Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems

Naveen Garg [*]
Computer Science and Engineering
Indian Institute of Technology, New Delhi

Jochen Könemann [†]
Dept. of Combinatorics and Optimization
University of Waterloo, Ontario N2L 3G1

Abstract

This paper considers the problem of designing fast, approximate, combinatorial algorithms for multicommodity flows and other fractional packing problems. We present new faster and much simpler algorithms for these problems.

1 Introduction

Consider the problem of computing a maximum *s*-*t* flow in a graph with unit edge capacities. While there are many different algorithms known for this problem we discuss one which views the problem purely as one of packing *s*-*t* paths so that constraints imposed by edge-capacities are not violated. The algorithm associates a length with each edge and at any step it routes a unit flow along the shortest *s*-*t* path. It then multiplies the length of every edge on this path by $1+\varepsilon$ for a fixed ε. Thus the longer an edge is the more is the flow through it. Since we always choose the shortest *s*-*t* path to route flow along, we essentially try to balance the flow on all edges in the graph. One can argue that, if, after sufficiently many steps, M is the maximum flow through an edge, then the flow computed is almost M times the maximum *s*-*t* flow. Therefore scaling the flow by M gives a feasible flow which is almost maximum.

Note that the length of an edge at any step is exponential in the total flow going through the edge. Such a length function was first proposed by Shahrokhi and Matula [22] who used it to compute the throughput of a given multicommodity flow instance. While this problem (and all other problems considered in this paper) can be formulated as a linear program and solved to optimality using fast matrix multiplication [24], the authors of [22] were mainly interested in providing fast, possibly approximate, combinatorial algorithms. Their procedure, which applied only to the case of uniform edge capacities, computed a $(1+\omega)$-approximation to the maximum throughput in time polynomial in ω^{-1}. The key idea of their procedure, which was adopted in numerous subsequent papers, was to compute an initial flow by disregarding edge capacities and then to reroute this, iteratively, along short paths so as to reduce the maximum congestion on any edge.

The running time of [22] was improved significantly by Klein *et.al.* [17]. It was then extended and refined to the case of arbitrary edge capacities by Leighton *et.al.* [18], Goldberg [11] and Radzik [21] to obtain better running times; see Table 1 for the current best bound.

[*]Supported by the EU ESPRIT LTR Project N. 20244 (ALCOM-IT). Work done while the author was at the Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.

[†]Work done while the author was at the Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany.

Plotkin, Shmoys and Tardos [20] observed that a similar technique could be applied to solve any fractional packing problem. Their approach, for packing problems, starts with an infeasible solution. The amount by which a packing constraint is violated is captured by a variable which is exponential in the extent of this violation. At any step the packing is modified by a *fixed amount* in a direction determined by these variables. Hence, the running time of the procedure depends upon the maximum extent to which any constraint could be violated; this is referred to as the *width* of the problem [20]. The running time of their algorithm for packing problems being only pseudo-polynomial, [20] suggest different ways of reducing the width of the problem.

Grigoriadis and Khachiyan [13] consider block angular packing problems which are problems of the form

$$\min\left\{\lambda \,|\, \sum_{i=1}^{k} f^i(x^i) \le \lambda e, x^i \in B^i, 1 \le i \le k\right\}$$

where B^i is a convex set, $f^i : B^i \to \mathbb{R}^m$ is a non-negative convex function and e is the vector of all 1's. They assume the existence of an oracle which given $i, 1 \le i \le k$, non-negative vector y and scalar μ computes $\min\left\{y^T f^i(x) \,|\, f^i(x) \le \mu e, x \in B^i\right\}$ and show how to find a $(1+\varepsilon)$-approximation to the block angular packing problem with only $k^2 \ln m(\varepsilon^{-2} + \ln k)$ calls to this oracle. In [14], Grigoriadis and Khachiyan, show that this problem can also be solved in $km(\varepsilon^{-2}\ln\varepsilon^{-1} + \ln m)$ calls to an oracle which computes $\min\left\{y^T f^i(x) | x \in B^i\right\}$. Note that both these running times are independent of the width of the problem.

All the problems that we consider in this paper can be formulated as block angular packing problems. This is immediate for the maximum concurrent flow and the min-cost multicommodity flow problems. For these problems, the blocks are single commodity flows. In [13], the oracle corresponds to finding a min-cost single-commodity flow, while in [14], the oracle is a shortest path computation.

The maximum multicommodity flow problem can also be formulated as a block-angular packing problem with one block, B, which is the set of all multicommodity flows of total value 1. For a flow $x \in B$, $f(x)$ is a vector denoting the fraction of the capacity utilized by x on the edges. If for a flow x, $f(x) \le \lambda e$, then the flow x/λ satisfies all capacities and routes $1/\lambda$ units. Thus computing maximum flow is the same as minimizing λ. A similar idea can also be used to formulate fractional packing as a block-angular convex program.

In a significant departure from this line of research and motivated by ideas from randomized rounding, Young [25] proposed an *oblivious rounding* approach to packing problems. Young's approach has the essential ingredient of previous approaches — a length function which measures, and is exponential in, the extent to which each constraint is violated by a given solution. However, [25] builds the solution from scratch and at each step adds to the packing a variable which violates only such packing constraints that are not already too violated. In particular, for multicommodity flow, it implies a procedure which does not involve rerouting flow (the flow is only scaled at the end) and which for the case of maximum *s*-*t* flow reduces to the algorithm discussed at the beginning of this section.

Our Contributions. In this paper we provide a unified framework for multicommodity flow and packing problems which yields significantly simpler and faster algorithms than previously known. Our approach is similar to Young's approach for packing problems. However, we develop a new and simple combinatorial analysis which has the added flexibility that it allows us to make the greatest possible advance at each step. Thus for the setting of maximum *s*-*t* flows with integral edge capacities, Young's procedure routes a unit flow at each step while our procedure would route enough flow so as to saturate the minimum capacity edge on the shortest *s*-*t* path. This simple modification is quite powerful and delivers a slightly better running time and much simpler proof.

Our approach yields a new, very natural, algorithm for maximum concurrent flow (section 5) which extends in a straightforward manner to min-cost multicommodity flows (section 6). These algorithms use a min-cost flow computation as a subroutine and have running times that match the best known. We also provide algorithms for these two problems which use shortest path computations as a subroutine and are faster than previous algorithms. One idea in these algorithms which is key to the faster running times is to organize all computation sequentially, and to use the length updates done at one step in the computations done at all subsequent steps. This is, in some ways, similar to the round-robin idea employed by Radzik [21].

This paper first appeared as a technical report in [9] and then as an extended abstract in [10]. Subsequently the approach presented here has been extended and improved results obtained for almost all the problems considered here. For the maximum multicommodity flow problem, Fleischer [7] obtained a running time that is independent of the number of commodities. Karakostas [15], obtained a corresponding result for the maximum multicommodity flow and min-cost multicommodity flow problems. We discuss the ideas behind these improvements in the appropriate sections.

Bienstock and Iyengar [4] recently adapted a method by Nesterov [19] in order to obtain a $(1+\omega)$-approximation for generalized packing problems. The ω-dependence of the running time of their algorithm is $O((1/\omega)\cdot\log 1/\omega)$ as opposed to a dependence of $O(1/\omega^2)$ of our algorithms. However, their algorithm needs to solve a convex quadratic program in each iteration and this is computationally substantially more expensive than the oracle calls necessary in our algorithms. As a result, our algorithms remain faster than the algorithm by Bienstock and Iyengar for a fixed or moderately small ω.

Table 1 summarizes our results. All our algorithms are deterministic and compute a $(1+\omega)$-approximation to the optimum solution. In giving the running times we ignore polylog factors; the $\tilde{O}$ denotes this fact.

Problem	Previous Best	Our running time	Subsequent Improvement
Max. multicomm. flow	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}km^2)$	$\tilde{O}(\omega^{-2}m^2)$ [7]
Fractional Packing	$\tilde{O}(\omega^{-2}mD)$ [14]	$\tilde{O}(\omega^{-2}mD)$	$\tilde{O}(\omega^{-2}(mL+D))$ [26]
Maximum concurrent flow	$\tilde{O}(\omega^{-2}kmn)$ [21]	$\tilde{O}(\omega^{-2}kmn)$	
	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}(k+m)m)$	$\tilde{O}(\omega^{-2}m^2)$ [15]
Max. cost-bounded concurrent flow	$\tilde{O}(\omega^{-2}kmn)$ [12]	$\tilde{O}(\omega^{-2}kmn)$	
	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}(k+m)m)$	$\tilde{O}(\omega^{-2}m^2)$ [15]

Table 1: A summary of our results. Here, D denotes the number of non-zero entries in the given constraint matrix and L is the maximum number of non-zero entries in a column.

The framework introduced in this paper for multicommodity flow problems was extended by Fleischer and Wayne [8] to generalized flow. The book by Bienstock [3] is a good survey of the theoretical issues and computational studies done on this topic.

2 Maximum multicommodity flow

Given a graph $G=(V,E)$ with edge capacities $c: E \rightarrow \mathbb{R}^+$ and k pairs of terminals $(s_1,t_1),\ldots,(s_k,t_k)$, with one commodity associated with each pair, we want to find a multicommodity flow such that the sum of the flows of all commodities is maximized. Let $\mathscr{P}_j$ be the set of s_j,t_j-paths in G for all $1 \le j \le k$, and define $\mathscr{P}$ to be the union of $\mathscr{P}_1,\ldots,\mathscr{P}_k$. Also let $\mathscr{P}_e$ be the set of paths in $\mathscr{P}$ that use edge e for all $e \in E$. The path-flow linear programming formulation for the maximum-multicommodity flow problem has a variable $x(p)$ for the flow sent along each path $p \in \mathscr{P}$:

$$\begin{aligned} \max \quad & \sum_{p\in\mathscr{P}} x(p) && (P_{\text{mmc}}) \\ \text{s.t.} \quad & \sum_{p\in\mathscr{P}_e} x(p) \le c(e) \quad \forall e \in E \\ & x \ge 0. \end{aligned}$$

The dual to this linear program associates a length $l(e)$ with each of the edges $e \in E$.

$$\begin{aligned} \min \quad & D(l) \overset{\text{def}}{=} \sum_{e\in E} c(e)\cdot l(e) && (D_{\text{mmc}}) \\ \text{s.t.} \quad & \sum_{e\in p} l(e) \ge 1 \quad \forall p \in \mathscr{P} \\ & l \ge 0. \end{aligned}$$

Observe that the above two linear programs (P_{mmc}) and (D_{mmc}) have exponential size. Optimal solutions can, however, be found in polynomial time as equivalent polynomial-size edge-flow formulations exist (e.g., see [1]).

In the following let $\texttt{dist}_j(l)$ be the length of the shortest s_j,t_j-path with respect to length l for $1 \le j \le k$. Also let $\alpha(l) \overset{\text{def}}{=} \min_j \texttt{dist}_j(l)$ be the minimum length path between any pair of terminals. Then (D_{mmc}) is equivalent to finding a length function $l : E \to \mathbb{R}^+$ such that $\frac{D(l)}{\alpha(l)}$ is minimized. Let $\beta \overset{\text{def}}{=} \min_l D(l)/\alpha(l)$.

The algorithm proceeds in iterations. Let l_{i-1} be the length function at the beginning of the i^{th} iteration and f_{i-1} be the total flow routed in iterations $1 \ldots i-1$. Let P be a path of length $\alpha(l_{i-1})$ between a pair of terminals and let c be the capacity of the minimum capacity edge on P. In the i^{th} iteration we route c units of flow along P. Thus $f_i = f_{i-1} + c$. The function l_i differs from l_{i-1} only in the lengths of the edges along P; these are modified as $l_i(e) = l_{i-1}(e)(1 + \varepsilon c/c(e))$, where ε is a constant to be chosen later.

Initially every edge e has length δ, *ie.*, $l_0(e) = \delta$ for some constant δ to be chosen later. For brevity we denote $\alpha(l_i), D(l_i)$ by $\alpha(i), D(i)$ respectively. The procedure stops after t iterations where t is the smallest number such that $\alpha(t) \ge 1$.

2.1 Analysis

For every iteration $i \ge 1$

$$\begin{aligned} D(i) &= \sum_e l_i(e)c(e) \\ &= \sum_e l_{i-1}(e)c(e) + \varepsilon \sum_{e\in P} l_{i-1}(e)c \\ &= D(i-1) + \varepsilon(f_i - f_{i-1})\alpha(i-1) \end{aligned}$$

which implies that

$$D(i) = D(0) + \varepsilon \sum_{j=1}^{i} (f_j - f_{j-1})\alpha(j-1) \tag{1}$$

Consider the length function $l_i - l_0$. Note that $D(l_i - l_0) = D(i) - D(0)$ and $\alpha(l_i - l_0) \geq \alpha(i) - \delta L$ where L is the maximum number of edges on any simple path in G. Hence

$$\beta \leq \frac{D(l_i - l_0)}{\alpha(l_i - l_0)} \leq \frac{D(i) - D(0)}{\alpha(i) - \delta L} \tag{2}$$

Substituting this bound on $D(i) - D(0)$ in equation (1) we get

$$\alpha(i) \leq \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^{i} (f_j - f_{j-1}) \alpha(j-1)$$

To solve the above recurrence we first note that the sequence $x(0), x(1), \ldots x(i), \ldots$ where $x(i) = \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^{i} (f_j - f_{j-1}) x(j-1)$ dominates the sequence $\alpha(0), \alpha(1), \ldots, \alpha(i), \ldots$ where $x(0) = \alpha(0)$. Now

$$\begin{aligned} x(i) &= \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^{i-1} (f_j - f_{j-1}) x(j-1) + \frac{\varepsilon}{\beta} (f_j - f_{j-1}) x(i-1) \\ &= x(i-1)(1 + \varepsilon(f_i - f_{i-1})/\beta) \\ &\leq x(i-1) e^{\varepsilon(f_i - f_{i-1})/\beta} \end{aligned}$$

Since $x(0) = \alpha(0) \leq \delta L$ we have $x(i) \leq \delta L e^{\varepsilon f_i/\beta}$ and this implies

$$\alpha(i) \leq \delta L e^{\varepsilon f_i/\beta}$$

By our stopping condition

$$1 \leq \alpha(t) \leq \delta L e^{\varepsilon f_t/\beta} \tag{3}$$

and hence

$$\frac{\beta}{f_t} \leq \frac{\varepsilon}{\ln(\delta L)^{-1}} \tag{4}$$

Claim 2.1. *There is a feasible flow of value* $\frac{f_t}{\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}}$

Proof. Consider an edge e. For every $c(e)$ units of flow routed through e the length of e increases by a factor of at least $1+\varepsilon$. The last time its length was increased, e was on a path of length strictly less than 1. Since every increase in edge-length is by a factor of at most $1+\varepsilon$, $l_t(e) < 1+\varepsilon$. Since $l_0(e) = \delta$ it follows that the total flow through e is at most $c(e) \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$. Scaling the flow, f_t, by $\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$ then gives a feasible flow of claimed value. □

Thus the ratio of the values of the optimum dual and the primal solutions, γ, is $\frac{\beta}{f_t} \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$. By substituting the bound on β/f_t from (4) we obtain

$$\gamma \leq \frac{\varepsilon \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}}{\ln(\delta L)^{-1}} = \frac{\varepsilon}{\ln(1+\varepsilon)} \frac{\ln \frac{1+\varepsilon}{\delta}}{\ln(\delta L)^{-1}}$$

The ratio $\frac{\ln(1+\varepsilon)\delta^{-1}}{\ln(\delta L)^{-1}}$ equals $(1-\varepsilon)^{-1}$ for $\delta = (1+\varepsilon)((1+\varepsilon)L)^{-1/\varepsilon}$. Hence with this choice of δ we have

$$\gamma \leq \frac{\varepsilon}{(1-\varepsilon)\ln(1+\varepsilon)} \leq \frac{\varepsilon}{(1-\varepsilon)(\varepsilon - \varepsilon^2/2)} \leq (1-\varepsilon)^{-2}$$

Since this quantity should be no more than our approximation ratio $(1+w)$ we choose ε appropriately.

2.2 Running time

In the i^{th} iteration we increase the length of the minimum capacity edge along P by a factor of $1+\varepsilon$. Since for any edge e, $l_0(e) = \delta$ and $l_t(e) < 1+\varepsilon$ the number of iterations in which e is the minimum capacity edge on the path chosen in that iteration is at most $\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} L \rceil$. Using the fact that there are m edges we get the following theorem.

Theorem 2.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum multicommodity flow in time $O(\omega^{-2} km \log L \cdot T_{sp})$ where L is the maximum number of edges on a path between any source-sink pair and T_{sp} is the time required to compute the shortest s-t path in a graph with non-negative edge-weights.*

2.3 Subsequent Improvements

Fleischer [7] made the interesting observation that it suffices to route flow along an approximate shortest path and that if the path chosen at each step is an a approximation to the shortest path then the approximation guarantee worsens only by a multiplicative factor a. Her algorithm proceeds in phases each of which is composed of k iterations. If at the start of the i^{th} phase the shortest path between each pair has length at least α then in the j^{th} iteration of this phase we route the j^{th} commodity along any path of length at most $\alpha(1+\varepsilon)$ and move to the next iteration only when the shortest path between s_j, t_j is at least $\alpha(1+\varepsilon)$. This ensures that at the end of the i^{th} phase every (s_j, t_j) pair is at least $\alpha(1+\varepsilon)$ apart. Hence the number of phases is at most $\log_{1+\varepsilon} \delta^{-1}$. The algorithm performs one shortest path computation in each iteration that does not result in flow being routed. Hence the total number of shortest path computations is $(m+k)\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} L \rceil$. Since k can be as large as O(n^2), Fleischer eliminates the dependence of the running time on k by routing all commodities with the same source in an iteration. It is possible to do this without additional effort since Dijkstra's algorithm for computing shortest paths gives the shortest path to every node in the graph.

3 Packing LP

A packing **LP** is a linear program of the kind $\max\left\{c^T x | Ax \le b, x \ge 0\right\}$ where A, b and c are $(m \times n), (m \times 1)$ and $(n \times 1)$ matrices all of whose entries are positive. We also assume that for all i, j, the $(i, j)^{\text{th}}$ entry of A, $A(i, j)$, is at most $b(i)$. The dual of this **LP** is $\min\left\{b^T y | A^T y \ge c, y \ge 0\right\}$.

We view the rows of A as edges and the columns as paths. $b(i)$ is the capacity of edge i and every unit of flow routed along the j^{th} column consumes $A(i, j)$ units of capacity of edge i while providing a benefit of $c(j)$ units.

The dual variable $y(i)$ corresponds to the length of edge i. Define the *length* of a column j with respect to the dual variables y as $\texttt{length}_y(j) \overset{\text{def}}{=} \sum_i A(i, j) y(i) / c(j)$. Finding a shortest path now corresponds to finding a column whose length is minimum; define $\alpha(y) \overset{\text{def}}{=} \min_j \texttt{length}_y(j)$. Also define $D(y) \overset{\text{def}}{=} b^T y$. Then the dual program is equivalent to finding a variable assignment y such that $D(y)/\alpha(y)$ is minimized.

Once again our procedure will be iterative. Let y_{k-1} be the dual variables and f_{k-1} the value of the primal solution at the beginning of the k^{th} iteration. Let q be the minimum length column of A *ie.*, $\alpha(y_{k-1}) = \texttt{length}_{y_{k-1}}(q)$ — this corresponds to the path along which we route flow in this iteration. The minimum capacity edge is the row for which $b(i)/A(i, q)$ is minimum; let this be row p. Thus in this iteration we will increase the primal variable $x(q)$ by an amount $b(p)/A(p, q)$ so that $f_k = f_{k-1} + c(q) b(p)/A(p, q)$. The dual variables are modified as

$$y_k(i) = y_{k-1}(i)\left(1+\varepsilon\frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right)$$

where ε is a constant to be chosen later.

The initial values of the dual variables are given by $y_0(i) = \delta/b(i)$, for some constant δ to be chosen later. For brevity we denote $\alpha(y_k), D(y_k)$ by $\alpha(k), D(k)$ respectively. Thus $D(0) = m\delta$. The procedure stops at the first iteration t such that $D(t) \geq 1$.

3.1 Analysis

The analysis here proceeds almost exactly as in the case of maximum multicommodity flow. For every iteration $k \geq 1$

$$\begin{aligned} D(k) &= \sum_i b(i)y_k(i) \\ &= \sum_i b(i)y_{k-1}(i) + \varepsilon\frac{b(p)}{A(p,q)}\sum_i A(i,q)y_{k-1}(i) \\ &= D(k-1)+\varepsilon(f_k - f_{k-1})\alpha(k-1) \end{aligned} \tag{5}$$

which, as before, implies that

$$D(k) = D(0) + \varepsilon\sum_{l=1}^{k}(f_l - f_{l-1})\alpha(l-1)$$

Let $\beta \overset{\text{def}}{=} \min_y D(y)/\alpha(y)$. Then $\beta \leq D(l-1)/\alpha(l-1)$ and so

$$D(k) \leq m\delta + \frac{\varepsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})D(l-1).$$

In order to solve this recurrence, we first define

$$x(i) = m\delta + \frac{\varepsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})x(l-1)$$

for all $i \geq 0$. We note that the sequence $(x(i))_{i\geq 0}$ dominates the sequence $(D(i))_{i\geq 0}$. Now

$$\begin{aligned} x(k) &= m\delta + \frac{\varepsilon}{\beta}\sum_{l=1}^{k-1}(f_l - f_{l-1})x(l-1) + \frac{\varepsilon}{\beta}(f_k - f_{k-1})x(k-1) \\ &= \left(1+\frac{\varepsilon}{\beta}(f_k - f_{k-1})\right)x(k-1) \\ &\leq e^{\varepsilon(f_k - f_{k-1})/\beta}x(k-1) \\ &\leq e^{\varepsilon f_k/\beta}x(0) = m\delta \cdot e^{\varepsilon f_k/\beta}. \end{aligned}$$

Using $D(k) \leq x(k)$ we therefore obtain

$$D(k) \leq m\delta e^{\varepsilon f_k/\beta}$$

and by our stopping condition

$$1 \leq D(t) \leq m\delta e^{\varepsilon f_t/\beta} \tag{6}$$

and hence

$$\frac{\beta}{f_t} \leq \frac{\varepsilon}{\ln(m\delta)^{-1}}$$

Claim 3.1. *There is a feasible solution to the packing* **LP** *of value* $\frac{f_t}{\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}}$

Proof. The primal solution x we constructed has value f_t. However, it may not be feasible since some packing constraint $(\sum_j A(i,j)x(j))/b(i) \leq 1$ may be violated. When we pick column q and increase $x(q)$ by $b(p)/A(p,q)$ we increase the left-hand-side (LHS) of the i^{th} constraint by $\frac{A(i,q)b(p)}{b(i)A(p,q)}$ $(= z$ say). Simultaneously we increase the dual variable $y(i)$ by a multiplicative factor of $1+\varepsilon z$. By our definition of p it follows that $z \leq 1$ and hence increasing the LHS of the i^{th} constraint by 1 causes an increase in $y(i)$ by a multiplicative factor of at least $1+\varepsilon$. Note that $y_{t-1}(i) < 1/b(i)$ and so $y_t(i) < (1+\varepsilon)/b(i)$. Since $y_0(i) = \delta/b(i)$ it follows that the final value of the LHS of the i^{th} constraint is no more than $\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$. Since this is true for every i, scaling the primal solution by $\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$ gives a feasible solution of value as in the claim. □

The rest of the analysis is exactly the same as in section 2.1 with m replacing L. Thus $\delta = (1+\varepsilon)((1+\varepsilon)m)^{-1/\varepsilon}$.

3.2 Running time

In the k^{th} iteration we increase the dual variable of the "minimum capacity" row by a factor of $(1+\varepsilon)$. Since for any row i, $y_0(i) = \delta/b(i)$ and $y_t(i) < (1+\varepsilon)/b(i)$ and there are m rows in all, the total number of iterations is at most $m\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon} m\rceil$. For explicitly given packing programs one requires O(D) time to compute the minimum length column, where D is the number of non-zero entries in the matrix A. This implies a running time of $mD\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon} m\rceil$ for computing a $(1-\varepsilon)^{-2}$-approximation to the Packing **LP**.

Theorem 3.1. *There is an algorithm that computes a* $(1+\omega)$*-approximation to the Packing* **LP** *in time* $O(\omega^{-2}mD\log m)$ *where* m *is the number of rows and* D *is the number of non-zero entries in the given constraint matrix.*

3.3 Subsequent Improvements

Young [26] observed that with Fleischer's technique this running time improves to $\tilde{O}(\omega^{-2}(mL+D))$ where L is the maximum number of non-zero entries in a column and D is the number of non-zero entries in the constraint matrix.

4 Spreading metrics

Given a graph $G = (V,E)$ with edge costs $c : E \rightarrow \mathbb{R}^+$, a spreading metric is an assignment of lengths to the edges, $l : E \rightarrow \mathbb{R}^+$, so as to minimize $\sum_e l(e)c(e)$ subject to the constraint that for any set $S \subseteq V$ and vertex $r \in S$, $\sum_{v\in S} \texttt{dist}_{r,v}(l) \geq f(S)$ where $\texttt{dist}_{r,v}(l)$ is the distance from r to v under the length function l and $f()$ is a function only of the size of S. For the linear arrangement problem $f(S) = (|S|-1)(|S|-3)/4$ [6] while for the problem of computing a ρ-separator[1] $f(S)$ is defined as $|S| - \rho|V|$ [5].

[1]a minimum cost set of edges whose removal disconnects the graph into connected components each of which has at most $\rho|V|$ vertices.

Since the length function l is positive, the shortest paths from r to the other vertices in S forms a tree — the shortest path tree rooted at r. Thus the above constraints can be equivalently stated as: for any tree T, for any subset S of vertices in T and for any vertex $r \in S$

$$\sum_{v \in S} \texttt{dist}_{r,v}(l,T) \geq f(S)$$

where $\texttt{dist}_{r,v}(l,T)$ denotes the distance from r to v in tree T under the length function l.

Let $u_e(T,S,r)$ be the number of vertices of S in the subtree below edge e when T is rooted at r. Then the above constraint can be rewritten again to obtain the **LP**

$$\begin{array}{ll} \min & \sum_e l(e)c(e) \\ \text{s.t.} & \sum_{e \in T} l(e)u_e(T,S,r) \geq f(S) \quad \forall T, \forall S \subseteq T, \forall r \in S \\ & l \geq 0 \end{array}$$

The dual of this program, which is a packing **LP**, has a non-negative variable $x(T,S,r)$ for every tree T, subset $S \subseteq T$ and vertex $r \in S$ and is as follows

$$\begin{array}{ll} \max & \sum_{T,S,r} x(T,S,r) f(S) \\ \text{s.t.} & \sum_{T: e \in T} x(T,S,r)u_e(T,S,r) \leq c(e) \quad \forall e \in E \\ & x \geq 0 \end{array}$$

Note that the packing **LP** has exponentially many variables. However, the $(1+w)$-approximation to the optimum fractional solution, in the previous section, only needed an oracle that returned the "most violated constraint" of the dual **LP**. In this setting, this oracle is a subroutine, which, given a length function l finds a triple (T,S,r) for which $(\sum_{e \in T} l(e)u_e(T,S,r))/f(S)$, or equivalently $(\sum_{v \in S} \texttt{dist}_{r,v}(l,T))/f(S)$, is minimum.

Our subroutine will try out all n choices for vertex r and for each of these it will determine the best choice of T,S. For a given r and every subset S, the expression $\sum_{v \in S} \texttt{dist}_{r,v}(l,T)$ is minimized when T is the tree of shortest paths from r and under the length function l. Therefore, for a given r, our choice of T will be the shortest path tree rooted at r. Since $f(S)$ depends only on $|S|$, given that $|S| = k$, the ratio $(\sum_{v \in S} \texttt{dist}_{r,v}(l,T))/f(S)$ is minimized when S is the set of k nearest vertices to r. Amongst the n different choices for k, and hence for S, we choose the set for which the above ratio is minimum. Having found the best triple (T,S,r) we now determine the extent to which $x(T,S,r)$ is increased by considering all edges in T and finding the edge for which $c(e)/u_e(T,S,r)$ is minimum.

The subroutine thus requires n single-source shortest path computations. The running time of the procedure is obtained by noting that the subroutine is invoked once in each of the $m\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} m \rceil$ iterations.

Theorem 4.1. *There is an algorithm that computes a $(1+\omega)$-approximation to Spreading metrics in time $O(\omega^{-2} mn \log m \cdot T_{\text{sp}})$ where T_{sp} is the time required to compute single-source shortest paths in a graph with non-negative edge-weights.*

It is easy to improve the running time by a factor n by using Fleischer's idea. After computing the shortest path tree from a certain root vertex and finding the best set S we continue with the same root vertex till the ratio is at least $(1+\varepsilon)$ times the ratio at the start of the phase. Our analysis is now almost exactly the same as for the maximum multicommodity flow problem and leads to a running time of $\tilde{O}(\omega^{-2} m^2)$.

5 Maximum concurrent flow

Once again we are given a graph with edge capacities $c : E \to \mathbb{R}^+$ and k commodities with s_j, t_j being the source and sink, respectively, for commodity j. Now each commodity has a demand $d(j)$ associated with it and we want to find the largest λ such that there is a multicommodity flow which routes $\lambda d(j)$ units of commodity j.

In the following, let $\mathscr{F}_j$ be the set of flows that transport $d(j)$ units of flow from s_j to t_j for all $1 \le j \le k$. Similar to Section 2 we use $\mathscr{F}$ to denote the union of $\mathscr{F}_1, \ldots, \mathscr{F}_k$. For a flow $f \in \mathscr{F}$ and an edge $e \in E$, we let f_e be the amount of flow sent across e. We can then formulate the maximum concurrent flow problem as the following LP:

$$\begin{aligned} \max \quad & \lambda & (P_{\mathrm{mcf}}) \\ \text{s.t.} \quad & \sum_{f \in \mathscr{F}} f_e \cdot x(f) \le c(e) \quad \forall e \in E \\ & \sum_{f \in \mathscr{F}_j} x(f) \ge \lambda \quad \forall 1 \le j \le k \\ & x \ge 0, \lambda \ge 0. \end{aligned}$$

Its dual has a length $l(e)$ for each edge $e \in E$, and a variable $z(j)$ for each commodity $1 \le j \le k$.

$$\begin{aligned} \min \quad & D(l) \overset{\text{def}}{=} \sum_{e \in E} c(e) l(e) & (D_{\mathrm{mcf}}) \\ \text{s.t.} \quad & \sum_{e \in E} f_e \cdot l(e) \ge z(j) \quad \forall 1 \le j \le k, \forall f \in \mathscr{F}_j \\ & \sum_{j=1}^{k} z(j) \ge 1 \\ & l, z \ge 0. \end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the minimum cost of shipping $d(j)$ units of flow from s_j to t_j under cost function l; henceforth denoted by $\texttt{min\_cost}_j(l)$. Let

$$\alpha(l) \overset{\text{def}}{=} \sum_{j=1}^{k} \texttt{min\_cost}_j(l).$$

LP (D_{mcf}) can now be recast as finding an assignment of lengths to the edges, $l : E \to \mathbb{R}^+$, such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum. For now we assume that $\beta \ge 1$ and shall remove this assumption later.

The algorithm now proceeds in phases; each phase is composed of k iterations. Consider the j^{th} iteration of the i^{th} phase and let $l_{i,j-1}$ be the length function before this iteration. In this iteration we route $d(j)$ units of commodity j along the paths given by $\texttt{min\_cost}_j(l_{i,j-1})$. Let $f_{i,j}(e)$ be the flow through edge e. The length function is modified as $l_{i,j}(e) = l_{i,j-1}(e)(1 + \varepsilon f_{i,j}(e)/c(e))$. Then

$$\begin{aligned} D(l_{i,j}) &= \sum_e l_{i,j}(e) c(e) \\ &= D(l_{i,j-1}) + \varepsilon \sum_e l_{i,j-1}(e) f_{i,j}(e) \\ &= D(l_{i,j-1}) + \varepsilon \cdot \texttt{min\_cost}_j(l_{i,j-1}) \end{aligned}$$

The lengths at the start of the $(i+1)^{\text{th}}$ phase are the same as that at the end of the i^{th} phase, *ie.*, $l_{i+1,0} = l_{i,k}$. Initially, for any edge e, $l_{1,0}(e) = \delta/c(e) = l_{0,k}(e)$.

5.1 The Analysis

We shall be interested in the values of the functions $D(), \alpha()$ only for the length functions $l_{i,k}, i \geq 0$. For brevity we denote $D(l_{i,k}), \alpha(l_{i,k})$ by $D(i), \alpha(i)$ respectively. With these new notations we have for $i \geq 1$

$$D(i) = D(l_{i,k}) = D(l_{i,0}) + \varepsilon \sum_{j=1}^{k} \texttt{min\_cost}_j(l_{i,j-1})$$

Since the edge-lengths are monotonically increasing $\texttt{min\_cost}_j(l_{i,j-1}) \leq \texttt{min\_cost}_j(l_{i,k})$ and hence

$$D(i) \leq D(l_{i,0}) + \varepsilon \sum_{j=1}^{k} \texttt{min\_cost}_j(l_{i,k}) = D(i-1) + \varepsilon\alpha(i) \tag{7}$$

Since $\frac{D(i)}{\alpha(i)} \geq \beta$ we have

$$D(i) \leq \frac{D(i-1)}{1-\varepsilon/\beta}$$

Since $D(0) = m\delta$ we have for $i \geq 1$

$$\begin{aligned} D(i) &\leq \frac{m\delta}{(1-\varepsilon/\beta)^i} \\ &= \frac{m\delta}{1-\varepsilon/\beta}(1+\frac{\varepsilon}{\beta-\varepsilon})^{i-1} \\ &\leq \frac{m\delta}{1-\varepsilon/\beta} e^{\frac{\varepsilon(i-1)}{\beta-\varepsilon}} \\ &\leq \frac{m\delta}{1-\varepsilon} e^{\frac{\varepsilon(i-1)}{\beta(1-\varepsilon)}} \end{aligned}$$

where the last inequality uses our assumption that $\beta \geq 1$.

The procedure stops at the first phase t for which $D(t) \geq 1$. Therefore,

$$1 \leq D(t) \leq \frac{m\delta}{1-\varepsilon} e^{\frac{\varepsilon(t-1)}{\beta(1-\varepsilon)}}$$

which implies

$$\frac{\beta}{t-1} \leq \frac{\varepsilon}{(1-\varepsilon)\ln\frac{1-\varepsilon}{m\delta}} \tag{8}$$

In the first $t-1$ phases, for every commodity j, we have routed $(t-1)d(j)$ units. However, this flow may violate capacity constraints.

Claim 5.1. $\lambda > \frac{t-1}{\log_{1+\varepsilon} 1/\delta}$.

Proof. Consider an edge e. For every $c(e)$ units of flow routed through e, we increase its length by at least a factor $1+\varepsilon$. Initially, its length is $\delta/c(e)$ and after $t-1$ phases, since $D(t-1) < 1$, the length of e satisfies $l_{t-1,k}(e) < 1/c(e)$. Therefore the total amount of flow through e in the first $t-1$ phases is strictly less than $\log_{1+\varepsilon}\frac{1/c(e)}{\delta/c(e)} = \log_{1+\varepsilon} 1/\delta$ times its capacity. Scaling the flow by $\log_{1+\varepsilon} 1/\delta$ implies the claim. □

Thus the ratio of the values of the dual and primal solutions, γ, is strictly less than $\frac{\beta}{t-1}\log_{1+\varepsilon} 1/\delta$. Substituting the bound on $\beta/(t-1)$ from (8) we get

$$\gamma < \frac{\varepsilon \log_{1+\varepsilon} 1/\delta}{(1-\varepsilon)\ln\frac{1-\varepsilon}{m\delta}} = \frac{\varepsilon}{(1-\varepsilon)\ln(1+\varepsilon)} \frac{\ln 1/\delta}{\ln\frac{1-\varepsilon}{m\delta}}$$

For $\delta = (m/(1-\varepsilon))^{-1/\varepsilon}$ the ratio $\frac{\ln 1/\delta}{\ln\frac{1-\varepsilon}{m\delta}}$ equals $(1-\varepsilon)^{-1}$ and hence

$$\gamma \le \frac{\varepsilon}{(1-\varepsilon)^2\ln(1+\varepsilon)} \le \frac{\varepsilon}{(1-\varepsilon)^2(\varepsilon-\varepsilon^2/2)} \le (1-\varepsilon)^{-3}$$

Now it remains to choose ε suitably so that $(1-\varepsilon)^{-3}$ is at most our desired approximation ratio $1+w$.

5.2 Running time

By weak-duality we have

$$1 \le \gamma < \frac{\beta}{t-1}\log_{1+\varepsilon}\frac{1}{\delta}$$

and hence the number of phases in the above procedure, t, is strictly less than $1+\beta\log_{1+\varepsilon} 1/\delta$ which implies that $t = \lceil \frac{\beta}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon}\rceil$.

The running time of our computation depends on β which can be reduced/increased by multiplying the demands/capacities appropriately. Let z_i be the maximum possible flow of commodity i and let $z \overset{\text{def}}{=} \min_i z_i/d(i)$. Then z denotes the maximum fraction of the demands that can be routed independently and hence $z/k \le \beta \le z$. We scale the capacities/demands so that $z/k = 1$ thus satisfying our assumption that $\beta \ge 1$. Note however that β could now be as large as k.

If our procedure does not stop within $2\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon}\rceil$ $(= T$, say) phases then we know that $\beta \ge 2$. We double the demands of all commodities and continue the procedure. Note that β is now half its value in the previous phase and is at least 1. We run the procedure for an additional T phases and if it does not halt we again double demands. Since we halve the value of β after every T phases, the total number of phases is at most $T\log k$.

Theorem 5.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2}k\log k\log m\cdot T_{\text{mcf}})$ where T_{mcf} is the time required to compute a minimum cost s-t flow in a graph with non-negative edge-costs.*

The number of phases can be reduced further using an idea from [20]. We first compute a 2-approximation to β using the procedure outlined above. This requires $O(\log k\log m)$ phases and returns $\hat{\beta}$, $\beta \le \hat{\beta} \le 2\beta$. Now create a new instance by multiplying demands by $\hat{\beta}/2$; this instance has $1 \le \beta \le 2$. Therefore we need at most an additional T phases to obtain a $(1+w)$-approximation. Thus the number of phases is $O(\log m(\log k + (\varepsilon\ln 1 + \varepsilon)^{-1}))$ which multiplied by k gives the number of single commodity min-cost flow computations required.

5.3 Avoiding min-cost flow computations

We now show how min-cost flow computations can be avoided in the above algorithm for the maximum-concurrent flow problem. Using the notation introduced in Section 2, an alternate path-flow LP formulation of the maximum concurrent flow problem is as follows:

$$\begin{aligned} \max \quad & \lambda && (P^2_{\mathrm{mcf}}) \\ \text{s.t.} \quad & \sum_{p \in \mathscr{P}_e} x(p) \le c(e) \quad \forall e \in E \\ & \sum_{p \in \mathscr{P}_j} x(p) \ge \lambda \cdot d(j) \quad \forall 1 \le j \le k \\ & x \ge 0, \lambda \ge 0. \end{aligned}$$

Its linear-programming dual has a length $l(e)$ for each edge $e \in E$ and a variable $z(j)$ for each commodity j.

$$\begin{aligned} \min \quad & D(l) \overset{\text{def}}{=} \sum_{e \in E} c(e) l(e) && (D^2_{\mathrm{mcf}}) \\ \text{s.t.} \quad & \sum_{e \in p} l(e) \ge z(j) \quad \forall 1 \le j \le k, \forall p \in \mathscr{P}_j \\ & \sum_{j=1}^{k} d(j) \cdot z(j) \ge 1 \\ & l, z \ge 0. \end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the shortest path between s_j and t_j under length function l. Define

$$\alpha(l) \overset{\text{def}}{=} \sum_j d(j) \mathtt{dist}_j(l)$$

where $\mathtt{dist}_j(l)$ denotes the shortest path distance between s_j and t_j under the length function l. The dual (D^2_{mcf}) can then be viewed as an assignment of lengths to edges, $l : E \to \mathbb{R}^+$, such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum.

The structure of this new algorithm is similar to that in the previous section. Thus the algorithm runs in phases each of which is composed of k iterations. In the jth iteration of the ith phase we route $d(j)$ units of commodity j in a sequence of steps. Let $l^{s-1}_{i,j}$ be the length function before the sth step and let $P^s_{i,j}$ be the shortest path between s_j and t_j, *ie.*, $P^s_{i,j}$ has length $\mathtt{dist}_j(l^{s-1}_{i,j})$. In this step we route $f^s_{i,j} = \min\left\{c, d^{s-1}_{i,j}\right\}$ units of flow along $P^s_{i,j}$ where c is the capacity of the minimum capacity edge on this path. We now set $d^s_{i,j}$ to $d^{s-1}_{i,j} - f^s_{i,j}$; the iteration ends after p steps where $d^p_{i,j} = 0$.

Thus at each step we perform a shortest path computation instead of a min-cost flow computation as in section 6. The length functions are modified in exactly the same manner as before and the analysis is almost exactly the same. Thus after routing all flow of commodity j we have

$$D(l^p_{i,j}) \le D(l^0_{i,j}) + \varepsilon \cdot d(j) \mathtt{dist}_j(l^p_{i,j})$$

and after routing all commodities in the ith phase we have

$$D(l_{i,k}) \le D(l_{i,0}) + \varepsilon \sum_{j=1}^{k} d(j) \mathtt{dist}_j(l_{i,k})$$

Using the same abbreviations as before we again obtain

$$D(i) \le D(i-1) + \varepsilon \alpha(i)$$

Beyond this point we follow the analysis of section 5.1 to argue that we have a $(1+\omega)$-approximation for the same choice of ε and δ.

For the running time we again note that in each step, except the last one in an iteration, we increase the length of at least one edge by a factor $1+\varepsilon$. Since each edge has an initial length of δ and a final length less than $1+\varepsilon$, the number of steps exceeds the number of iterations by at most $m\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$. Thus the total number of steps is at most $(2k\log k+m)\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon}\rceil$ and each of these involves one shortest path computation.

Recall from the previous section that we needed k initial max-flow computations to compute an approximate interval for the optimum throughput. We now describe a technique introduced by Grigoriadis and Khachiyan [14] to compute a slightly larger interval using k shortest path computations.

Define a length $l(e) = 1/c(e)$ for each edge $e \in E$. For $1 \le i \le k$, let P_i be a shortest s_i,t_i-path with respect to this length. Then let f be the flow obtained by sending $d(i)$ units of flow along path P_i for all $1 \le i \le k$ concurrently. Let f^* be an optimum concurrent flow that feasibly routes $\beta \cdot d(i)$ units of flow from s_i to t_i for each commodity i and define $\overline{f}$ as $(1/\beta)f^*$. Flow $\overline{f}$ routes the full demand of $d(i)$ units of flow between s_i and t_i for each $1 \le i \le k$ while sending at most $(1/\beta)\cdot c(e)$ of flow on each edge $e \in E$. The total length of flow $\overline{f}$ under l is

$$\sum_{e\in E}\frac{1}{c(e)}\cdot\overline{f}_e \le \frac{m}{\beta}.$$

It is not hard to see that the total length of flow f is at most that of $\overline{f}$ and hence

$$\frac{1}{c(\overline{e})}\cdot f_{\overline{e}} \le \sum_{e\in E}\frac{f_e}{c(e)} \le \frac{m}{\beta}$$

for all edges $\overline{e} \in E$. Equivalently, the flow $f\cdot(\beta/m)$ is feasible. The maximum congestion of f is given by

$$\lambda = \max_{e\in E}\frac{f_e}{c(e)} \le \frac{m}{\beta}.$$

Thus, the flow f/λ is feasible and has a throughput of at least β/m, i.e., $\beta \in [1/\lambda, m/\lambda]$. Using this interval for β, the total number of phases used in our algorithm becomes $T\log m$.

Theorem 5.2. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2}(k\log m+m)\log m\cdot T_{\mathrm{sp}})$ where T_{sp} is the time required to compute the shortest s-t path in a graph with non-negative edge-weights.*

5.4 Subsequent improvements

Karakostas [15] improved the running time of the above algorithm by removing the dependence on k. This was done in a manner similar to the approach followed for maximum multicommodity flow. Thus in an iteration all commodities with the same source are considered together. The shortest path to all sinks are computed with one call to Dijkstra's algorithm and flow is routed along these paths in ratio of the demands of the various commodities. As before, in each step of the iteration, except the last, the length of at least one edge increases by a factor $1+\varepsilon$. However, the number of iterations in a phase is now at most $\min(k,n)$ and hence the overall running time is $\tilde{O}(\omega^{-2}m^2)$.

6 Minimum cost multicommodity flow

Given an instance of the multicommodity flow problem, as in the previous section, edge costs $b : E \to \mathbb{R}^+$, where $b(e)$ represents the cost incurred in shipping 1 unit of flow along edge e, and a bound B, we consider the problem of maximizing λ subject to the additional constraint that the cost of the flow is no more than B. In the following LP formulation we use the notation introduced in Section 5 and we let $b(f)$ denote the cost of a flow $f \in \mathscr{F}$ under cost-function b.

$$\begin{aligned}
\max \quad & \lambda && (P_{\text{mcmcf}}) \\
\text{s.t.} \quad & \sum_{f \in \mathscr{F}} f_e \cdot x(f) \le c(e) \quad \forall e \in E \\
& \sum_{f \in \mathscr{F}_j} x(f) \ge \lambda \quad \forall 1 \le j \le k \\
& \sum_{f \in \mathscr{F}} b(f) \cdot x(f) \le B \\
& x \ge 0, \lambda \ge 0
\end{aligned}$$

Its dual has a length $l(e)$ for each edge $e \in E$, variables $z(1), \ldots, z(k)$ for the throughput constraints, and a variable ϕ for the cost constraint. We will view ϕ as the *length* of the cost constraint.

$$\begin{aligned}
\min \quad & D(l,\phi) \overset{\text{def}}{=} \sum_{e \in E} c(e) l(e) + B \cdot \phi && (D_{\text{mcmcf}}) \\
\text{s.t.} \quad & \sum_{e \in E} f_e \cdot (l(e) + b(e)\phi) \ge z(j) \quad \forall 1 \le j \le k, \forall f \in \mathscr{F}_j \\
& \sum_{j=1}^{k} z(j) \ge 1 \\
& l, z \ge 0.
\end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the minimum cost of shipping $d(j)$ units of flow from s_j to t_j under cost function $l + b\phi$. Define $\alpha(l,\phi) \overset{\text{def}}{=} \sum_j \texttt{min\_cost}_j(l + \phi b)$. Then (D_{mcmcf}) can be restated as finding a length function (l,ϕ) such that $D(l,\phi)/\alpha(l,\phi)$ is minimum; let β denote this minimum value. As in the case of maximum concurrent flow we begin by assuming that $\beta \ge 1$.

Once again the algorithm proceeds in phases each of which is composed of k iterations. In the j^{th} iteration of the i^{th} phase we begin with length functions $(l_{i,j-1}, \phi_{i,j-1})$ and route $d(j)$ units of commodity j. As before, for all edges e, define $l_{i+1,0}(e) = l_{i,k}(e)$ and $l_{1,0}(e) = l_{0,k}(e) = \delta/c(e)$. Similarly $\phi_{i+1,0} = \phi_{i,k}$ and $\phi_{1,0} = \delta/B$.

The flow in each iteration is routed in a sequence of steps; in each step we only route so much flow that its cost does not exceed the bound B. Let $(l_{i,j}^{s-1}, \phi_{i,j}^{s-1})$ be the length functions at the start of the s^{th} step (see Fig. 1); the lengths at the start of the first step are given by $l_{i,j}^0 = l_{i,j-1}$ and $\phi_{i,j}^0 = \phi_{i,j-1}$. Further, let $d_{i,j}^{s-1}$ be the flow of commodity j that remains to be routed in this iteration. We compute $f_{i,j}^s \overset{\text{def}}{=} \texttt{min\_cost}_j(l_{i,j}^{s-1} + b\phi_{i,j}^{s-1})$ which routes $d(j)$ units of flow of commodity j. Since we need to route only $d_{i,j}^{s-1}$ units of flow we multiply the flow function $f_{i,j}^s$ by $d_{i,j}^{s-1}/d(j)$. If $B_{i,j}^s$ is the cost of flow $f_{i,j}^s$ then the cost of the scaled flow is $B_{i,j}^s d_{i,j}^{s-1}/d(j)$. If this quantity exceeds B then we multiply the original flow function $f_{i,j}^s$ by $B_{i,j}^s/B$. We reuse notation and denote the final scaled flow and its cost by $f_{i,j}^s, B_{i,j}^s$ respectively. Now $f_{i,j}^s$ routes at most $d_{i,j}^{s-1}$ units of flow at cost $B_{i,j}^s \le B$.

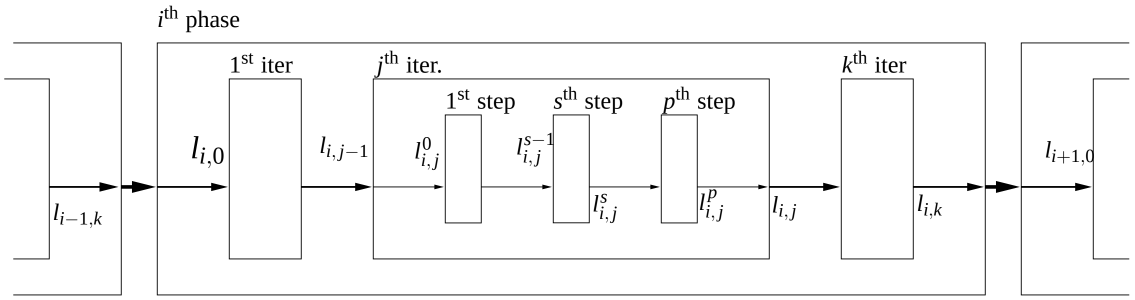

Figure 1: The notation used in section 6. The length functions above the central axis are the lengths *before* the box on the right and the ones below are the lengths *after* the box on the left.

The length functions are modified in a similar manner as before. Thus $l^s_{i,j} = l^{s-1}_{i,j}(1+\varepsilon f^s_{i,j}(e)/c(e))$ and $\phi^s_{i,j} = \phi^{s-1}_{i,j}(1+\varepsilon B^s_{i,j}/B)$. Further, only $d^s_{i,j} = d^{s-1}_{i,j} - f^s_{i,j}$, more units of commodity j remain to be routed in this iteration. The iteration ends at the step p for which $d^p_{i,j} = 0$. The procedure stops at the first step at which $D()$ exceeds 1; let this happen in the t^{th} phase.

6.1 Analysis

Note that now

$$\begin{aligned}
& D(l^s_{i,j}, \phi^s_{i,j}) \\
= \;& D(l^{s-1}_{i,j}, \phi^{s-1}_{i,j}) + \varepsilon \cdot \texttt{min\_cost}_j(l^{s-1}_{i,j} + b\phi^{s-1}_{i,j}) f^s_{i,j}/d(j) \\
\leq \;& D(l^{s-1}_{i,j}, \phi^{s-1}_{i,j}) + \varepsilon \cdot \texttt{min\_cost}_j(l^p_{i,j} + b\phi^p_{i,j}) f^s_{i,j}/d(j)
\end{aligned}$$

where the last inequality holds because the edge-lengths are monotonically increasing over steps. The total flow routed in the p steps equals the demand of commodity j, *ie.*, $\sum_{s=1}^p f^s_{i,j} = d(j)$. Summing over all p steps we get

$$D(l^p_{i,j}, \phi^p_{i,j}) \leq D(l^0_{i,j}, \phi^0_{i,j}) + \varepsilon \cdot \texttt{min\_cost}_j(l^p_{i,j} + b\phi^p_{i,j})$$

The length functions at the start of the $(j+1)^{\text{th}}$ iteration are given by $l_{i,j} = l^p_{i,j}$ and $\phi_{i,j} = \phi^p_{i,j}$. Moving from steps to iterations we have

$$\begin{aligned}
& D(l_{i,j}, \phi_{i,j}) \\
\leq \;& D(l_{i,j-1}, \phi_{i,j-1}) + \varepsilon \cdot \texttt{min\_cost}_j(l_{i,j} + b\phi_{i,j}) \\
\leq \;& D(l_{i,j-1}, \phi_{i,j-1}) + \varepsilon \cdot \texttt{min\_cost}_j(l_{i,k} + b\phi_{i,k})
\end{aligned}$$

where the last inequality uses the fact that the edge-lengths are monotonically increasing over iterations. Summing over all iterations in the i^{th} phase we have

$$\begin{aligned}
D(l_{i,k}, \phi_{i,k}) &\leq D(l_{i,0}, \phi_{i,0}) + \varepsilon \sum_{j=1}^{k} \texttt{min\_cost}_j(l_{i,k} + b\phi_{i,k}) \\
&= D(l_{i-1,k}, \phi_{i-1,k}) + \varepsilon \alpha(l_{i,k}, \phi_{i,k})
\end{aligned}$$

As before we abbreviate $D(l_{i,k}, \phi_{i,k}), \alpha(l_{i,k}, \phi_{i,k})$ to $D(i), \alpha(i)$ respectively to obtain

$$D(i) \leq D(i-1) + \varepsilon\alpha(i)$$

The remainder of the analysis is exactly as in section 5.1. The only modification is in the claim about the throughput of the flow routed. Now we need to argue that the cost of the flow after we scale it by $\log_{1+\varepsilon} 1/\delta$ is at most B, or equivalently, that the cost of the flow routed in the first $t-1$ iterations is at most $B \log_{1+\varepsilon} 1/\delta$. This follows from the fact that $\phi_{t-1,k} < 1/B$ (since $D(t-1) < 1$), that $\phi_{1,0} = \delta/B$ and that in our procedure every time we route flow whose total cost is B we increase ϕ by at least a factor $1+\varepsilon$.

6.2 Running time

Note that except for the last step in each iteration, in all other steps we increase the length function ϕ by a factor $1+\varepsilon$. This implies that the total number of steps exceeds the number of iterations by at most $\log_{1+\varepsilon} 1/\delta$.

Now define z_i as the maximum possible flow of commodity i of cost no more than B. Again $z \overset{\text{def}}{=} \min_i z_i/d(i)$ denotes the maximum fraction of the demands that can be routed if the capacity constraints and the bound B on the cost of the flow applied independently to each commodity. Thus $z/k \leq \beta \leq z$ and we multiply demands suitably so that for the new instance $1 \leq \beta \leq k$. As before we double the demands, thereby halving β, after every $T = 2\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} \frac{m}{1-\varepsilon} \rceil$ phases. Thus the number of iterations is $kT \log k$ and our procedure for minimum cost multicommodity flow needs at most $(2k \log k + 1)\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} \frac{m}{1-\varepsilon} \rceil$ single-commodity min-cost flow computations.

Theorem 6.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum cost-bounded concurrent flow in time $O(\omega^{-2} k \log k \log m \cdot T_{\text{mcf}} + kT_{\text{mcbf}})$ where T_{mcf} is the time required to compute a minimum cost s-t flow in a graph with non-negative edge-costs and T_{mcbf} is the time required to compute the maximum s-t flow of cost at most B in a capacitated network with non-negative edge costs.*

6.3 Avoiding min-cost flow computations

Much like in Section 5.3 we can give an alternate path-flow formulation for the minimum-cost multicommodity flow problem. In the following we let $b(p)$ denote the cost of path $p \in \mathscr{P}$.

$$\begin{aligned}
\max \quad & \lambda && (P^2_{\text{mcmcf}}) \\
\text{s.t.} \quad & \sum_{p \in \mathscr{P}_e} x(p) \leq c(e) \quad \forall e \in E \\
& \sum_{p \in \mathscr{P}_j} x(p) \geq \lambda \cdot d(j) \quad \forall 1 \leq j \leq k \\
& \sum_{p \in \mathscr{P}} b(p) \cdot x(p) \leq B \\
& x \geq 0, \lambda \geq 0.
\end{aligned}$$

Its linear-programming dual has a length $l(e)$ for each edge $e \in E$, a length ϕ for the cost constraint and a variable $z(j)$ for each commodity j.

$$\begin{aligned} \min \quad & D(l,\phi) \overset{\text{def}}{=} \sum_{e \in E} c(e)l(e) + B \cdot \phi && (D^2_{\text{mcmcf}}) \\ \text{s.t.} \quad & \sum_{e \in p} (l(e) + b(e)\phi) \geq z(j) \quad \forall 1 \leq j \leq k, \forall p \in \mathscr{P}_j \\ & \sum_{j=1}^{k} d(j) \cdot z(j) \geq 1 \\ & l, z \geq 0. \end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the shortest path between s_j and t_j under length function $l + b\phi$. We now define $\alpha(l,\phi) \overset{\text{def}}{=} \sum_j d(j)\mathtt{dist}_j(l + b\phi)$. The dual to the min-cost multicommodity flow problem is an assignment of lengths to edges, $l : E \to \mathbb{R}^+$, and a scalar ϕ such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum.

The algorithm differs from the one developed in section 6 in that at any step we route flow along only one path, which, if this is the s^{th} step of the j^{th} phase of the i^{th} iteration, is the shortest path between s_j and t_j under the length function $l^{s-1}_{i,j} + b\phi^{s-1}_{i,j}$. If the minimum capacity edge on this path has capacity c then the flow function at this step, $f^s_{i,j}$, corresponds to routing c units of flow along this path. If $c \leq d^{s-1}_{i,j}$ and the cost of this flow is less than B we route this flow completely. Else we scale it so that the flow routed in this step has cost no more than B and the total flow routed in this iteration does not exceed $d(j)$.

The analysis of the algorithm proceeds as in section 6.1 with the only modification that $\mathtt{min\_cost}_j(.)$ is replaced with $d(j)\mathtt{dist}_j(.)$. For the running time we need only observe that in each step, except the last step in an iteration, we increase, either the length of some edge or the value of ϕ by a factor $1+\varepsilon$. The lengths of the edges and ϕ can each be increased by a factor $1+\varepsilon$ at most $\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$ times. Hence the number of steps exceeds the number of iterations by at most $(m+1)\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \rceil$.

Similar to Section 5.3, we now describe how an idea proposed by Grigoriadis and Khachiyan [14] can be adapted to find a good estimate on the maximum throughput β subject to capacity and cost-bounds. Once again, we define the length l_e of each edge $e \in E$ as $b(e)/B + 1/c(e)$. For each $1 \leq i \leq k$, let P_i be the shortest s_i,t_i-path for this length. Then define f to be the flow obtained by routing $d(i)$ units of flow long P_i for all commodities i simultaneously. As before, let f^* be an optimum cost-bounded flow with throughput β and define $\bar{f}$ as $f^* \cdot (1/\beta)$. The flow $\bar{f}$ routes $d(i)$ units of flow between the terminals of each of the k commodities. The total length of flow $\bar{f}$ under length l is

$$\sum_{e \in E} (b(e)/B + 1/c(e)) \cdot \bar{f}_e \leq \frac{m+1}{\beta}.$$

The total length of flow f is at most that of $\bar{f}$ and hence

$$\frac{1}{c(e)} \cdot f_e \leq \frac{m+1}{\beta}$$

for all edges $e \in E$ and

$$\frac{1}{B} \cdot \sum_{e \in E} b(e) f_e \leq \frac{m+1}{\beta}.$$

The flow $(\beta/(m+1))f$ is therefore a feasible cost-bounded flow with throughput $\beta/(m+1)$.

Define the congestion of f as

$$\lambda = \max\left\{\frac{\sum_{e\in E} b(e) f_e}{B}, \max_{e\in E} \frac{f_e}{c(e)}\right\} \le \frac{m+1}{\beta}.$$

From the above we conclude that the optimal throughput β must be in the interval $[1/\lambda, (m+1)/\lambda]$. Using this interval for β, the total number of phases used in the algorithm becomes $T\log(m+1)$.

Theorem 6.2. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2}(m+k\log m)\log m \cdot T_{\mathrm{sp}})$ where T_{sp} is the time required to compute the shortest s-t path in a graph with non-negative edge-weights.*

6.4 Subsequent Improvements

Karakostas [15] showed how to remove the dependence of the running time on k by grouping commodities with a common source. The shortest paths are now computed with respect to the length function $l+\phi b$ and only so much flow is routed that the cost of flow routed is no more than B. This leads to a $(1+\omega)$-approximation algorithm for computing the maximum cost-bounded concurrent flow in time $\tilde{O}(\omega^{-2}m^2)$.

7 Integrality

A multicommodity flow has integrality q if the flow of every commodity on every edge is a non-negative integer multiple of q. In this section we show how small modifications to the algorithms discussed in previous sections lead to flows that have small integrality.

Our algorithm for maximum multicommodity flow routes flow along a path P in the i^{th} iteration. If c is the minimum capacity of an edge on P then we require that the flow routed in this iteration be no more than c. However, note that if we route $q < c$ units along P and increase the length of an edge e on P by a factor $(1+\varepsilon q/c(e))$ then the algorithm still delivers a $(1-\varepsilon)^{-2}$-approximation to the maximum multicommodity flow, albeit with a worse running time. To obtain a feasible flow we eventually scale the flow constructed in this manner by $\log_{1+\varepsilon} 1/\delta$. Thus if we were routing q units in a certain iteration then only $\frac{q}{\log_{1+\varepsilon} 1/\delta}$ units “appear” in the feasible solution.

Theorem 7.1. *Let e be the minimum capacity edge in G and $q \le c(e)$. Then one can in polynomial time compute a flow f which is a $(1-\varepsilon)^{-2}$-approximation to the maximum multicommodity flow and has integrality $\frac{q\varepsilon}{\log_{1+\varepsilon} L}$.*

Corollary 7.2. *If all edges in G have capacity at least $\frac{1}{\varepsilon}\log_{1+\varepsilon} L$ then there is an integral flow which is a $(1-\varepsilon)^{-2}$-approximation to the maximum multicommodity flow.*

For maximum concurrent flow we use the algorithm from section 5.3. Recall that in the s^{th} step of the j^{th} iteration in the i^{th} phase we route $f^s_{i,j} = \min\left\{c, d^{s-1}_{i,j}\right\}$ units of flow along path $P^s_{i,j}$ where c is the minimum capacity of an edge on this path and $d^{s-1}_{i,j}$ is the residual demand of the j^{th} commodity. As in the case of maximum multicommodity flow we route $q < f^s_{i,j}$ units of flow in this step and increase the length of an edge e on P by a factor $(1+\varepsilon q/c(e))$. To ensure that exactly q units of flow can be routed in each step of the j^{th} iteration we require that $d(j)$ be an integral multiple of q. To obtain a feasible flow we scale the flow constructed by $\log_{1+\varepsilon} 1/\delta$. Hence in the final solution the flow appears in units of $\frac{q\varepsilon}{\log_{1+\varepsilon} m/(1-\varepsilon)}$.

Theorem 7.3. *Let e be the minimum capacity edge in G and $q \leq c(e)$. If all demands are integral multiples of q then one can, in polynomial time, compute a flow f which is a $(1-\varepsilon)^{-3}$-approximation to the maximum concurrent flow and f has integrality $\frac{q\varepsilon}{\log_{1+\varepsilon} m/(1-\varepsilon)}$.*

Corollary 7.4. *If all edges in G have capacity at least $\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon}$ and all demands are integral multiples of $\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon}$ then there is an integral flow which is a $(1-\varepsilon)^{-3}$-approximation to the maximum concurrent flow.*

The above theorem and its corollary also hold for the setting of min-cost multicommodity flows.

8 Improvements in Practice

In this section we propose a heuristic for our algorithms that turns out to improve running times greatly in practice. The idea is best explained with the example of the maximum multicommodity algorithm from Section 2. To route $(f_i - f_{i-1})$ units of flow in iteration i, we computed k shortest paths which was later improved to one shortest path computation by Fleischer.

The idea now is to allow flow to be routed along paths which have length more than the shortest path. More precisely, let l be the vector of current edge lengths and let f be the total flow routed so far. Let $\widehat{\beta}$ be an upper-bound on β. We allow flow to be routed along a path P if its length is at most $L\delta e^{\varepsilon f/\widehat{\beta}}$ where L, ε and δ are defined as in Section 2. The amount of flow routed along P equals the minimum capacity of an edge on P. The edge lengths are updated in the same manner as before. The procedure stops when

$$1 \leq L\delta e^{\varepsilon f/\widehat{\beta}}.$$

We first show that in the modified algorithm we can always find a path whose length is at most the given bound. Observe that the $\alpha(j-1)$ on the right side of equation 1 really denotes the length of the path along which flow was routed in the j^{th} iteration. As our induction hypothesis we assume that this quantity is at most $\delta L e^{\varepsilon f_{j-1}/\widehat{\beta}}$, which in turn is at most $\delta L e^{\varepsilon f_{j-1}/\beta}$; we denote this last expression by $y(j-1)$. This implies that the length of the shortest path at the i^{th} iteration, $\alpha(i)$, is bounded as

$$\alpha(i) \leq \delta L + \frac{\varepsilon}{\beta}\sum_{j=1}^{i}(f_j - f_{j-1})y(j-1)$$

Recall the solution of the recurrence for the sequence x in Section 2.1. It follows that the expression on the right is at most $\delta L e^{\varepsilon f_i/\beta}$ which shows that the shortest path between any pair has length less than the specified bound.

In the original algorithm in Section 5 we used the stopping condition in two ways. We argued that the length of any edge is no more than $1+\varepsilon$ and that $\delta L e^{\varepsilon f_t/\beta} \geq 1$. The termination condition of the modified algorithm is the same as the second property. The first property also holds since all paths along which flow was ever routed had length at most 1.

This modification to the algorithm allows one to continue sending flow on a path till its length exceeds the specified bound. Thus we can now route more flow for every shortest path computation performed. This same heuristic can be adapted to the other problems considered in this paper to obtain better running times in practice.

Acknowledgments. The authors thank Philip Klein, Cliff Stein and Neal Young for useful discussions.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[2] B. Awerbuch and F. T. Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. In *Proceedings, ACM Symposium on Theory of Computing*, pages 487–496, 1994.

[3] D. Bienstock. *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*. Kluwer Academic Publishers, 2002.

[4] D. Bienstock and G. Iyengar. Approximating fractional packings and coverings in O(1/epsilon) iterations. *SIAM Journal on Computing*, 35(4):825–854, 2006.

[5] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28, 1999.

[6] G. Even, S. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47, 2000.

[7] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

[8] L.K. Fleischer and K.D. Wayne. Fast and simple approximation schemes for generalized flow. *Mathematical Programming*, 91(2):215–238, 2002.

[9] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. Technical Report 97-1-025, Max-Planck Institut für Informatik, 1997.

[10] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[11] A. V. Goldberg. A natural randomization strategy for multicommodity flow and related algorithms. *Inform. Process. Lett.*, 42:249–256, 1992.

[12] M. Grigoriadis and L. G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{O}(\varepsilon^{-2}knm)$ time. *Math. Programming*, 75:477–482, 1996.

[13] M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM J. Optimization*, 4(1):86–107, 1994.

[14] M. D. Grigoriadis and L. G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Math. Oper. Res.*, 21(2):321–340, 1996.

[15] G. Karakostas. Faster approximation schemes for fractional multicommodity flow problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 166–173, 2002.

[16] D. Karger and S. Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *Proceedings, ACM Symposium on Theory of Computing*, pages 18–25, 1995.

[17] P. Klein, S. Plotkin, C. Stein, and E. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[18] T. Leighton, F. Makedon, S. Plotkin, C. Stein, S. Tragoudas, and E. Tardos. Fast approximation algorithms for multicommodity flow problems. *J. Comput. System Sci.*, 50:228–243, 1995.

[19] Yu. Nesterov. Smooth minimization of non-smooth functions. *Math. Programming*, 103(1):127–152, 2005.

[20] S. Plotkin, D. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20:257–301, 1995.

[21] T. Radzik. Fast deterministic approximation for the multicommodity flow problem. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 486–492, 1995.

[22] F. Shahrokhi and D. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.

[23] C. Stein. *Approximation algorithms for multicommodity flow and scheduling problems*. PhD thesis, MIT, 1992.

[24] P. M. Vaidya. Speeding up linear programming using fast matrix multiplication. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 332–337, 1989.

[25] N. Young. Randomized rounding without solving the linear program. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, 170-178, 1995.

[26] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.