Lexicographically Minimum and Maximum Load Linear Programming Problems

Dritan Nace
Laboratoire Heudiasyc UMR CNRS 6599,
Université de Technologie de Compiègne,
60205 Compiègne Cedex, France,
nace@utc.fr
and James B. Orlin*
MIT Sloan School of Management,
Cambridge, MA 02139, USA
jorlin@mit.edu

August 19, 2004

Abstract

In this paper we introduce the lexicographically minimum load linear programming problem and we provide a polynomial approach followed by the proof of correctness. This problem has applications in numerous areas where it is desirable to achieve an equitable distribution or sharing of resources. We consider the application of our technique in the problem of lexicographically minimum load in capacitated multi-commodity networks and discuss a special non-linear case, the so-called Kleinrock load function. We next define the lexicographically maximum load linear programming problem and deduce a similar approach. An application in the lexicographically maximum concurrent flow problem is depicted followed by a discussion on the minimum balance problem as a special case of the lexicographically maximum load problem.

1 Introduction

In this paper we define the lexicographically minimum load linear programming problem and propose a polynomial approach to solve it. This problem has applications in numerous areas where it is desirable to achieve an equitable distribution or sharing of resources and thus closely related to maxmin or minmax optimization problems, widely studied in literature. We first introduce the lexicographically minimum load linear programming problem and define the load vector.

*This work was supported in part by Office of Naval Research grant ONR N00014-98-1-0317.

Consider the following linear programming feasibility problem

$$\sum_{j=1}^{m} a_{ij}x_j \leq b_i \quad \forall i \in 1..m$$
$$Dx = d$$
$$x \geq 0,$$

where $b_i > 0$ for all i and the system $\{Dx = d; x \geq 0\}$ is feasible. We say that γ is a *load vector* if each of its values is real valued number or negative infinity and if there is a feasible solution to

$$\sum_{j=1}^{m} a_{ij}x_j \leq \gamma_i b_i \quad \forall i \in 1..m$$
$$Dx = d$$
$$x \geq 0.$$

Let $\overrightarrow{\gamma}$ be the vector γ with its indices reordered so that the components are in non-increasing values. We want to find a load vector γ that is leximax minimal, that is $\overrightarrow{\gamma}$ is lexicographically minimum, as defined in [8]:

Definition 1 *A vector $\gamma \in X \subset R^n$, is said to be leximax minimal if for every vector $\gamma' \in X$, $\overrightarrow{\gamma}$ is lexicographically less than or equal to $\overrightarrow{\gamma'}$.*

The paper is organized as follows: in Section 2, we provide a mathematical framework and introduce an iterative algorithm for solving the lexicographically minimum load linear programming problem. Next, we provide a proof of the correctness of our approach. In Section 3, we show how to determine a lexicographically minimum load in capacitated multi-commodity networks. We consider the case of a linear load function as well as the case of Kleinrock load function. We show that any leximax minimal load vector for one is also leximax minimal for the other. In Section 4, we extend our results to lexicographically maximum load linear programming problems, and we discuss its application to a generalized version of the concurrent flow problem. In the last section, we show that the minimum balance problem, as defined by Schneider and Schneider ([10]), is a special case of the lexicographically maximum load problem.

2 A polynomial approach

We show in the following how to compute a lexicographically minimum load vector by solving a sequence of linear programs. We accomplish it as follows:
Algorithm 1. Computing lexicographically minimum load vector.
Step 1.
Find a minimum scalar value α^* such that $\gamma_i = \alpha^*$ for all i is a feasible load

vector. We do this by solving the linear program (P_1):

$$Minimize \quad \alpha$$

$$\sum_{j=1}^{m} a_{i,j} x_j \leq \alpha b_i \quad \forall i \in 1..m \tag{1}$$

$$Dx = d \tag{2}$$

$$x \geq 0. \tag{3}$$

If $\alpha^* = -\infty$, then every variable γ_i gets the load of negative infinity and the algorithm ends here, otherwise go to Step 2.
Step 2.
2.a.) Determine a feasible vector x for linear program (1) to (3) such that x has the fewest number of binding constraints among the constraints $\sum_{j=1}^{m} a_{ij} x_j \leq \alpha^* b_i$ for $i = 1$ to m. This can be accomplished by multiplying constraints (1), (2) and (3) by the decision variable $\lambda \geq 1$ and replacing λx by y. We then add "slack variables" s that are bounded above 1. The resulting problem (B_1) is as follows:

$$Maximize \quad \sum_{i=1}^{n} s_i$$

$$\sum_{j=1}^{m} a_{ij} y_j - \lambda \alpha^* b_i + s_i \leq 0 \quad \forall i \in 1..m \tag{4}$$

$$Dy - \lambda d = 0 \tag{5}$$

$$0 \leq s_i \leq 1, \lambda \geq 1, y \geq 0. \tag{6}$$

2.b.) If constraint i of (1) must be tight with respect to the x variables, then the only solution with respect to the transformed variables has $s_i = 0$. After determining the constraints for which $\sum_{j=1}^{m} a_{ij} x_j = \alpha^* b_i$, we let L denote the indices of the tight constraints and let $\gamma[k] = \alpha^*$ for all $k \in L$. Transfer these constraints to (2), and iterate between Steps 1 and 2 until no inequality constraints in (1) remains.

In all, at most $2m-1$ linear programs are solved throughout this algorithm. Notice that if constraint i is not necessarily tight, then one can choose a value of λ so large in the transformed problem so that we can set $s_i = 1$. So, constraint i for (1) is not binding if and only if $s_i = 1$ in an optimal solution for linear program (4) to (6). Notice also that if all of constraints (i.e., $i \in 1..m$) are binding at the end of the first iteration, the obtained solution is a totally balanced one.

Correctness proof. We next show the correctness of our approach. In **Theorem 1** we prove formally the optimality of the computed load vector.

Proposition 1 *The load vector computed by Algorithm 1 is unique.*

Proof. First, any solution obtained at the end of a given iteration k is necessarily a feasible solution for each problem P_s solved through previous iterations of the algorithm ($s < k$), since all relevant constraints are satisfied and

the value of the objective function is the same. Obviously this holds for the solution obtained at the end of the algorithm. Further, the uniqueness of the load vector obtained at the end of the algorithm follows from the uniqueness of the set of binding constraints with respect to the optimal load value α^* obtained during each iteration. ♦

We have thus shown that the load vector computed by our algorithm is unique. We next deduce its optimality, that is, we show γ is leximax minimal.

Theorem 1 *The load vector obtained at the end of Algorithm 1 is leximax minimal and it can be obtained in polynomial time.*

Proof. We suppose that there exists a feasible load vector γ^* such that γ^* is leximax less than γ, and we derive a contradiction. Thus $\exists k \in [1, m]$ such that $\overrightarrow{\gamma}^*[k] < \overrightarrow{\gamma}[k]$ and $\forall i < k$, we have $\overrightarrow{\gamma}^*[i] = \overrightarrow{\gamma}[i]$. *We assume inductively* that the solution corresponding to load vector γ^* satisfies all constraints of the problem P_i for every $i \in [1, k-1]$. Consequently, such a solution also satisfies all constraints of the problem P_k, and it could not give a better result for α^*, which proves that $\overrightarrow{\gamma^*}[k] = \overrightarrow{\gamma}[k]$. We have thus shown that $\overrightarrow{\gamma^*} = \overrightarrow{\gamma}$ and consequently the optimality of the vector $\overrightarrow{\gamma}$. Last, since they are both possible solutions obtained at the end of the algorithm, according to Proposition 1, $\gamma = \gamma^*$.
The polynomiality of the approach follows from the fact that Algorithm 1 computes the optimal lexicographically minimum load vector in at most $(2m - 1)$ LP computations and each of them can be solved in polynomial time. ♦

3 Lexicographically minimum load networks

In this section, we apply the results of Section 2 to the problem of leximax minimization of load among network links. Thus, 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 we call a *lexicographically minimum load network*. We consider the multi-commodity network and we focus on the splittable case, (the unsplittable one is NP-complete; see for instance [3]). Some relevant work on this problem is presented in [2] where the authors propose a strongly polynomial approach. However, they consider allocating bandwidth between two endpoints of a backbone network so that the network is equitably loaded, and therefore this work is limited to the case of a single-commodity network. In the following, we generalize this result to multi-commodity networks and consider linear objetive functions as well as the non-linear Kleinrock function. To start with, let us give some definitions and notation useful for the remainder of the paper.

3.1 Definitions and notation

A capacitated network is a given triplet $N = (V, E, C)$, where V is a nonempty finite set of nodes. E is the set of arcs, and C_e is the capacity associated with arc e. Let D be the set of commodities with labels in $\{1, 2, ..., |D|\}$. Each commodity d has a source node s^d and a sink node t^d, and an associated value ($T^d \geq 0$). A flow f^d in network N with respect to commodity d is a function from E to $\mathbb{R}^+$ that respects the mass-balance (or Kirchoff) and capacity constraints. Then, the multi-commodity flow is said to be *feasible* when it satisfies all the

traffic constraints, (*i.e.*, for each demand, the value of the flow from s^d to t^d is not less than T^d). Mathematically the multi-commodity flow problem can be formulated as follows:

$$\sum_{j:(i,j)\in E} f_{ij}^d - \sum_{j:(j,i)\in E} f_{ji}^d = \begin{cases} T_d, i = s^d \\ -T_d, i = t^d \\ 0, otherwise \end{cases} \quad \forall i \in V, \forall d \in D \tag{7}$$

$$\sum_{d\in D} f_{ij}^d \leq C_{ij} \qquad \forall (i,j) \in E \tag{8}$$

$$f_{ij}^d \geq 0 \quad \forall (i,j) \in E, \forall d \in D. \tag{9}$$

Finally, a load function $L(f, C)$ (linear or not) 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 $\frac{f_e}{C_e}$. A special case of non-linear load functions widely used in the telecommunication domain is the Kleinrock function (see [4]). It is expressed as $\frac{f_e}{(C_e - f_e)}$, and it takes value in $[0, +\infty)$. For more information on multicommodity flows, see Ahuja et al [1].

Definition 2 *Given a network (V,E,C), a multi-commodity flow f and a load function $L(f, C)$, the vector $\gamma = \{\gamma[e], e \in E\}$, whose coordinates give the load value associated with each arc (i.e., $\gamma[e] = L(f_e, C_e)$), 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 phrase *lexicographically minimum load network problem* to refer to the problem of computing a multi-commodity flow such that its load vector is leximax minimal in the set of feasible load vectors with respect to F. Similarly, a leximax minimal load vector is also referred as a *lexicographically minimum load vector*. Now, the lexicographically minimum load network problem can be restated simply: *given a capacitated network $N(V, E, C)$ and a set of commodities D with values T^d, find the leximax minimal load vector γ and a respective multi-commodity flow.*

3.2 Kleinrock function

In this paragraph we will briefly discuss the problem for a special case of non-linear arc load functions. Let us consider the well-known Kleinrock function defined as $\frac{f_e}{(C_e - f_e)}$. All arcs having $f_e = C_e$ will be considered as equally maximally loaded. One can easily see that the corresponding mathematical formulation is not linear any more. We can get around this difficulty by using the following result, which is elementary and stated without proof.

Proposition 2 *Any multi-commodity flow whose load vector is leximax minimal for the linear load function $\frac{f_e}{C_e}$ is also leximax minimal for the Kleinrock load function, and vice-versa.*

4 Lexicographically maximum load linear programming problem

In an analogous way we can formulate the lexicographically maximum load linear programming problem. We next show how to apply this result to a generalization of the maximum concurrent flow problem and discuss the minimum balance problem as a special case of the lexicographically maximum load problem.

Consider the following linear programming feasibility problem

$$\begin{aligned} \sum_{j=1}^{m} a_{ij}x_j \geq b_i \quad \forall i \in 1..m \\ Dx = d \\ x \geq 0, \end{aligned}$$

where we require that all b_i are strictly positive and that there is a solution to $\{Dx = d; x \geq 0\}$. As before, we say that γ is a *load vector*, which can take on any real value or possibly positive infinity, if there is a feasible solution to

$$\begin{aligned} \sum_{j=1}^{m} a_{ij}x_j \geq \gamma_i b_i \quad \forall i \in 1..m \\ Dx = d \\ x \geq 0. \end{aligned}$$

In this case, we would want to find $\overleftarrow{\gamma}$ so that is lexicographically maximal. Note that $\overleftarrow{\gamma}$ is the vector obtained from γ by ordering indices in non-decreasing order. Similarly to Section 1 we define:

Definition 3 *A vector $\gamma \in X \subset R^n$, is said to be leximin maximal if for every vector $\gamma' \in X$, $\overleftarrow{\gamma}$ is lexicographically greater than or equal to $\overleftarrow{\gamma'}$.*

We next modify Algorithm 1 so that it computes leximin maximal load vectors.

Algorithm 2. Computing lexicographically maximum load vector.
Step 1.
Find a maximum scalar value α such that $\gamma_i = \alpha$ for all i is a feasible load vector. We do this by solving the linear program (P_1'):

$$\begin{aligned} &Maximize \quad \alpha \\ &\sum_{j=1}^{m} a_{ij}x_j \geq \alpha b_i \quad \forall i \in 1..m \qquad (10) \\ &Dx = d \qquad (11) \\ &x \geq 0. \qquad (12) \end{aligned}$$

Let α^* be the optimal value. If $\alpha^* = +\infty$, then every variable gets the same load $(+\infty)$ and the algorithms stops here; otherwise, go to Step 2.
Step 2.
2.a.) Determine a feasible vector x that has the fewest number of binding constraints among the constraints (10).
2.b.) We next transfer these tight constraints to (11), and iterate. The correctness of the Algorithm 2 can be shown in a similar way to Theorem 1.

Application to lexicographically maximum concurrent flow problem. We next formulate the lexicographically maximum concurrent flow problem, which generalizes the concurrent flow problem ([11]). See also [5] where the author has presented a method for the fair single-source splittable flow problem, and some recent work in [6] and [7]. The concurrent flow problem is defined as a multicommodity flow problem in a capacitated network $N = (V, E, C)$ and a set of commodities D. For each commodity d, the goal is to send T_d units from the source node s^d to its destination node t^d. If there is no feasible solution, then the objective is to maximize the satisfaction ratio for each commodity, that is, find the largest α such that for each commodity we can send simultaneously αT_d units. In the lexicographically maximum concurrent flow problem, we look for a load vector that is lexicographically maximal. This models the situation in which one wants to maximize the worst satisfaction ratio; then one wants to maximize the second worst satisfaction ratio, and so on.

5 The minimum balance problem

In this section we focus on the minimum balance problem and show that the minimum balance problem is a special case of the lexicographically maximum load problem.
Let G = (V, E) be a strongly connected directed graph with cost c_{ij} associated with each arc $(i, j) \in E$. For any vector node potential π, let c^π be the vector of reduced costs, that is, $c^\pi_{ij} = c_{ij} + \pi_i - \pi_j$ for all $(i, j) \in E$.

A subset of the nodes of G is called *minimum-balanced* with respect to the cost vector c if the minimum cost among arcs entering the subset is the same as is the minimum cost among arcs leaving the subset. The *minimum-balance problem* for a strongly connected directed graph G is to determine a vector π of node potentials such that every strict subset $V' \subset V$ of nodes, V' is minimum balanced with respect to c^π. Schneider and Schneider ([10]) developed this concept in the context of matrix balancing, and showed how to solve it in $O(n^2m)$ time as a sequence of $O(n)$ minimum cycle mean problems. Young, Tarjan, and Orlin ([9]) improved the running time to $O(nm + n^2 log n)$.

We next show that the minimum balance problem is a special case of the lexicographically maximum load linear programming problem. To see this, we say that a vector γ is a feasible load vector if it satisfies the following system:

$$c_{ij} + \pi_i - \pi_j \geq \gamma_{ij}, \ \forall (i, j) \in E \tag{13a}$$

$$\pi, \gamma \ \ \textit{unrestricted in sign.} \tag{13b}$$

The *mean* of a cycle C is $\mu(C) = \sum_{(i,j)\in C} c_{ij}/|C|$. The *minimum cycle mean* of G is $\lambda^* = min\{\mu(C) : \text{C is a cycle of } G\}$.

Theorem 2 *Let $G = (V, E)$ be a strongly connected directed graph. Suppose that (π^*, γ) is a feasible solution for the system (13), and that $\overleftarrow{\gamma}$ is a lexicographically maximal load vector for (13). Then π^* solves the minimum balance problem on E.*

Proof. We first show that there is a solution to the lexicographically maximum load problem, and the solution is finite valued. First, for any arc $(i, j), \gamma_{ij} \geq \lambda^*$, where λ^* is the minimum cycle mean. This statement holds since $\lambda^* = \overleftarrow{\gamma}[1]$.

Next, for any arc $a \in E$, there is a cycle C containing arc a, and

$$c_a^\pi + \sum_{e \in C \setminus a} c_e^\pi = \sum_{e \in C} c_e \leq |C| c_{max},$$

where c_{max} is the maximum cost of an arc in G. Therefore,

$$\gamma_a \leq c_a^\pi \leq |C| c_{max} - \sum_{e \in C \setminus a} c_e^\pi \leq |C| c_{max} - \sum_{e \in C \setminus a} \gamma_e \leq |C| c_{max} - (|C|-1)\lambda^*.$$

We now prove by contradiction that the optimum solution π^* solves the minimum balance problem on E. Let $d_{ij} = c_{ij} + \pi_i^* - \pi_j^*$. Suppose that there is a subset $V' \subset V$

$$\alpha = min\{d_{ij} : i \in V', j \in V \setminus V'\} \neq \beta = min\{d_{ij} : i \in V \setminus V', j \in V'\}.$$

We consider the case that $\alpha < \beta$, and the case that $\alpha > \beta$ is proved the same way.
Let $\epsilon = (\beta - \alpha)/2$. Let

$$\pi'_j = \begin{cases} \pi_j^* + \epsilon, & \forall j \in V' \\ \pi_j^*, & \forall j \in V \setminus V' \end{cases}$$

When π^* is replaced by π', the loads for arcs with both endpoints contained in V' or both endpoints contained in $V \setminus V'$ remain the same. The loads for arcs from V' to $V \setminus V'$ are increased, and the loads for arcs directed from $V \setminus V'$ into V' are reduced. Therefore, π', yields a lexicographically greater load vector $\overleftarrow{\gamma'}$ than does π^*. ♦

One can implement the algorithm given in Section 4 as applied to the minimum balance problem so that solves the minimum balance problem as a sequence of $O(n)$ minimum cycle mean problems. The first minimum cycle mean problem is on the original graph. At the k^{th} iteration, the minimum cycle mean problem is obtained by contracting all arcs whose load vector is determined in a previous iteration. This is equivalent to the algorithm given by Schneider and Schneider ([10]).

6 Concluding remarks

In this paper we have modeled and proposed algorithms for the lexicographically minimum/maximum load linear programming problems and we have proved the correctness of the proposed approaches. We have considered two network applications, one for the lexicographically minimum load networks and the other for the lexicographically maximum concurrent flow problem. We show that the minimum balance problem is a special case of the lexicographically maximum load problems. We note that this is the first paper that observes the connection of the minimum balance problem to lexicographically maximum load problems.

References

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

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

[3] 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.

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

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

[6] 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.

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

[8] 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.

[9] N.E. Young, R.E. Tarjan, and J.B. Orlin. *Faster Parametric Shortest Path and Minimum Balance Algorithms. Networks. 21*, 1991, 205-221.

[10] H. Schneider and M. Schneider, *Max-balancing weighted directed graphs and Matrix Scaling. Mathematics of Operations Research 16*, 1991, 208 -222.

[11] F. Shahrokhi and D.W. Matula *The Maximum Concurrent Flow Problem. in Journal of the ACM*, Vol. 37, pp. 318-334, 1990.