with respect to P_k, and having distinct sets of saturated arcs at the end of step k, it is clear that $\Phi = \frac{(\Phi_1+\Phi_2)}{2}$ does not saturate any of them. We then can show that the multi-commodity flow Φ can be augmented, giving thus a better value for α. We can therefore state that the intersection of all these sets of arcs is necessarily not empty. □

Let us consider in the following some properties of the multi-commodity flow solutions obtained at the end of a given step of the algorithm with respect to saturated commodities and saturated arcs.

Lemma 2. *All multi-commodity flows obtained at the end of the Algorithm MMF saturate commodities in a unique order*[3]*.*

In the light of the Algorithm MMF, it can easily be seen that commodities are saturated in a given order, since this is fixed through the consecutive steps of the algorithm. What is not clear is the uniqueness of sets D_k computed by this algorithm. To demonstrate this we need to show that $D_k = D_k^*$. First, it is obvious that $D_k^* \subseteq D_k$. Let us now prove that $D_k \subseteq D_k^*$ also holds. This is straightforward since all demands added in D_k are such that the corresponding dual variable $\pi_{s^d,d} > 0$ (or $\pi_{t^d,d} > 0$) and according to the LP theory, the associated constraint is tight for all solutions of problem P_k and thus it is contained in D_k^*. □

Lemma 3. *All multi-commodity flows obtained at the end of the k^{th} step of the Algorithm MMF saturate all arcs in* $\bigcup_{1 \leq l \leq k} S_l^*$.

It is obvious that the solution obtained at the end of the k^{th} step of Algorithm MMF saturates all arcs in S_k^*, and since such a multi-commodity flow is a possible solution for all previous steps $(P_l, l < k)$, the result also holds for all arcs in S_l^*, for all $l < k$. We have proved that for any solution obtained at the end of the step k, we have $\bigcup_{1 \leq l \leq k} S_l^* \subseteq S_k$, concluding thus the proof of the lemma. □

We can now state and prove an immediate result concerning the number of steps of the Algorithm MMF.

[3]The commodities contained in the same D_k are interchangeable in the saturating order.

Lemma 4. *The Algorithm MMF terminates in at most* $Min\{|D|, |A|\}$ *steps.*

According to Lemma 1, at each step of the algorithm a non empty set of commodity flows and arcs are saturated. We cannot therefore have more than $|A|$ or $|D|$ distinct D_k, and consequently the number of steps cannot exceed $|A|$ or $|D|$. □

Let us now examine some other properties related to arc-connectivity in undirected networks. We also denote as first-level max-min fair multi-commodity flows all multi-commodity flows obtained as solution at the end of the first step of the Algorithm MMF.

Proposition 2. *All first-level max-min fair multi-commodity flows in a capacitated k-arc connected undirected network saturate in at least k arcs.*

Notice first that each first-level max-min fair multi-commodity flow is necessarily saturated in at least one commodity d. Furthermore, the terminal nodes s_d and t_d cannot be connected through paths containing only non-saturated arcs because the flow value of the commodity d cannot be increased. Consequently, all disjoint paths (at least k) between s_d and t_d are saturated. Which is to say that at least k arcs are saturated at the end of the first step[4]. □

In the same way, we note that the terminal nodes of saturated commodities are placed in two distinct/complementary sets linked by saturated arcs, thus defining a cut. At the end of the first step, let N_1 denote the subgraph obtained from the initial graph N when removing the saturated arcs. Similarly, at the end of the k^{th} step, let N_k denote the subgraph obtained when removing from N_{k-1} the newly-saturated arcs. At the end of the k^{th} step, the terminal nodes of newly saturated commodities are necessarily disconnected in the subgraph N_k and the number of its components is strictly superior to this in N_{k-1}. This is because these new commodities were not saturated before executing the most recent step, and consequently the respective terminal nodes were in the same component. This component cannot therefore remain connected in the new subgraph N_k, and consequently the number of components becomes strictly superior to that obtained at the end of the preceding step. So after each step, the number of components of the remaining subgraph is incremented by at least one. Since at the end of the first step the number of components is at least 2, it follows that after $k-1$ further steps the number of components becomes greater or equal to $k+1$. Combining this with the fact that a connected graph cannot be decomposed into more than $|V|$ components, we see that there cannot be more than $|V|-1$ decompositions of the network or steps of the algorithm, and have thus proved the following result:

Proposition 3. *The max-min fair flow vector in undirected networks has at most* $Min\{|D|, |A|, |V|-1\}$ *distinct values.*

3.2 Robustness of MMF multi-commodity flows

Let notice first that the MMF routing problem in a telecommunication network is quite similar to the MMF multi-commodity flow problem studied above, except that we should replace $\alpha, \lambda[d]$ with respectively $\alpha T_d, \lambda[d]T_d$ in all (three)

[4] A large number of commodities could be thus saturated during the first step.

traffic constraints (9, 10, 11). In the new formulation T_d gives the traffic value associated with traffic demand $d \in D$ and $\alpha, \lambda[d]$ give the satisfaction ratio of traffic requirements with respect to demands in D. It is worth noting that the max-min fair routing computing approach is able to cope with a homogeneous traffic increase/decrease across all demands and is particularly suitable for elastic traffic. More generally, the MMF solution provides robust routing schemes even for the general case when a set of traffic demands represented in a matrix have to be routed through a given network. It is well known that traffic prevision is a difficult task and errors are common in real-life situations. Our approach to route computation also addresses this problem through a robust routing scheme. Generally speaking different routing schemes do not address overload traffic situations in the same way. With respect to our MMF routing solution, we can show that variations in traffic within certain bounds are completely absorbed (we assume that routing is feasible, *i.e.,* $\overrightarrow{\lambda}[1] \geq 1$). As an illustration, let us suppose that the traffic for all demands $d \in D$ are increased within a given ratio r_d with respect to the traffic prevision (T_d). We are able to state that as long as $r_d \leq \lambda[d] - 1$, for all $d \in D$, the computed routing will remain feasible and all demands will be entirely routed. It is straightforward that the overload ratio vector $\{\lambda[d] - 1$, for all $d \in D\}$ is also MMF. In other words, the MMF routing provided by our algorithm guarantees for each demand a certain ratio of overload such that it is not possible to do better without decreasing the guaranteed overload ratio of other traffic demands with lower values.

4 Computational results for MMF routing

Our algorithm for computing max-min fair routing was implemented in C++ with CPLEX 8.0. All tests were run on a machine with the following configuration: Sun 4000, SOLARIS 2.5, 6 UltraSparc 167Mhz processors, 256 MB of RAM. We implemented both the arc-node and the arc-path models using a Simplex algorithm (CPLEX). We begin by comparing max-min fairness (as discussed in this paper) and throughput objectives and end the section with some computational results on CPU time.

4.1 Max-min fairness versus throughput

It seems to us more interesting to start by comparing max-min fairness (as discussed in this paper) and throughput objectives. It is obvious that these two objectives are often contradictory, so that maximizing network throughput as a primary objective may lead to a high degree of unfairness: in the worst case some commodities may get a zero throughput. Providing both efficiency and some form of fairness is a key objective of congestion control in telecommunication networks. To this aim, we solve three problems for each test network with respect to fairness and throughput criteria. The first problem is the traditional maximal flow problem. The second is also a maximal flow problem but with respect to first-level fair multi-commodity flows (*i.e.,* each commodity takes a value not less than $\overrightarrow{\lambda}[1]$). The third is the MMF problem as discussed in this paper. We have denoted the relative variations for the three throughput values above as $Gap1$ and $Gap2$, where $Gap1$ is the relative variation between maximal flow throughput and first-level MMF throughput, and $Gap2$ the variation be-

tween maximal flow throughput and MMF throughput. These were computed as $100 * (y - x)/y$ (that is to say as percentages). Two series of tests were run, one for random networks and the other for well-dimensioned realistic networks provided by France Telecom R&D.

4.1.1 Random networks

We have considered here six undirected random networks with respectively 15, 21, 26, 31, 36 and 41 nodes where the traffic matrix contains all possible demands (*i.e.,* each pair of nodes is associated with a demand) and arc capacities taking values uniformly in [1, 5]. Starting from a fully meshed network we remove arcs at random in order to compute different instances $Instance(n, i)$, $i = 0..5$, such that n gives the number of edges and i the meshing degree, *i.e.,* $i = 0$ is a full mesh, last, corresponding to $i = 5$, is a spanning tree. Some comparisons between $Gap1$ and $Gap2$ with respect to each case network are given in Figures 1-6.

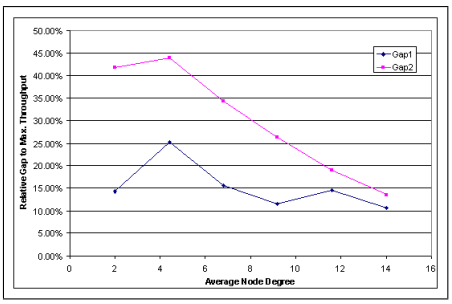

Figure 1: Gap1 and Gap2 for the 15-node networks

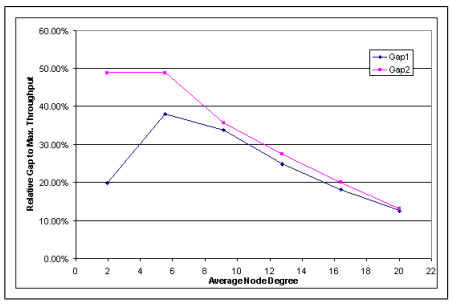

Figure 2: Gap1 and Gap2 for the 21-node networks

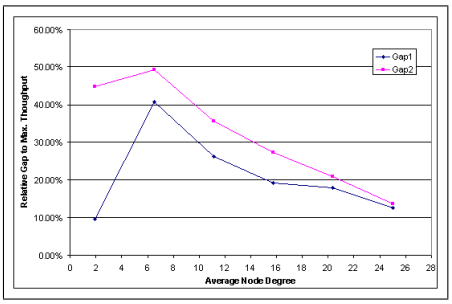

Figure 3: Gap1 and Gap2 for the 26-node networks

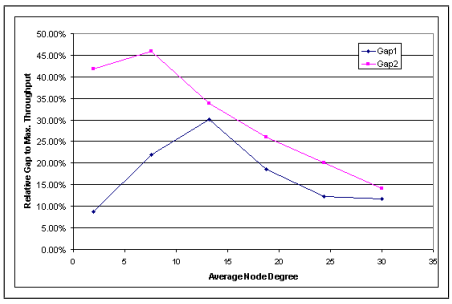

Figure 4: Gap1 and Gap2 for the 31-node networks

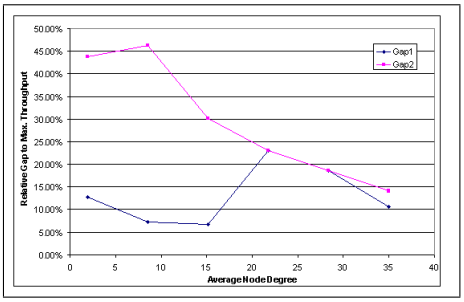

Figure 5: Gap1 and Gap2 for the 36-node networks

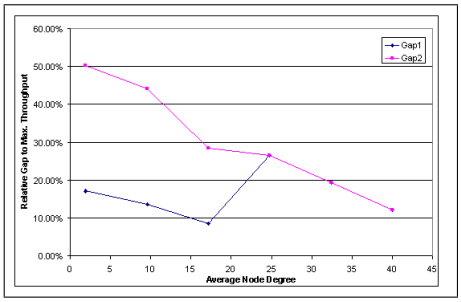

Figure 6: Gap1 and Gap2 for the 41-node networks

Perhaps not surprisingly, two straightforward remarks can be drawn from the obtained computational results: first, we have $Gap2 > Gap1 > 10\%$ for all highly-meshed networks, in particular complete graphs. We observe that max-min fairness (but not first-level max-min fairness) has a high cost in terms of throughput, and this is especially pronounced for unmeshed networks. Almost all the figures show that diminishing the density of networks (*i.e.,* getting to a spanning tree) means an accompanying significant increase in the difference between the two *gaps*, suggesting that first-level max-min fairness could give an acceptable compromise between fairness and throughput objectives for this kind of random network.

4.1.2 Well-dimensioned networks

It should be pointed out that solutions maximizing network throughput constantly neglect certain "costly" traffic demands, *i.e.,* numerous demands in both random and well-dimensioned networks have a throughput equal to zero, which makes them more or less unacceptable for telecommunication operators. The neglect of these "costly" demands can be explained simply by the fact that the routing paths for these demands in such networks are often long, thus consuming precious resources, and hence making them uninteresting when the aim is to maximize throughput. We studied the behavior of MMF using tests performed on well-dimensioned telecommunication networks. We then evaluated the relative variation $Gap1$ and $Gap2$ as reported in Figure 7.

In Table 1 we summarize the main characteristics of four undirected network instances that we used for our computations. All test networks were provided

Network	NET_26	NET_41	NET_70	NET_114
Nodes	26	41	70	114
Arcs	43	70	131	183
Demands	264	319	1687	1435

Table 1: Description of test networks

by France Telecom R&D and dimensioned to route completely the traffic associated with the demand matrix. However, in order to differentiate the first-level max-min fair routing from max-min fair routing and to avoid restrictions, the

networks are within certain limits over-dimensioned. This is due either to the need for additional capacity to withstand failures, or to the modularity of installed capacity (*i.e.,* the installed capacity is generally greater than the minimum capacity needed to route all traffic demands). As it can be seen in the following figure, the difference between $Gap1$ and $Gap2$ is often quite small. This can be essentially explained by the fact that in practice the realistic networks are not significantly over-dimensioned and do not leave too much room for additional traffic to be routed.

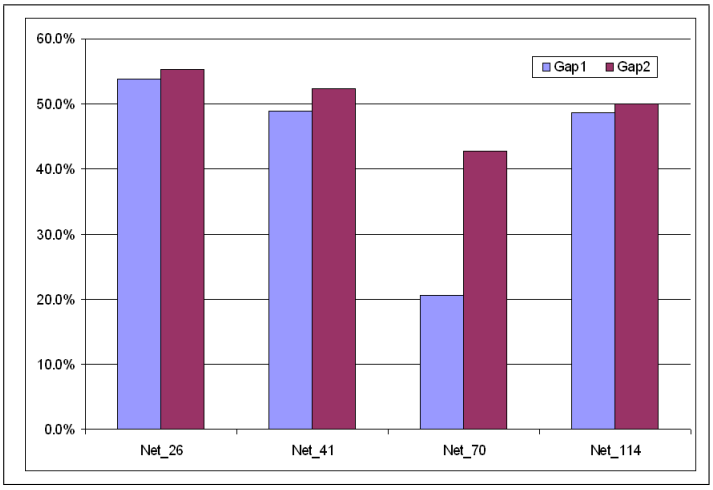

Figure 7: Gap1 and Gap2 for well-dimensioned networks

4.2 Computational time

We present here the calculation time of the max-min fair routing case for well-dimensioned networks. The implemented model is based on the arc-path formulation which is more convenient for realistic cases, since it allows the length of routing paths to be restricted. Let us recall that in the arc-path formulation, each path in the network is associated to a variable. Although this yields a large number of variables, a solution can in practice be obtained very efficiently through column generation (see for instance [8]). Finally, we also provide in Table 2 results on the percentages of single-path routed demands which could be interesting from an operational point of view.

Instances	CPU Time(s)	Mono-routed demands (%)
NET_26	1.81	89.39%
NET_41	2.78	87.46%
NET_70	27.09	92.14%
NET_114	76.20	97.42%

Table 2: Numerical results on max-min fair routing calculation.

5 Lexicographically minimum load networks

In this section, we consider the problem of load-balancing (the term is used in [13]) in a given network. We aim to distribute the load fairly among the network links while satisfying a given set of traffic constraints. More precisely, we aim not only to minimize the maximal load among links, but also to minimize lexicographically the sorted (non-increasing) load values through network links in order to obtain what is called a *lexicographically minimum load network* [21]. In contrast to the max-min fair multi-commodity flow problem, the problem here is to load the network fairly in the min-max sense. This problem arises in telecommunication networks when the operator needs to define routing with respect to a given traffic demand matrix such that the network load is fairly distributed among the network links. We consider the multi-commodity network and we focus on the splittable case. Some relevant work on balanced networks is presented in [13] where the authors propose a strongly polynomial approach for lexicographically-optimal balanced networks. The problem they address is allocating bandwidth between two endpoints of a backbone network so that the network is equitably loaded, and therefore it is limited to the case of single-commodity networks. How to generalize this result to multi-commodity networks is already shown in [21], but we consider here a large range of link load functions, especially non-linear, frequently used in the telecommunication area, and show that each of them can be reduced to the linear case (see also [25]). Let us begin by giving some definitions and notation.

5.1 Link load functions

A load function $Y(f)$ (linear or not) defined as $Y : A \rightarrow \mathbb{R}^+$ gives the load associated with each arc. An arc load function is generally closely related to the arc occupation ratio and thus to the flow traversing it. It can typically be expressed linearly as $\frac{f_a}{C_a}$, where $f_a = \sum f^d_{ij} : (ij) = a$ and C_a is the capacity of link a giving thus the relative load. Another currently used criterion (to maximize) is the residual capacity $C_a - f_a$. From the telecommunication point of view, the choice of relative load or absolute residual capacity as a criterion to optimize may reflect different robustness goals. For instance, if the main concern is to be able to cope with new unforeseen demands, keeping as much bandwidth available as possible on all links is preferable: this corresponds to maximizing the absolute network remaining capacity $\min_{a \in A} C_a - f_a$. But if the main goal is to address a proportional simultaneous increase across all demands, it is better to minimize the maximal relative arc load $\max_{a \in A} f_a/C_a$. Lexicographic optimization extends and improves these basic observations. The global network resources optimization can also involve non-linear link criteria, such as for instance the well-known Kleinrock function $\frac{f_a}{(C_a - f_a)}$ which takes values in $[0, +\infty)$. More general link load functions (to minimize) are commonly used, as $(\alpha - 1)^{-1}(1 - f_a/C_a)^{1-\alpha}$ or $(\alpha - 1)^{-1}(C_a - f_a)^{1-\alpha}$ $\alpha \in R^+ \setminus \{1\}$; (see e.g. [2, 16]). We show that, in any of these cases, it is possible to use one of the two previous linear examples to obtain optimal solutions.

5.1.1 Load vector and computing approach framework

This paragraph is devoted to defining the load vector, stating the problem and giving the main lines of the computing approach.

Definition 6. *Given a network (V, A, C), a multi-commodity flow f and a load function Y, the vector $\gamma = \{\gamma_a, a \in A\}$ whose components give the load value associated with each arc (i.e., $\gamma_a = Y(a)$) is called the load vector of multi-commodity flow f.*

A load vector is called *feasible* if there exists a feasible multi-commodity flow with the same load vector. Let F denote the set of all the feasible multi-commodity flows in network N with respect to a set of commodities D. We use the term *lexicographically minimum load network problem* to refer to the problem of computing a feasible multi-commodity flow such that its load vector is leximax minimal in the set of feasible load vectors with respect to F. Now, the lexicographically minimum load network problem can be stated as follows: *given a capacitated network $N = (V, A, C)$ and a set of commodities D with traffic values $\{T_d\}_{d \in D}$, find the leximax minimal load vector γ and a corresponding feasible multi-commodity flow.*

We are not going to detail the approach for solving the lexicographically minimum load network problem defined above, since it is quite similar to Algorithm MMF. This brings us to the main idea behind our approach (or behind any lexicographic optimization algorithm): the highest load value of the vector is to be minimized before the second highest, and inductively, the minimization for a component of the load vector is carried out only after components with worse values have been minimized. A link whose load has already been minimized is called *minimally loaded*. The algorithm terminates when all links are loaded. During each step we resolve an LP problem similar to P_k described for the Algorithm MMF. In the following we present two linear link load functions and give the mathematical formulation of the LP problem associated with the iterative approach.

5.1.2 Linear link load functions

As remarked above, two principal criteria are generally used to evaluate the load of a link in telecommunication networks. The first is the relative load $\frac{f_a}{C_a}$ that we need to minimize, and the second is the residual capacity $C_a - f_a$ that we need to maximize. When considering the relative load, the problem P_k (we

use the same term as for the Algorithm MMF) can be then stated as:

$$\sum_{j:(i,j)\in A} f_{ij}^d - \sum_{j:(j,i)\in A} f_{ji}^d = \begin{cases} Minimize & \alpha \\ T_d, i = s^d & \\ -T_d, i = t^d & \\ 0, otherwise & \end{cases} \quad \forall i \in V, \forall d \in D \tag{15}$$

$$\sum_{d\in D} f_{ij}^d \leq C_{ij} \quad \forall (i,j) \in A \setminus L_{k-1} \tag{16}$$

$$\sum_{d\in D} f_{ij}^d \leq \alpha C_{ij} \quad \forall (i,j) \in A \setminus L_{k-1} \tag{17}$$

$$\sum_{d\in D} f_{ij}^d = \gamma[i,j] C_{ij} \quad \forall (i,j) \in L_{k-1} \tag{18}$$

$$f_{ij}^d \geq 0 \quad \forall (i,j) \in A, \forall d \in D, \tag{19}$$

where constraints (15) are flow conservation constraints, (16) are capacity constraints and (17, 18) are load constraints. Finally, constraints (19) indicate that arc-flow values must be non-negative. Here L_{k-1} gives the set of links loaded through the first $k-1$ iterations of the algorithm. To conclude the iteration, we need to identify all links loaded to α by searching for the active constraints (17). Then, we set $\gamma[i,j] = \alpha$ for all these newly-loaded links. Through a process of reasoning similar to that used in the Algorithm MMF, it can be deduced that there is at least one such minimally loaded link at the end of each iteration. Furthermore, one can show that there are at most $|v|-1$ distinct load values in the obtained load vector.

Regarding the function $C_a - f_a$, we need to maximize lexicographically the residual capacity, and the residual load vector needs here to be leximin maximal. Then, the problem P_k can be formulated similarly to the above P_k, but changing the objective function to maximize and replacing load constraints (17, 18) by (20, 21):

$$C_{ij} - \sum_{d\in D} f_{ij}^d \geq \alpha \quad \forall (i,j) \in A \setminus L_{k-1} \tag{20}$$

$$C_{ij} - \sum_{d\in D} f_{ij}^d = \gamma[i,j] \quad \forall (i,j) \in L_{k-1}. \tag{21}$$

5.1.3 General link load functions

Let us consider first some increasing function ϕ and the system composed of functions $\phi \circ f_i$. It can easily be shown that the above results hold.

Proposition 4. *Let ϕ be an increasing function in R. Some x achieves a leximin maximal (resp. leximax minimal) vector γ with respect to system (1) if and only if x achieves a leximin maximal (resp. leximax minimal) vector for the corresponding system composed of functions $\{\phi \circ f_i\}$ given by $\phi(\gamma)$.*

Proposition 5. *Let ϕ be a decreasing function in R. Some x achieves a leximin maximal (resp. leximax minimal) vector γ with respect to system (1), if and only if x achieves a leximax minimal (resp. leximin maximal) vector for the corresponding system composed of functions $\{\phi \circ f_i\}$ given by $\phi(\gamma)$.*

We can now use these last results for computing lexicographically minimum loaded networks for all link load functions enumerated above.

Corollary 1. *Let $\alpha \in R^+ \setminus \{1\}$. Any multi-commodity flow whose load vector is leximax minimal for the linear load function $\frac{f_a}{C_a}$ is also leximax minimal for the generalized link load function $(\alpha-1)^{-1}(1-f_a/C_a)^{1-\alpha}$, and vice-versa.*

Corollary 2. *Let $\alpha \in R^+ \setminus \{1\}$. Any multi-commodity flow whose load vector is leximin maximal for the linear load function $C_a - f_a$ is leximax minimal for the generalized link load function $(\alpha-1)^{-1}(C_a-f_a)^{1-\alpha}$, and vice-versa.*

5.2 Computational results for lexicographically minimum load networks

We ran tests on the same realistic networks provided by France Telecom R&D. We considered only the f_a/C_a criterion and ran similar tests to those run for MMF routing. We are not reporting the CPU running time, since it is quite similar to this obtained for MMF routing.

We computed the load distribution obtained when seeking respectively to minimize the global network load, minimize lexicographically the load, and minimize the global network load while ensuring that link loads are below a given threshold determined by the first-level min-max load[5]. We show in Figure 8 the results obtained for the 26-node network; for the other networks the results are essentially similar. In Table 3, we show the dispersion and mean value for the load vectors obtained for the three cases, *i.e.,* minimum global network load, first-level min-max load and min-max load for the f_a/C_a criterion. It will again be noticed that the first-level minimum load approach allows a lower consumption of resources (see the columns in Mean); thus, it could give an acceptable compromise for the network manager.

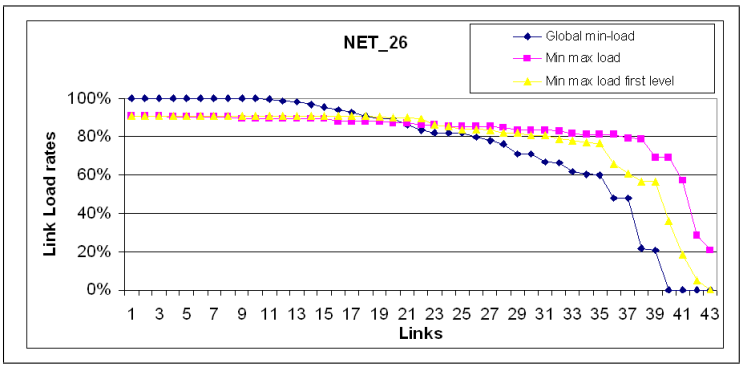

Figure 8: Comparison of load distributions for the 26-node network

[5]The first-level min-max load is defined in a similar way to the first-level max-min fair routing in Section 3.1.

Instances	Dispersion			Mean		
	MinGlobalLoad	1rstLevel	MinMaxLoad	MinGlobalLoad	1rstLevel	MinMaxLoad
NET_26	0.311	0.224	0.142	0.733	0.773	0.815
NET_41	0.284	0.213	0.157	0.730	0.764	0.813
NET_70	0.358	0.348	0.116	0.406	0.411	0.534
NET_114	0.286	0.259	0.076	0.536	0.543	0.648

Table 3: Comparison of load minimization approaches.

6 Concluding remarks

In this paper we have provided a thorough study of MMF multi-commodity flows in a capacitated network and lexicographically minimum load networks. We have examined some theoretical properties of MMF multi-commodity flows and provided some discussion on the robustness of such flows. We have shown that the problem can be easily transposed for the lexicographically minimum load problem in networks. We are most interested by applications to telecommunication networks with respect to max-min fair routing and lexicographically minimum load networks. We have considered several cases of load functions and show how each case can be handled. Some computational results illustrate the trade-off between throughput achieved by MMF flows and maximal flows when varying the network's density, as well as the computational efficiency of the approach itself. Similar numerical results are provided for minimal network load problems. In both cases, an interesting observation has to be highlighted: the first-level lexicographic optimization seems sufficient in practice to obtain results very close to those obtained in complete lexicographic optimization. For instance, when dealing with maximal flow issues, a network manager could first optimize the minimal flow value over all demands, and then maximize the global throughput, while obtaining very satisfactory results.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, New Jersey, 1993.

[2] W. Ben Ameur, N. Michel, E. Gourdin et B. Liau. Routing Strategies for IP Networks. *Telektronikk*, 2/3, pp 145-158, 2001.

[3] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Engelwood Cliffs, N.J., 1992.

[4] S. Boulahia-Oueslati. *Qualité de Service et Routage des Flots Élastiques dans un Réseau Multiservice.* PhD thesis, ENST, Paris, France, November 2000.

[5] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[6] A. Charny, D. D. Clark and R. Jain. Congestion control with explicit rate indication. In *Proc. of IEEE International Conference on Communications*, 1995.

[7] S. Chen and K. Nahrstedt. Maxmin Fair Routing in Connection-oriented Networks. In *Proc. of Euro-Parallel and Distributed Systems Conference, pp. 163-168, Vienna, Austria,* July, 1998.

[8] G. Desaulniers, J. Desrosiers, M. M. Solomon (eds.) *Column Generation.* Springer-Verlag, N.Y, June 2005.

[9] G. Fodor, G. Malicsko, M. Pioro and T. Szymanski. Path Optimization for Elastic Traffic under Fairness Constraints. In *Proc. of ITC 2001.*

[10] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, N.J., 1962.

[11] R. M. Freund and S. Mizuno. Interior Point Methods: Current Status and Future Directions. In *High Performance Optimization*, H. Frenk et al. (eds.), Kluwer Academic Publishers, pp. 441-466, 2000.

[12] R. M. Freund, R. Roundy, and M.J. Todd. Identifying the Set of Always-Active Constraints in a System of Linear Inequalities by a Single Linear Program. *Working paper 1674-85*, Massachusetts Institute of Technology (MIT), Sloan School of Management, 1985.

[13] L. Georgiadis, P. Georgatsos, K. Floros, S. Sartzetakis Lexicographically Optimal Balanced Networks. In *Proc. Infocom'01*, pp. 689-698, 2001.

[14] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, vol. 49, pp. 237-252, 1998.

[15] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in Routing and Load Balancing. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, 1999.

[16] L. Kleinrock. *Communications Nets: Stochastic Message Flow and Delay*, McGraw-Hill, New York, 1964.

[17] Q. Ma. *Quality-of-Service Routing in Integrated Service Networks.* PhD thesis, Carnegie Mellon Univ., Pittsburgh, USA, Jan. 1998.

[18] N. Megiddo. Optimal Flows in Networks with Sources and Sinks. *Mathematical Programming*, 7, 1974.

[19] L. Massoulié and J.W. Roberts. Bandwidth sharing: objectives and algorithms. In *Proc. of IEEE INFOCOM'99*, 1999.

[20] T. Bonald and L. Massoulié. Impact of fairness on Internet performance. In *Proc. of ACM SIGMETRICS 2001*, 2001.

[21] D. Nace, J. B. Orlin. Lexicographically Minimum and Maximum Load Linear Programming Problems, *to appear in Operations Research.*

[22] D. Nace. A Linear Programming based Approach for Computing Optimal Splittable Fair Routing. In *Proc. of The Seventh IEEE Symposium on Computers and Communications 2002, ISCC'2002. pp. 468-474*, July 2002, Taormine, Italy.

[23] D. Nace, L.N. Doan, Eric Gourdin and Bernard Liau. Computing Optimal Max-min Fair Resource Allocation for Elastic Flows, *to appear in IEEE Transactions on Networking, December 2006.*

[24] D. Nace, N-L. Doan, Some Results on Max-min Fair Routing, INOC 2003, Evry, 27-29 October 2003. *In Proc. of INOC 2003, Evry, October 2003.*

[25] D. Nace, O. Klopfenstein, On the Lexicographically Minimum Loaded Networks, *In Proc. of INOC 2005, mars 2005, Lisbone Portugal.*

[26] M. Pioro, P. Nilsson, E. Kubilinskas and G. Fodor. On efficient Max-min Fair Routing ALgorithms. *In Proc. of ISCC 2003, Antalya, Turkey, July 2003.*

[27] M. Pioro and D. Medhi, *Routing, Flow and Capacity Design in Communication and Computer Networks*, Morgan Kaufmann Publishers (2004).

[28] B. Radunovic, J.-Y. Le Boudec. A Unified Framework for Max-Min and Min-Max Fairness with Applications. *Proceedings of 40th Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, October 2002.