A polynomial algorithm for solving a general max-min fairness problem

Artur Tomaszewski*

Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warszawa, Poland

SUMMARY
An iterative algorithm of polynomial complexity is presented that solves a max-min fairness problem which is often encountered while dealing with traffic routing or capacity allocation problems. The algorithm does not depend on any specific traffic routing problem formulation and is sufficiently general to be applied to a broad class of traffic routing and capacity allocation problems. The correctness of the algorithm and its complexity are formally proved. Copyright © 2005 AEIT.

1. INTRODUCTION

Probably, the most commonly considered network design problem is routing of traffic streams in the capacitated network (cf. Figure 1) [1]. This problem is formulated as follows. A set of network links and their size (capacity) are given. A set of traffic streams, their size and their origin and destination nodes are also given. The objective is to determine routes for traffic streams, the size of flows that are assigned to these routes and the total size of flows that load the links of the network.

In general, one can obtain either a satisfactory or an unsatisfactory solution to the above problem (cf. Figure 2). In the first case, all traffic streams are fully admitted, although it still might have to be decided how much unused capacity should be left on each network link (cf. Figure 2(a)). In the second case, not all traffic streams are fully admitted because either the capacity of links is not sufficient or the applied routing optimisation algorithms are not good enough (cf. Figure 2(b)).

If network capacity is insufficient to fully accommodate all traffic streams while providing the required grade of service, the task to determine what fraction of every stream to admit into the network may be an important part of the traffic routing problem. Several strategies can be applied that decide on the amount of admitted traffic [1]. The strategies are oriented either at maximising network revenue or at providing some form of fairness to individual traffic streams.

An example of strategy which is oriented at maximising the revenue is to maximize the sum (optionally—the weighted sum) of admitted flows’ sizes. In general, the ‘revenue-oriented’ approaches (R) may lead to the situation, where little traffic is admitted from particular traffic streams. Usually this is not acceptable and for this reason the ‘fairness-oriented’ strategies are used.

The most common ‘fairness-oriented’ approach is to admit equal amount of traffic from every stream. Depending on whether this amount is supposed to be an absolute value or a fraction of a traffic stream size, either absolute fairness (AF) or relative fairness (RF), respectively, is provided. Unfortunately, both these approaches can result in poor network capacity utilisation since for many streams much more traffic could still be admitted than this actual amount. Thus, one of the alternative approaches, based on the so-called max-min fairness (MMF) condition [2], is to admit as much traffic as possible from every traffic stream while making the smaller admitted amounts as large as possible. The resulting max-min fair vector of flow sizes is characterised by the property that an attempt to
Figure 1. Traffic routing problem.

Figure 2. Satisfactory (a) and unsatisfactory (b) solutions.
increase the size of any flow would require the reduction of some flow of smaller size. Still another popular fairness-related notion is the so-called proportional fairness (PF) [3]: a vector of flow sizes is proportionally fair if for any other vector of flow sizes the aggregate of proportional differences of their individual flow sizes is zero or negative (this requires that the total of the logarithms of flow sizes be maximised). In practice, this strategy provides the trade-off between the fairness and the revenue since it favours traffic streams that are assigned to shorter paths.

The difference between different strategies of deciding on the amount of admitted traffic is illustrated in Figure 3. The network consists of two links of a given size and there are three traffic streams. For each flow size assignment strategy the chart shows the resulting flow size of every traffic stream. Substantial differences can be observed between the results obtained applying different traffic admission strategies.

A number of approaches have been proposed to solve the so-called max-min fair traffic routing problem, that is the problem of traffic routing with the objective to determine such a flow size vector that fulfils the MMF condition. For example, in Reference [4] an iterative algorithm is proposed which can be applied whenever the original traffic routing problem is formulated as a linear programming programme.

Often, the MMF condition can be incorporated directly into a traffic routing problem formulation. However, even if the original formulation corresponds to an easily solved linear programming problem, the result might be a hard-to-solve mixed-integer programme (in fact it can be NP-complete [5]). As an example, consider the problem of assigning flows to predefined network paths when the capacities of network links (edges) are given. The objective is to maximise the total flow and to provide the max-min fair distribution of flow sizes.

**Problem 1:** Max-min fair path flows optimization.

**indices**
- \( e = 1, \ldots, E \) links
- \( p = 1, \ldots, P \) paths

**constants**
- \( \delta_{ep} \) equals 1 if link \( e \) is used by path \( p \); 0, otherwise
- \( c_e \) capacity of link \( e \)

**variables**
- \( x_p \) flow allocated to path \( p \)
- \( y_e \) total flow allocated to link \( e \)
- \( z_e \) maximal flow on link \( e \)
- \( \theta_{ep} \) 0, if edge \( e \) is saturated and the flow allocated to path \( p \) is maximal on \( e \); 1, otherwise

**objective**

\[
\max \sum_p x_p
\]

**constraints**

\[
\begin{align*}
\sum_p \delta_{ep} x_p &= y_e & e &= 1, 2, \ldots, E \\
y_e &\leq c_e & e &= 1, 2, \ldots, E \\
c_e \theta_{ep} &\geq z_e - y_e & e &= 1, 2, \ldots, E, p &= 1, \ldots, P \\
\sum_p \delta_{ep} (1 - \theta_{ep}) &\geq 1 & p &= 1, 2, \ldots, P \\
x_p \theta_{ep} &\leq z_e & e &= 1, 2, \ldots, E, p &= 1, \ldots, P \\
c_e x_p &\geq z_e - x_p & e &= 1, 2, \ldots, E, p &= 1, \ldots, P \\
x_p &\geq 0 & p &= 1, 2, \ldots, P \\
z_e &\geq 0 & e &= 1, 2, \ldots, E \\
\theta_{ep} &\in \{0, 1\} & e &= 1, 2, \ldots, E, p &= 1, \ldots, P
\end{align*}
\]

Let \( \{1, 2, \ldots, E\} \) be a set of edges and \( \{1, 2, \ldots, P\} \) be a set of paths. Let \( c_e \) denote the capacity of edge \( e \) and \( x_p \) denote the size of the flow assigned to path \( p \). Each feasible flow size assignment must satisfy condition \( \sum_p \delta_{ep} x_p \leq c_e \) for \( e = 1, 2, \ldots, E \) (\( \delta_{ep} \) equals 1 if link \( e \) belongs to path \( p \); 0, otherwise). It can be shown that a path flows vector \( x = (x_1, x_2, \ldots, x_P) \) satisfies the MMF condition, if and only if, for each path \( p \) there exists edge \( e \) used by \( p \), such that \( e \) is saturated and the flow \( x_p \) on \( p \) is maximal among all paths that use edge \( e \). This condition can be introduced into the problem formulation in the form of a number of...
additional constraints. Let $\theta_{e p}$, $e = 1, \ldots, E$ and $p = 1, \ldots, P$, be binary variables such that $\theta_{e p}$ should equal zero when edge $e$ is saturated and the flow of path $p$ is maximal on $e$. The path flows $x_p, p = 1, 2, \ldots, P$ that solve the mixed linear-integer programme of Problem 1 satisfy the MMF condition.

The rest of the paper is devoted to the answers to the following questions: How to formalise and generalise the problem of finding a max-min fair solution? Is there a general method that can be used to solve this problem? Is it possible to guarantee reasonable (i.e., polynomial) computational complexity of the solution algorithm?

2. PROBLEM AND ALGORITHM

Let $Y$ be a set of $m$-vectors. For each vector $y \in Y$, $y = (y_1, y_2, \ldots, y_m)$, let $(y) = (y(1), y(2), \ldots, y(m))$ denote the vector obtained from $y$ by rearranging its components in the non-decreasing order. This means that $y^{(1)} \leq y^{(2)} \leq \ldots \leq y^{(m)}$ and there exists a permutation $\pi$ of the set of indices such that $y^{(i)} = y_{\pi(i)}$ for $j = 1, 2, \ldots, m$. Vector $y$ fulfils the MMF condition with respect to set $Y$ if the following condition is satisfied:

$$y^{(i)} > y^{0}_{(i)} \Rightarrow \exists j < i \ y^{(j)} < y^{0}_{(j)} \ \forall \ y \in Y$$

This condition states (cf. [2]) that if elements of each vector from $Y$ are sorted in a non-decreasing order, vector $y^{(0)}$ is lexicographically greater than any other vector from $Y$ (cf. Figure 4).

The general max-min fairness problem can be formulated as follows. Let $X$ be the solution space of a given optimisation problem. Let $x$ denote a solution to this problem, $x \in X$. Let function $f, f : X \rightarrow \mathbb{R}^m$, be used to

![Figure 4. Solution sorting and mix-min fairness.](image-url)
characterise solutions to $\Pi$. Let $y$ denote an outcome vector characterising solution $x$ to $\Pi$, $y = f(x)$. Let $Y$ be a set of all outcome vectors corresponding to solutions to $\Pi$: $Y = \{ y : y = f(x), x \in X \}$. The problem consists in finding a solution vector $x^0 \in X$ and the corresponding outcome vector $y^0 \in Y$, $y^0 = f(x^0)$, which fulfils the MMF condition (Equation 2.1) with respect to set $Y$ defined by function $f$.

The general max-min fairness problem is related to the so called ordered weighted averaging (OWA) problem [6] which is to maximise the weighted sum of outcome variables when the weights are assigned to ordered outcome variables, that is to the worst value, to the second worst value and so on. If problem $\Pi$ is a linear programming problem and weights are non-increasing, OWA can be solved as an extended linear programme with $O(m^2)$ additional variables and constraints [6]. It is easy to see that OWA solves exactly a max-min fairness problem if differences between weights are large enough; this however may lead to serious numerical problems.

Another approach to solving the max-min fairness problem in case when $\Pi$ is a linear programming problem is proposed in References [7,8]. There the problem is considered in the context of general linear programming multiple criteria problems. The proposed solution approach is based on a modification of the simplex algorithm, which is oriented towards the sequential rearrangement of the simplex tableau.

In this paper, an iterative algorithm of polynomial complexity is presented that solves the max-min fairness problem exactly and does not require that $\Pi$ be a linear programming problem. It is only assumed that the solution space $X$ of problem $\Pi$ is convex and that function $f$ is concave. The algorithm is very general; it does not depend on any specific traffic routing problem formulation and can be applied to a broad class of traffic routing and capacity allocation problems.

The algorithm is iterative (cf. Figure 5). In a single iteration a value of at least one outcome variable is determined and assigned.

The variables are assigned their values in a non-decreasing order. The assignment is done in two steps (cf. Figure 6). In Step 1, a new, better value of the common lower bound of still unassigned outcome variables is determined. In Step 2, for each unassigned outcome variable it is checked if the determined lower bound is also the upper

Figure 5. General idea of the algorithm.
bound of this variable; if so the variable is assigned this value.

Let \( I \) be the set of all indices of the outcome variables. Let \( I_k' \) be the set of indices of outcome variables that still have their values undetermined at the beginning of iteration \( k \). Let \( \gamma_k \) denote a common lower bound on the values of these variables, which is determined in iteration \( k \). Let \( I_k \) be the set of indices of these outcome variables that have their values assigned in iteration number \( k \). The pseudocode of the algorithm is presented below.

**Algorithm 1: Solves a general max-min fairness problem.**

Step 0: Put \( y^0 := 0 \), \( k := 1 \) and \( I_k' := I \).

Step 1: If \( I_k' = \emptyset \) then stop \((x^0, y^0)\) are the optimal solution of the problem). Else, solve problem \( P(I_k', y^0) \) and denote the resulting optimal solution by \((x^0_k, y^0_k, r_k)\).

Step 2: Put \( I_k := \emptyset \). For each index \( i \in I_k' \) such that \( y^0_i = \gamma_k \) solve test problem \( T(I_k', y^0_i, r_k) \), denote the resulting optimal solution by \((x^0_i, y^0_i, r^0_k)\) and if the optimal objective value \( r^0_k \) equals \( \gamma_k \) put \( I_k := I_k \cup \{i\} \).

Step 3: Put \( x^0 := x^0_k, y^0 := y^0_k \) and \( I_{k+1} := I_k \). Put \( k := k + 1 \) and go to Step 1.

An elementary sub-problem \( T(I', y^0, r^0, i) \) that allows to check in Step 2 if a new lower bound \( \gamma^i \) of an outcome variable \( y_i \) is also its upper bound (if it is, \( y_i \) is a so-called blocking variable), can be formulated as a convex mathematical programming problem as follows:

\[
\text{maximize} \quad \gamma \\
\text{subject to} \quad y_j \geq \gamma \quad j \in I' \\
y_j = y^0_j \quad j \in I - I' \\
y = f(x) \\
x \in X.
\]

3. PROOF AND COMPLEXITY

To prove the correctness of the proposed algorithm the following assertion will be proved first.

**Proposition 1:** If set \( X \) of solution vectors of problem II is convex and function \( f \) is concave then sequence \( \{y^k\} \) generated by Algorithm 1 is strictly increasing.

**Proof.** Consider solution vectors \( x^0_i, i \in I_{k+1} \), generated by the algorithm in Step 2 of iteration \( k \). Each of these vectors satisfies (2.3b–f). Because set \( X \) is convex and function \( f \) is concave, the convex combination \( x = \sum_{i \in I_{k+1}} x^0_i \) satisfy (Equation 2.3b–f). Since for each of these vectors \( f_i(x^0_i) = y^0_i > y^0_k \) and \( f_j(x^0_j) > y^0_k \) hold for all \( i, j \in I_{k+1} \), then \( f_i(x) > y^0_k \) for all \( i \in I_{k+1} \). Thus \( y \) and \( \gamma \) solve problem \( P(I_{k+1}, y^0_{k+1}) \) and \( y^0_{k+1} > y^0_k \).

Now, the correctness of the proposed algorithm can be proved.

**Proposition 2:** If set \( X \) of solution vectors of problem II is convex and function \( f \) is concave then Algorithm 1 generates a max-min fair solution to problem II.
Proof. Let us assume that the algorithm generates the solution vector \( \mathbf{x}^* \) of problem \( \Pi \) and the outcome vector \( \mathbf{y}^0 \), and that vector \( \mathbf{y}^0 \) does not fulfil the condition (Equation 2.1). Thus, there exists solution vector \( \mathbf{x} \in X \), outcome vector \( \mathbf{y} = f(\mathbf{x}) \) and index \( i \in I \) such that:

\[
y_{(i)} > y_{(i)}^0 \land \forall j < i \ y_{(j)} = y_{(j)}^0
\]  

(3.4)

Since \( \mathbf{y}^0 \) is the result of the algorithm then \( y_{(i)}^0 > y_{(i)}^k \) for some value \( k \). Let \( z_i(\mathbf{y}) \) denote the index of the \( i \)-th smallest element of an outcome vector \( \mathbf{y} \); thus \( y_{(i)} = y_{z_i(\mathbf{y})} \). Let \( z_i^0 \) and \( z_i \) be short forms of \( z_i(\mathbf{y}^0) \) and \( z_i(\mathbf{y}) \). Let us define for each value \( m \), \( 1 \leq m \leq k \), a pair of sets \( Q_m^0 \) and \( Q_m \) as follows:

\[
Q_m^0 = \{ x^0(j) : y_{(j)}^0 = y_{(j)}^m \} \quad \text{and} \quad Q_m = \{ z(j) : y_{(j)} = y_{z(j)} \}
\]

One can notice that \( Q_m^0 = I_m \). Due to constraints (Equation 2.3): (a) for each value \( j < i \) index \( z(j) \) belongs to some \( Q_{m_j} \), (b) for each value \( m \leq k \) the number of elements in \( Q_m \) is the same as in \( Q_m^0 \), and (c) the number of elements in set \( Q_k^0 \) is greater than in set \( Q_k \) (since \( y_{(i)} > y_{(i)}^k \) then \( z(i) \notin Q(k) \) and \( x^0(i) \notin Q_k^0 \)). Thus there must exist value \( m \), \( m < k \), such that \( Q_m^0 \neq Q_m \).

Let \( n = \min \{ m : m \leq k, Q_m^0 \neq Q_m \} \). Then: (a) there exists index \( j \) such that \( j \in Q_n^0 \) and \( j \notin Q_n \), (b) for each value \( m < n : Q_m = Q_m^0 = I_m \). Thus \( y_h = y_{(j)}^n \) for \( m < n, h \in I_n \), and, since sequence \( \{ y_{(j)}^n \} \) is strictly increasing, \( y_{(j)}^n > y_{(j)}^0 \) for other values of index \( h \), in particular for \( h = j \). Therefore vector \( \mathbf{x} \) is a feasible solution of problem \( T(I', y^{0,i'}) \). This leads to the contradiction since the value of \( y_h \) delivered by \( T(I', y^{0,i'}) \) cannot be greater than \( y_{(j)}^n \). \( \blacksquare \)

A by-product of the above reasoning is the following assertion.

Proposition 3: If the set of solution vectors \( X \) of problem \( \Pi \) is convex and function \( f \) is concave then there is only one max-min fair outcome vector \( \mathbf{y}^0 \in Y \).

This proposition can be proved in a similar way as the correctness of the proposed algorithm was shown: the starting point is to assume that there is an outcome variable vector \( \mathbf{y} \in Y \) such that \( \mathbf{y} \neq \mathbf{y}^0 \) and \( \langle \mathbf{y} \rangle = \langle \mathbf{y}^0 \rangle \).

The following reasoning shows that the proposed algorithm solves a max-min fairness problem in a polynomial number of steps with respect to the number of outcome variables. The total number of steps depends on the number of outer iterations (composed of Step 1 and Step 2) and the number of inner iterations in Step 2. Since in each outer iteration at least one outcome variable is assigned a value, the number of outer iterations is bound by the number of variables. In each outer iteration problem \( \mathbf{P}(I', \mathbf{y}^0) \) is solved once and a number of inner iterations is performed. In Step 2 a single inner iteration consists of solving problem \( T(I', y^{0,i'}) \) for a single unassigned outcome variable. Thus the number of inner iterations in a single execution of Step 2 does not exceed the number of outcome variables. Therefore the total number of times either of the two elementary problems is solved is equal to \( O(1/2m^2) \).

This also shows that the max-min fairness problem is not considerably more difficult than the original problem \( \Pi \) since neither problem \( \mathbf{P}(I', \mathbf{y}^0) \) nor problem \( T(I', y^{0,i'}) \) is more difficult than problem \( \Pi \): in both cases just the values of particular outcome variables are constrained with auxiliary lower and upper bounds.

The proposed algorithm has been successfully applied to solving complex problems of robust network design [9]. There, several extensions of the algorithm were suggested to increase its computational performance. In particular, since in practice an outcome variable tested in Step 2 rarely appears to be blocking, in Step 2 one can run inner iterations only until the first blocking outcome variable is detected. Additionally, if \( \Pi \) is a linear programming problem, in Step 2 of the algorithm it is not necessary to solve problems \( T(I', y^{0,i'}) \) to detect the blocking outcome variables; instead, it is enough to inspect the optimal values of the dual variables of problem \( \mathbf{P}(I', \mathbf{y}^0) \) which is solved in Step 1.

REFERENCES

AUTHOR’S BIOGRAPHY

Artur Tomaszewski was born in 1967 in Warsaw. He received the M.Sc. and Ph.D. degrees in Telecommunications in 1990 and 1993, respectively, both from the Warsaw University of Technology. Since 1993, he has been an assistant professor at the Division of Switching and Computer Networks. His research interests focus on network architectures and network optimization methods. He is an author and co-author of about 30 papers on core and access network optimization, network planning processes and methods, and network planning support tools.