Faster approximation algorithms for packing and covering problems[*]

D. Bienstock[†] G. Iyengar[‡]

August 13, 2004

Abstract

We adapt a method proposed by Nesterov [19] to design an algorithm that computes ϵ-optimal solutions to fractional packing problems by solving $O^*(\epsilon^{-1}\sqrt{Kn})$ separable convex quadratic programs, where K is the maximum number of non-zeros per row and n is the number of variables. We also show that the quadratic program can be approximated to any degree of accuracy by an appropriately defined piecewise-linear program. For the special case of the maximum concurrent flow problem on a graph $G = (V, E)$ with rational capacities and demands we obtain an algorithm that computes an ϵ-optimal flow by solving shortest path problems – the number of shortest paths computed grows as $O(\epsilon^{-1}\log(\frac{1}{\epsilon}))$ in ϵ, and polynomially in the size of the problem. In contrast, previous algorithms required $\Omega(\epsilon^{-2})$ iterations. We also describe extensions to other problems, such as the maximum multicommodity flow problem and covering problems.

1 Packing problems

The prototypical example of the problems considered in this paper is the *maximum concurrent flow* problem defined as follows. Given a graph $G = (V, E)$, where each edge $e \in E$ has a positive capacity u_e, and K commodities with associated demands $d_k \in \mathbf{R}^{|V|}$, $k = 1, \ldots, K$; solve

$$\begin{array}{ll} \min & \max_{e\in E}\left\{\frac{\sum_{k=1}^{k} f_{k,e}}{u_e}\right\} \\ \text{s. t.} & Nf_k = d_k, \quad k = 1, \ldots, K, \\ & f_k \geq 0, \quad k = 1, \ldots, K. \end{array}$$

where $N \in \mathbf{R}^{|V|\times|E|}$ is the node-arc incidence of the graph. Here and below we will assume that commodities are grouped by source, i.e., for each k there is one vertex $s(k)$ with $d_{k,s(k)} > 0$. We will call a nonnegative vector $f_k \in \mathbf{R}^{|E|}$ a *flow* vector for commodity k if $Nf_k = d_k$, i.e. it routes the corresponding demand d_k. A nonnegative vector $f = (f_1, f_2, \ldots, f_K) \in \mathbf{R}^{K|E|}$ is called a *flow* if f_k is a flow vector for commodity k, $k = 1, \ldots, K$. For a given flow f, the quantity $(\sum_k f_{k,e})/u_e$ is called the *load* or *congestion* on the edge $e \in E$. Thus, the maximum concurrent flow problem computes a flow that minimizes the maximum congestion. Problems of this nature arise in the

[*]Preliminary version appeared in the proceeding of STOC 04.

[†]IEOR Dept. Columbia University. Research partially supported by NSF awards CCR-ITR-0213844 and DMI-0200221

[‡]IEOR Dept. Columbia University. Research partially supported by NSF grants CCR-00-09972, DMS-01-04282 and ONR grant N000140310514.

context of routing in capacitated networks. Furthermore, many network design algorithms solve maximum concurrent flow problems as subroutines in order to find a feasible routing.

The maximum concurrent flow problem is a special case of the *fractional packing* problem. Let $A \in [a_1, \ldots, a_m^T]^T \in \{0,1\}^{m \times n}$, and let $Q \subseteq \mathbf{R}^n_+$ be a polyhedron. Then the fractional packing problem $\textsc{Pack}(A, Q)$ is given by

$$\begin{array}{rcl} \lambda^*_{A,Q} & \doteq & \min \quad \lambda(x) \doteq \max_{1 \le i \le m} \{a_i^T x\}, \\ & & x \in Q. \end{array} \tag{1}$$

The fractional packing problem is itself a special case of the *generalized packing* problem, where each entry a_{ij} of the matrix A is assumed to be nonnegative, rather than $a_{ij} \in \{0, 1\}$.

A special case of the generalized packing problem of particular interest is the so-called *block-angular* problem. In such problems, there exists a positive integers $k > 0$, and n_i, $i = 1, \ldots, k$, with $\sum_i n_i = n$, such that the set $Q = Q^1 \times Q^2 \times \cdots Q^k$, $Q^i \subseteq \mathbf{R}^{n_i}$, $i = 1, \ldots, k$. When Q is a polyhedron this simply says that the nonzeros in the constraint matrix defining Q are arranged, without loss of generality, into a block-diagonal structure with k blocks. The maximum concurrent flow problem is clearly a block-angular problem with each block corresponding to a commodity; the constraints in a given block describe flow conservation and nonnegativity of the flow variables for the corresponding commodity.

Generalized packing problems are linear programs, and consequently can be solved to optimality in polynomial time. Nevertheless, these problems and in particular the maximum concurrent flow problem have long been recognized as extremely challenging. State-of-the-art implementations of the simplex method, and also of interior point methods, running on fast machines, can require an inordinate amount of time to solve maximum concurrent flow problems on networks with a few thousand nodes, even when physical memory is adequate. Frequently, these codes can also consume an extremely large amount of memory. Consequently, they often prove unusable. For more background see [3].

It is not clear, from a theoretical standpoint, why traditional linear programming methods tend to require a large number of iterations to solve maximum concurrent flow problems. However, it is fairly clear why such methods tend to require a large running *time*. This is due to very expensive matrix computations, e.g. matrix inversions or Cholesky factorizations, that must be repeatedly carried out. As an example, consider a maximum concurrent flow problem on a network with $|V|$ vertices, $|E|$ edges and with K commodities. The overall constraint matrix for the linear program will have $|E| + K|V|$ rows and $K|E|$ variables. Even when the graph is very sparse ($|E| = \theta(|V|)$), if we have $K = \theta(|V|)$ the numerical linear algebra will be over matrices of dimension $\theta(|V|^2)$. Further, even though the constraint matrix may be very sparse, the matrix computations will usually experience “fill-in”.

These facts were recognized early on, in the context of block-angular linear programs, and were a motivating factor for some of the earliest decomposition ideas in mathematical programming, such as Dantzig-Wolfe decomposition and Lagrangian relaxation. See [14, 3] for further background. The goal of these methods was to try to reduce the solution of the linear program $\min\{c^T x : Ax \le b,\ x \in Q\}$ to a set of linear programs over Q. For a block-angular problem, a linear program over Q reduces to a set of linear programs over the blocks; therefore, when there are many blocks and each block is small relative to the overall Q, each iteration becomes cheap while direct solution of the complete problem using a standard method may be impossible. For the concurrent flow problem a linear program over Q reduces to a set of shortest path problems for each commodity. In essence, this approach leads to a model of computation where optimization over Q is viewed as the building block of more complex algorithms; the critical theoretical issue then becomes that of minimizing the number of iterations needed to achieve convergence.

This model of computation is perhaps not as precise as one would like it to be – we need to clearly state what can be done in addition to optimization over Q. Nevertheless, the model does capture the notion that we want to avoid matrix algebra over large matrices. In this paper we will adopt this model – an algorithm that fits the model will be called a *decomposition method*, and achieving low iteration counts will be a central goal in the design of our algorithms.

1.1 Modern results

The difficulty of the maximum concurrent flow problem, and its practical relevance, have long provided a strong motivation for developing fast algorithms that can provide provably good, if not optimal, solutions.

Shahrokhi and Matula [23] developed the first approximation algorithm for the maximum concurrent flow problem. They considered the special case where the capacity on all edges are equal. The method in [23] considers an exponential potential function of the form

$$\sum_{e \in E} e^{\alpha\left(\sum_k f_{k,e}\right)}.$$

We will call a flow f with load at most $(1+\epsilon)$ times the optimal load an ϵ-optimal flow. It is shown in [23] that, given $\epsilon \in (0,1)$, one can choose $\alpha = \alpha(\epsilon)$ such that ϵ-optimal flow can be computed by minimizing the potential function to within an appropriate *absolute* error. The algorithm employed in [23] is, roughly, a first-order procedure to minimize the potential function, i.e. a procedure that approximates the potential function by its gradient, and the number of iterations required is $O(\epsilon^{-3})$ times a polynomial in the number of nodes and edges. In addition, the algorithm maintains a list of $O(\epsilon^{-2})$ paths, so that the dependence of the running time on ϵ is $O(\epsilon^{-5})$. Each iteration consists of a shortest-path computation or a single-commodity minimum-cost flow problem – thus this algorithm is a decomposition method.

The Shahrokhi-Matula result spurred a great deal of research that generalized the techniques to broader packing and covering problems, gradually reducing the dependence of the iteration count on ϵ to finally obtain ϵ^{-2}, reducing the overall complexity to $O(\epsilon^{-2})$ times a polynomial in the size of the graph, and also simplifying the overall approach. See [14, 16, 9, 10, 21, 22, 8, 5] for details; [22, 8, 5] attain the best bounds for the maximum concurrent flow problem. All of these algorithms rely, sometimes implicitly, on the exponential potential function, and can be viewed as first-order methods. [27] uses a logarithmic potential function. [4] shows that the "flow deviation" algorithm for the maximum concurrent flow problem in [7] yields an $O(\epsilon^{-2})$ algorithm; this time using a rational barrier function.

A natural issue is that of proving lower bounds on the number of iterations needed to obtain an ϵ-optimal solution to a packing problem; and, in particular, the dependence of the iteration count on ϵ. Klein and Young [15] studied the complexity of $\textsc{Pack}(A, Q)$ assuming that there exists an oracle that, given $c \in \mathbf{R}^n$, returns an optimal *extreme point* for the linear program $\min\{c^T x : x \in Q\}$. The main result in [15] is that (under appropriate assumptions) the number of oracle calls needed to find an ϵ-optimal solution to a packing problem an be as large as $\Omega(\rho\epsilon^{-2} \log m)$. Here $\rho = \max_{x \in Q} \max_{1 \leq i \leq m} a_i^T x$ (known as the *width* of the problem). This result applies when $\rho\epsilon^{-2} = O(m^{0.5-\delta})$ for some $\delta > 0$. In essence, thus, the result amounts to an $\Omega(m^{0.5})$ lower bound on the iteration count. The dependence on ϵ implied by the bound is striking, and it raises the question of whether ϵ^{-2} is indeed a lower bound on the dependence on ϵ for the number of iterations required by a decomposition method. We note here that the assumption that the optimization oracle returns an *extreme point* of Q is critical in the proof in [15].

For some time a parallel line of research has been followed in the nondifferentiable optimization community. Roughly stated, the focus of the work has been to produce solutions with small *absolute* error, rather than a small *relative* error as has been the focus of the algorithms community. This line of work has produced a number of interesting ideas and results (see [3] for references). Among them are decomposition algorithms that solve the generalized packing problem within additive error ϵ in $O(\epsilon^{-2})$ iterations – however, these are not polynomial-time algorithms, even for fixed ϵ, in that the $O()$ notation hides constants that depend on the matrix A and the set Q. Recently Nesterov [19] (also see [18]) obtained a major result: a decomposition algorithm that solves min-max problems to within additive error ϵ in $O(\epsilon^{-1})$ iterations. Again, this algorithm is, in general, not a polynomial-time algorithm, even for fixed ϵ. The algorithm combines, in a novel way, the (essentially folklore) idea of making use of old iterate information with that of an approximate second-order approximation of the exponential potential function. In Section 2.2 we present a streamlined analysis of the key ideas in [19].

1.2 New results

We show how to adapt the technique in [19] to obtain an ϵ-optimal solution (i.e., a solution with a *relative* error ϵ) to $\textsc{Pack}(A, Q)$ by solving at most

$$O\Big(\epsilon^{-1}\sqrt{Kn\log m}\Big)$$

convex, separable quadratic programs over sets of the form

$$Q(\lambda_u) \doteq \{x \in Q \,:\, 0 \le x_j \le \lambda_u,\ 1 \le j \le n\}, \tag{2}$$

where $\lambda_u > 0$, K denotes the maximum number of nonzero terms in any row of A. While some of the key ideas in this paper are derived from those in [19], the paper is self-contained. We also adapt a binary search technique from [9]. It is quite possible that in our iteration bound some of the constants and the dependency on m, n and K can be improved, with a somewhat more complex analysis. Also note that in the block-angular case each optimization optimizations over $Q(\lambda)$ breaks up up into a separate problem over each block.

Also, suppose that Q is given by a (linear) optimization oracle. It follows that the separation problem over $Q(\lambda)$ can be solved by a polynomial number of oracle calls. Under appropriate assumptions([24]; see [12]) this implies that a convex QP over $Q(\lambda)$ can be solved by making a a polynomial number of calls to the optimization oracle for Q.

The generalized packing problem can be reduced to a fractional packing problem as follows. Let N be the number of non-zeros in A. For each entry $a_{ij} > 0$ introduce a new variable y_{ij} and a new constraint

$$y_{ij} - a_{ij}x_{ij} \;=\; 0. \tag{3}$$

Let

$$P = \Big\{y \in \mathbf{R}^N : \exists x \in Q \text{ such that (3) holds for all } a_{ij} > 0\Big\}.$$

It follows that

$$\lambda^*_{A,Q} = \min_{y \in P} \max_{1 \le i \le m} \sum_{j\,:a_{ij}>0} y_{ij}. \tag{4}$$

Furthermore, P is closed convex, and because of (3), a convex separable quadratic program over P reduces to one over Q. Hence, our result for fractional packing problems implies that, for any $\epsilon \in (0,1)$, the number of iterations required to compute an ϵ-optimal solution for the generalized packing problem is

$$O\Big(\epsilon^{-1}\sqrt{N \log m}\Big).$$

Using ideas derived from [17], we show how the quadratic programs can be approximated by piecewise-linear programs. For the special case of maximum concurrent flows, this leads to an algorithm that computes ϵ-optimal flow by solving

$$O^*\left(\epsilon^{-1}K^2E^{\frac{1}{2}}\left(L_U + |E|\lceil \log D\rceil + |E|\log(\frac{1}{\epsilon})\right) + \epsilon^{-1}|V||E|\right)$$

shortest path problems, where L_U denotes the number of bits needed to store the capacities, D is the sum of all demands and K is the number of commodities grouped by source.

This paper is organized as follows. Section 2 presents our results on packing problems. Section 3 specializes our results to the maximum concurrent flow problem with rational capacities and demands. Section 4 describes the application of our techniques to the maximum multicommodity flow algorithms. Section 5 considers covering problems, and section 6 considers mixed packing and covering problems.

1.3 Comparison with previous results

First we note that our results appear to violate the $\Omega(\epsilon^{-2})$ Klein-Young lower bound. This is not a contradiction since our algorithms bypass the requirements needed for the analysis in [15] to hold – in particular, we dynamically change the bounds on the variables. This feature will, in general, result in iterates in the *interior* of the set Q. In addition, some of our algorithms solve quadratic programs over Q, again leading to interior solutions.

The previous fastest algorithms for finding ϵ-optimal solutions to generalized packing problems have a worst-case complexity that grows proportional to ϵ^{-2}. Our methods improve on this, but at the cost of increasing the dependence on n and m. Hence, for ϵ fixed or moderately small, our algorithms are *not* the fastest. Note that, for example, the algorithm for maximum concurrent flows in [5] has complexity $O^*(\epsilon^{-2}|E|(K+|E|))$, where K is the number of origin-destination commodity pairs. See also [22, 8] for similar bounds.

Another class of algorithms worthy of comparison are the standard interior-point polynomial-time algorithms for linear programming. Even though these are certainly not decomposition methods, their running time has a dependence on ϵ that grows like $\log \epsilon^{-1}$, and hence for very small ϵ these algorithms are certainly faster than ours. Within this class, the algorithms proposed by Kapoor and Vaidya [13], and Vaidya [25, 26] deserve special attention. These algorithms are *not* decomposition algorithms and when applied to a packing problem, [26] computes as a subroutine the inverse $n \times n$ matrices ([14] restates this as $m \times m$). Inspite of this, it is possible that the algorithms in [25, 26, 13] are faster than ours even for moderate ϵ.

In summary, the primary contribution of this paper is to show that there exist decomposition methods for which the iteration count for computing ϵ-optimal solutions to generalized packing problems (among others) grows like $O(\epsilon^{-1})$ and is polynomial in n and m. We expect that the actual running *time* of our algorithms can be improved using ideas such as those in [22, 8, 5], and possibly using randomization as in [14].

1.4 Notation

In this paper, an ϵ-optimal solution will denote a solution that has a *relative* error at most ϵ. We will also at various points compute solutions with a given *absolute* error. In all of these cases we clearly indicate this. We will use the standard $O^*()$ notation to supress polylog factors in parameters such as m, n and K. Also, since our focus in this paper is to design decomposition algorithms with improved dependence on ϵ, we will occasionally state $O()$ and $O^*()$ bounds in terms of their dependence on ϵ, while suppressing polynomial factors in m, n and K.

2 Fractional packing problems $\textsc{Pack}(A, Q)$

In this section we present our algorithm for $\textsc{Pack}(A, Q)$. As with many of the previous algorithms, our algorithm consists of an outer binary search procedure for refining an estimate for $\lambda^*_{A,Q}$ and an inner procedure that is a potential reduction algorithm.

2.1 Outer loop: Binary search

In this section we abbreviate $\lambda^*_{A,Q}$ as λ^*. We assume that we are given bounds $\lambda_l \leq \lambda^* \leq \lambda_u$ such that $\lambda_u \leq O(\min\{m, K\})\lambda_l$. Such bounds can be computed in polynomial time, see [3, 9] for details. Next, these bounds are refined using a binary search procedure introduced in [9] (see [3] for an alternate derivation). In the BINARY SEARCH procedure below, ABSOLUTE(Q,A,λ_u, δ) denotes any algorithm that returns an $x \in Q$ such that $\lambda(x) \leq \lambda^* + \delta$, i.e. an x that has an *absolute* error less than δ.

BINARY SEARCH

```
Input: values $(\lambda_l, \lambda_u)$ with $\lambda_l \leq \lambda^* \leq \lambda_u \leq 2\min\{m, K\})\lambda_l$
Output: $\hat{y} \in Q$ such that $\lambda(y) \leq (1+\epsilon)\lambda^*$
while $(\lambda_u - \lambda_l) \geq \epsilon\lambda_l$ do

    set $\delta = \frac{1}{3}(\lambda_u - \lambda_l)$
    set $\hat{x} \leftarrow$ ABSOLUTE$(Q, A, \lambda_u, \delta)$
    if $\lambda(\hat{x}) \geq \frac{1}{3}\lambda_l + \frac{2}{3}\lambda_u$ set $\lambda_l \leftarrow \frac{2}{3}\lambda_l + \frac{1}{3}\lambda_u$
    else set $\lambda_u \leftarrow \frac{1}{3}\lambda_l + \frac{2}{3}\lambda_u$

return $\hat{x}$
```

In section 2.2 we construct an algorithm ABSOLUTE with the following properties.

Theorem 1 *There exists an algorithm* ABSOLUTE$(A, Q, \lambda_u, \delta)$ *that computes, for any $\delta \in (0, \lambda_u)$, an $\hat{x} \in Q$ with $\lambda(\hat{x}) \leq \lambda^* + \delta$ by solving $O\left(\sqrt{Kn}\log m\frac{\lambda_u}{\delta}\right)$ separable convex quadratic programs over $Q(\lambda_u)$.*

As a consequence, we have the following complexity bound for the above BINARY SEARCH procedure.

Corollary 2 *The complexity of the* BINARY SEARCH *procedure is $O(\epsilon^{-1}C_q\sqrt{Kn\log m})$ plus a polynomial in K, n and m, where C_q is the cost of solving a convex separable quadratic program over $Q(\lambda_u)$.*

PROOF: It is easy to check that in each iteration the gap $(\lambda_u - \lambda_l)$ is decreased by a factor of 2/3. Thus, the total number H of iterations is $O(\log(\epsilon/mK))$ and the total number of quadratic programs solved by BINARY SEARCH is

$$\sqrt{Kn\log m}\sum_{h=0}^{H}\left(\frac{3}{2}\right)^h .$$

The result follows from the fact that the last term dominates the sum. ■

2.2 Inner loop: ABSOLUTE$(A, Q, \lambda_u, \delta)$

In this section we describe the ABSOLUTE$(A, Q, \lambda_u, \delta)$ algorithm. We construct this algorithm using techniques from [19].

Define $\bar{Q} \in \mathbf{R}^{2n}$ as follows.

$$\bar{Q} \doteq \left\{(x, y) \in \mathbf{R}^{2n} : x \in Q, x_j = \lambda_u y_j,\ y_j \le 1, j = 1, \dots, n\right\}.$$

Since $\lambda^* \le \lambda_u$, $\bar{Q} \ne \emptyset$. Let P denote the *projection* of $\bar{Q}$ onto the space of the y variables, i.e.

$$P = \{y \in \mathbf{R}^n : \exists x \text{ s.t. } (x, y) \in \bar{Q}\}.$$

If Q is a polyhedron, so are $\bar{Q}$ and P. Moreover, $P \subseteq [0, 1]^n$. Define

$$\lambda_P^* \doteq \min\left\{\lambda(y) = \max_{1\le i\le m}\{a_i^T y\} : y \in P\right\}.$$

Then it follows that $\lambda_P^* = \lambda^*/\lambda_u \le 1$.

In this section we describe an algorithm that, for any $\gamma \in (0, 1)$, computes $\hat{y} \in P$, with $\lambda(\hat{y}) \le \lambda^*(P) + \gamma$, by solving $O(\gamma^{-1}\sqrt{Kn\log m})$ separable convex quadratic programs over P. Note that these programs reduce to separable convex quadratic programs over $Q(\lambda_u)$, and in the case of multicommodity flow problems, these break up into separable convex quadratic min-cost flow problems over each commodity. Choosing $\gamma = \frac{\delta}{\lambda_u}$ will accomplish the objective of this section.

2.2.1 Potential reduction algorithm

For the purposes of this section, we assume that we are given a matrix $A \in \{0, 1\}^{m\times n}$ with at most K non-zeros per row, a (nonempty) closed convex set $P \subseteq [0, 1]^n$, and a constant $\gamma \in (0, 1)$. Furthermore, it is known that $\lambda_P^* \le 1$.

Define the **potential function** $\Phi(x)$ as follows [9, 21].

$$\Phi(x) \doteq \frac{1}{|\alpha|}\ln\left(\sum_i e^{\alpha a_i^T x}\right),$$

It is easy to show (see [9] for details) that for $\alpha \ge 0$

$$\lambda(x) \le \Phi(x) \le \lambda(x) + \frac{\ln m}{\alpha}, \quad \forall x \in P. \tag{5}$$

Let $\Phi^* \doteq \min\{\Phi(x) : x \in P\}$ and choose $\alpha = \frac{2\ln m}{\gamma}$. Then (5) implies that we only need to compute $x \in P$ with $\Phi(x) \leq \Phi^* + \gamma/2$.

In what follows, the notation $\langle x, y\rangle$ will denote the usual Euclidean inner product $x^T y$.

Algorithm QP

```
Input: $P \subseteq [0,1]^n$, 0/1 matrix $A$, $\alpha = \frac{\ln m}{\gamma}$, $L = \gamma^{-1}\sqrt{8Knln(m)}$
Output: $\hat{y} \in P$ such that $\lambda(\hat{y}) \leq \lambda_P^* + \gamma$
choose $x^{(0)} \in P$. set $t \leftarrow 0$.
while $(t \leq L)$ do
    $g^{(t)} \leftarrow \nabla\Phi(x^{(t)}$
    $y^{(t)} \leftarrow \text{argmin}_{x\in P}\Big\{\frac{K|\alpha|}{2}\sum_{j=1}^{n}(x_j - x_j^{(t)})^2 + \langle g^{(t)}, x - x^{(t)}\rangle\Big\}$
    $S_t(x) \leftarrow \frac{2}{(t+1)(t+2)}\Big\{K|\alpha|\sum_{j=1}^{n}(x_j - x_j^0)^2 + \sum_{h=0}^{t}(h+1)[\Phi(x^{(h)}) + \langle g^{(h)}, x - x^{(h)}\rangle]\Big\}$
    $z^{(t)} \leftarrow \text{argmin}_{x\in P}\{S_t(x)\}$
    $x^{(t+1)} \leftarrow \Big(\frac{2}{t+3}\Big)z^{(t)} + \Big(\frac{t+1}{t+3}\Big)y^{(t)},$
    $t \leftarrow t+1$
return $y^{(L)}$
```

The following result is established in Section 2.2.2.

Theorem 3 *For any* $t \geq 0$, $\Phi(y^{(t)}) \leq S_t(z^{(t)})$.

Theorem 3 implies the following corollary.

Corollary 4 $\Phi(y^{(L)}) \leq \Phi^* + \gamma/2$.

Proof: Fix $t \geq 0$. Let $x^* = \text{argmin}_{x\in P}\{\Phi(x)\}$. By definition

$$\begin{aligned} S_t(z^t) &\leq S_t(x^*), \\ &= \frac{2}{(t+1)(t+2)}\Big\{K\alpha\sum_{j=1}^{n}(x_j^* - x_j^{(0)})^2 + \sum_{h=0}^{t}(h+1)[\Phi(x^{(h)}) + \langle g^{(h)}, x^* - x^{(h)}\rangle]\Big\}, \\ &\leq \frac{2Kn\alpha}{(t+1)(t+2)} + \Phi^*, \qquad (6) \end{aligned}$$

where (6) follows the following facts:

$$\begin{aligned} 0 \leq x_i^*, x_i^{(0)} \leq 1 &\Rightarrow (x_i^* - x_i^{(0)}) \leq 1, \\ \Phi \text{ convex} &\Rightarrow \Phi(x^{(h)}) + \langle g^{(h)}, x^* - x^{(h)}\rangle \leq \Phi(x^*). \end{aligned}$$

Theorem 3 implies that $\Phi(y^{(t)}) \leq S_t(z^{(t)})$. Thus, we have that $\Phi(y^{(t)}) - \Phi^* \leq \gamma/2$ for $t \geq L = \frac{\sqrt{8Kn\ln m}}{\gamma}$. ■

2.2.2 Proof of Theorem 3

The following result is a refinement of the Taylor series bound in [9] (see also [3, 20]). The proof is provided in Appendix A.1.

Lemma 5 *For all $x, y \in P$, we have $\Phi(y) \leq \Phi(x) + \langle \nabla\Phi(x), y - x\rangle + \frac{K|\alpha|}{2}\sum_j (y_j - x_j)^2$.*

The rest of the proof of Theorem 3 closely mirrors the development in Section 3 of [19]. The proof is by induction on t. To establish the base case $t = 0$, note that

$$\begin{aligned}
S_0(z^{(0)}) &= K|\alpha|\sum_{j=1}^{n}(z_j^{(0)} - x_j^0)^2 + \Phi(x^0) + \langle g_0, z^{(0)} - x^0\rangle \\
&\geq \frac{K|\alpha|}{2}\sum_{j=1}^{n}(z_j^{(0)} - x_j^0)^2 + \Phi(x^0) + \langle g_0, z^{(0)} - x^0\rangle \\
&\geq \frac{K|\alpha|}{2}\sum_{j=1}^{n}(y_j^{(0)} - x_j^0)^2 + \Phi(x^0) + \langle g_0, y^{(0)} - x^0\rangle \qquad (7)\\
&\geq \Phi(y^{(0)}), \qquad (8)
\end{aligned}$$

where (7) follows from definition of $y^{(0)}$ and (8) follows from Lemma 5.

To establish the induction step, assume that $\Phi(y^{(t)}) \leq S_t(z^{(t)})$. By definition, we have

$$\begin{aligned}
S_{t+1}(x) &= \Big(\frac{t+1}{t+3}\Big)S_t(x) \\
&\quad + \Big(\frac{2}{t+3}\Big)[\Phi(x^{t+1}) + \langle g^{(t+1)}, x - x^{(t+1)}\rangle].
\end{aligned}$$

Since $\nabla^2 S_t = \frac{4K|\alpha|}{(t+1)(t+2)}I$, and S_t is minimized at z_t, we obtain

$$\begin{aligned}
S_{t+1}(x) &\geq \Big(\frac{t+1}{t+3}\Big)S_t(z^{(t)}) + \frac{2K|\alpha|}{(t+2)(t+3)}\sum_j (x_j - z_j^{(t)})^2 \\
&\quad + \frac{2}{t+3}[\Phi(x^{(t+1)}) + \langle g^{(t+1)}, x - x^{(t+1)}\rangle]. \qquad (9)
\end{aligned}$$

By the induction hypothesis and the convexity of Φ, it follows that $S_t(z^{(t)}) \geq \Phi(y^{(t)}) \geq \Phi(x^{(t+1)}) + \langle g^{(t+1)}, y^{(t)} - x^{(t+1)}\rangle$. Substituting this bound in (9) and rearranging terms, we get

$$\begin{aligned}
S_{t+1}(x) &\geq \Phi(x^{(t+1)}) + \frac{2K|\alpha|}{(t+2)(t+3)}\sum_j (x_j - z_j^{(t)})^2 \\
&\quad + \Big\langle g^{(t+1)}, \Big(\frac{2}{t+3}\Big)x + \Big(\frac{t+1}{t+3}\Big)y^{(t)} - x^{(t+1)}\Big\rangle, \\
&\geq \Phi(x^{(t+1)}) + \frac{2K|\alpha|}{(t+3)^2}\sum_j (x_j - z_j^{(t)})^2 \\
&\quad + \Big\langle g^{(t+1)}, \Big(\frac{2}{t+3}\Big)x + \Big(\frac{t+1}{t+3}\Big)y^{(t)} - x^{(t+1)}\Big\rangle, \\
&= \Phi(x^{(t+1)})
\end{aligned}$$

$$+\frac{K|\alpha|}{2}\sum_j\Big(\Big(\frac{2}{t+3}\Big)x_j+\Big(\frac{t+1}{t+3}\Big)y_j^{(t)}-x_j^{(t+1)}\Big)^2$$
$$+\Big\langle g^{(t+1)},\Big(\frac{2}{t+3}\Big)x+\Big(\frac{t+1}{t+3}\Big)y^{(t)}-x^{(t+1)}\Big\rangle, \tag{10}$$

where (10) is obtained by substituting $(\frac{2}{t+3})z^{(t)}=x^{(t+1)}-(\frac{t+1}{t+3})y^{(t)}$. Note that for $x\in P$, $(\frac{2}{t+3})x+(\frac{t+1}{t+3})y^{(t)}\in P$ as well. Thus, the expression in (10) is lower bounded by:

$$\begin{aligned}
S_{t+1}(x) &\geq \Phi(x^{(t+1)})+\min_{y\in P}\Big\{\frac{K|\alpha|}{2}\sum_j(y_j-x_j^{(t+1)})^2+\langle g^{(t+1)},y-x^{(t+1)}\rangle\Big\}\\
&= \Phi(x^{(t+1)})+\frac{K|\alpha|}{2}\sum_j(y_j^{t+1}-x_j^{(t+1)})^2+\langle g^{(t+1)},y^{t+1}-x^{(t+1)}\rangle, &(11)\\
&\geq \Phi(y^{(t+1)}, &(12)
\end{aligned}$$

where (11) follows from the definition of $y^{(t+1)}$ and (12) follows from Lemma 5.
Remark: Note that the proofs of Corollary 4, Lemma 5 and Theorem 3 do not require $\alpha>0$.

2.2.3 Notes on algorithm QP

When defining the set $\bar{Q}$ in Section 2.2 we impose the bounds $y_j\leq 1$ for all j. This bounding technique is essential in that it effectively reduces width to at most n. This technique can be read in the approach used in [9] and [14]. More recent algorithms, such as [8] and [5], do not directly rely on this technique (arguably, their method for controlling width is *similar* in that they directly control the magnitudes of the solution variables) and potentially our overall scheme could be improved along the lines of [8] and [5]. The scaling technique in (3) is essential for the analysis of algorithm QP to go through; the scaling methods implicit in [9] and [14] will not work in this context.

2.3 Piecewise-linear approximation

ALGORITHM QP requires one to solve a sequence of separable quadratic programs over P. In this section we describe a general method for approximating the the separable convex quadratic function by piecewise-linear function with arbitrarily small error. This method is derived from one given in Minoux [17],

Fix $\sigma>0$ and $w\geq 0$. Define a continuous convex piecewise-linear approximation $\mathcal{L}_{\sigma,w}(v)$ to the quadratic function $\frac{1}{2}(v-w)^2:\mathbf{R}_+\mapsto\mathbf{R}_+$ as follows

$$\mathcal{L}_{\sigma,w}(v)\ \doteq\ \frac{1}{2}q^2\sigma^2+\frac{w^2}{2}-wv+\Big(q+\frac{1}{2}\Big)\sigma(v-q\sigma),\quad v\in[q\sigma,(q+1)\sigma),\ q\in\mathbf{Z}_+.$$

For $q\in\mathbf{Z}_+$, the derivative $\mathcal{L}'_{\sigma,w}(q\sigma)$ is not defined. The *left-derivative* $\mathcal{L}^-_{\sigma,w}(q\sigma)=\Big(q-\frac{1}{2}\Big)\sigma-w$ and the *right-derivative* $\mathcal{L}^+_{\sigma,w}(q\sigma)=\Big(q+\frac{1}{2}\Big)\sigma-w$. For $v\in(q\sigma,(q+1)\sigma)$, $q\in\mathbf{Z}$, the derivative $\mathcal{L}'_{\sigma,w}(v)$ exists and $\mathcal{L}'_{\sigma,w}(v)=\mathcal{L}^+_{\sigma,w}(v)=\mathcal{L}^-_{\sigma,w}(v)=\Big(q+\frac{1}{2}\Big)\sigma-w$. The following properties are easy to obtain.

Lemma 6 *The following hold for any $\sigma>0$ and $w\in\mathbf{R}_+$.*

(i) For $q \in \mathbf{Z}_+$, $\mathcal{L}_{\sigma,w}(q\sigma) = \frac{1}{2}(q\sigma - w)^2$.

(ii) For $v \in \mathbf{R}_+$, $\frac{1}{2}(v-w)^2 \leq \mathcal{L}_{\sigma,w}(v) \leq \frac{1}{2}(v-w)^2 + \frac{\sigma^2}{8}$,

(iii) For $v \in \mathbf{R}_+$, $v - w - \frac{\sigma}{2} \leq \mathcal{L}^-_{\sigma,w}(v) \leq v - w \leq \mathcal{L}^+_{\sigma,w}(v) \leq v - w + \frac{\sigma}{2}$.

2.4 Algorithm QP$_\sigma$

Algorithm QP$_\sigma$ approximates each convex quadratic term $\frac{1}{2}(x_j - \bar{x}_j)^2$ by the piecewise-linear $\mathcal{L}_{\sigma_j,\bar{x}_j}(x_j)$ with possibly different $\sigma_j \in (0,1)$ for each variable x_j. Thus, each optimization problem in the course of Algorithm QP$_\sigma$ minimizes a piecewise-linear function over the convex set P.

Algorithm QP$_\sigma$

```
Input: $P \subseteq [0,1]^n$, $A$, $\alpha = \frac{\ln m}{\gamma}$, $L = \gamma^{-1}\sqrt{16Kn\ln(m)}$, $\sigma_j \leq 2^{-p}$, $p = \lceil 3\ln(L)\rceil = \lceil \frac{3}{2}\ln(16Kn\ln(m)) + 3\ln(\frac{1}{\gamma})\rceil$
Output: $\hat{y} \in P$ such that $\lambda(\hat{y}) \leq \lambda^*_P + \gamma$
choose $\hat{x}^0 \in P$. set $t \leftarrow 0$.
while $(t \leq L)$ do
    $g^{(t)} = \nabla\Phi(\hat{x}^{(t)})$
    $\hat{y}^{(t)} \leftarrow \text{argmin}_{x\in P}\Big\{K|\alpha|\sum_{j=1}^n \mathcal{L}_{\sigma_j,\hat{x}_j^{(t)}}(x_j)\langle g^{(t)}, x - \hat{x}^{(t)}\rangle\Big\}$,
    $\hat{S}_t(x) \leftarrow \frac{2}{(t+1)(t+2)}\Big\{2K|\alpha|\sum_{j=1}^n \mathcal{L}_{\sigma_j,\hat{x}_j^{(0)}}(x_j) + \sum_{h=0}^t (h+1)[\Phi(\hat{x}^h) + \langle g_h, x - \hat{x}^h\rangle]\Big\}$
    $\hat{z}^{(t)} \leftarrow \text{argmin}_{x\in P}\{\hat{S}_t(x)\}$
    $\hat{x}^{(t+1)} \leftarrow \Big(\frac{2}{t+3}\Big)\hat{z}^{(t)} + \Big(\frac{t+1}{t+3}\Big)\hat{y}^{(t)}$
    $t \leftarrow t+1$
return $y^{(L)}$
```

In view of Lemma 6, we would expect that Algorithm QP$_\sigma$ successfully emulates Algorithm QP$_\sigma$ if the σ_j are small enough. In the Appendix we provide a proof of the following fact:

Theorem 7 *For any $t \geq 0$,*

$$\Phi(\hat{y}^{(t)}) \leq \hat{S}_t(\hat{z}^{(t)}) + \Big(\frac{5K|\alpha|}{2}\Big)\Big(\sum_{h=1}^t \frac{1}{h^2} + t\Big)\Big(\sum_k \sigma_j\Big).$$

This theorem implies the correctness of Algorithm QP$_\sigma$:

Corollary 8 *The output $y^{(L)}$ of* Algorithm QP$_\sigma$ *satisfies $\Phi(y^{(L)}) \leq \Phi^* + \gamma/2$.*

Proof: Suppose $\sigma_j \leq 2^{-p}$. Using Theorem 7 and Lemma 6(ii), we obtain

$$\Phi(\hat{y}^{(t)}) - \Phi^* \quad \leq \quad \frac{2Kn|\alpha|}{(t+1)(t+2)}\left(1 + \frac{1}{4}(2^{-2p})\right)$$

$$
\begin{aligned}
&+ \Big(\frac{5K|\alpha|n}{2}\Big)\Big(\sum_{h=1}^{t}\frac{1}{h^2}+t\Big)2^{-p},\\
< \;& K|\alpha|n\Big(\frac{2+2^{-(2p+1)}}{t^2}+\frac{5}{2}(2+t)2^{-p}\Big).
\end{aligned}
$$

Suppose $t \geq 2$ and choose $p \geq 3\ln t$. Then $\Phi(\hat{y}^{(t)}) - \Phi^* \leq \frac{8K|\alpha|n}{t^2}$. A simple calculation now establishes the result. ■

3 Concurrent flows with rational capacities and demands

In this section we discuss the special case of maximum concurrent flow with rational capacities and show that the piecewise linear approximation introduced in section 2.3 can be solved efficiently for this special case.

Suppose we have a network $G = (V, E)$ with $|V|$ nodes, $|E|$ edges and K commodities. We assume that the capacity u_e of every edge e is a positive rational. The demand vector d_k of every commodity k is also assumed to be a rational vector. Since scaling capacities and demands by a common positive constant does not change the value of the problem, we assume that all capacities and demands are integers. Let $f_{k,e}$ denote the flow associated with commodity k on edge e and let f_k denote the $|E|$-vector with entries $f_{k,e}$. Then the maximum concurrent flow problem is given by

$$
\begin{aligned}
\lambda^* = \;& \min \;\; \lambda,\\
& \text{s.t.} \;\; \textstyle\sum_{k=1}^{K} f_{k,e} \leq \lambda u_e, \qquad \forall k, e,\\
& \qquad N f_k = d_k, \quad f_k \geq 0, \quad k = 1, \ldots, K,
\end{aligned}
$$

where N denotes the node-edge incidence matrix of the network. Let $\mathcal{F} = \{f : Nf_k = d_k, f_k \geq 0, k = 1, \ldots, K\}$ denote the polyhedron of feasible flows.

In order to describe our piecewise-linear approach, we next review how the procedures we described in the prior sections would apply to the concurrent flow problem.

Step 1: Define new scaled variables $g_{k,e} = f_{k,e}/u_e$. This scaling leaves the objective unchanged and the constraints are transformed to

$$
\begin{aligned}
&\textstyle\sum_k g_{k,e} \leq \lambda, \quad e = 1, \ldots, |E|,\\
&g \in Q = \{g \in \mathbf{R}^{K\times|E|} : \exists f \in \mathcal{F} \text{ with } g_{k,e} = f_{k,e}/u_e \;\forall k, e\}
\end{aligned}
$$

The problem is now in the canonical form described in Section 1, with $m = |E|$ and $n = K|E|$.
Step 2: After i iterations of Binary Search (see Section 2.1), we get lower and upper bounds λ_l and λ_u with

$$
\frac{\lambda_u - \lambda_l}{\lambda_l} \;\leq\; \Big(\frac{2}{3}\Big)^i O(\min\{m, K\}). \tag{13}
$$

Let $\delta = \frac{1}{3}(\lambda_u - \lambda_l)$. We seek a vector g with $\max_e \sum_k g_{k,e} \leq \lambda^* + \delta$. Upon computing g, we either reset $\lambda_l \leftarrow \lambda_l + \delta$ or $\lambda_u \leftarrow \lambda_u - \delta$.

STEP 3: Let $P(\lambda_u) \doteq \{z : \exists g \in Q \text{ with } z = g/\lambda_u, 0 \le z \le 1\}$. Procedure ABSOLUTE computes the vector g needed in Step 2 by approximately solving the scaled optimization problem

$$\begin{array}{ll} \min & \lambda, \\ \text{s.t.} & \sum_{k=1}^{K} x_{k,e} \le \lambda, \quad e = 1, \ldots, |E|, \\ & x \in P(\lambda_u). \end{array}$$

The value of this problem is $\lambda^*/\lambda_u \le 1$, and, ABSOLUTE computes a feasible x with $\max_e \sum_k x_{k,e} \le \lambda^*/\lambda_u + \gamma$, where $\gamma = \delta/\lambda_u$.

In section 2.2.1 we show that in order to achieve the goal of procedure ABSOLUTE it is sufficient that x satisfy $\Phi(x) \le \Phi^* + \gamma/2$. Corollary 8 in section 2.4 establishes that the output $\hat{y}^{(t)}$ produced by Algorithm QP$_\sigma$ will satisfy this condition, provided:

(a) For each commodity k and edge e (i.e. each variable $x_{k,e}$), we have $\sigma_{k,e} \le 2^{-p}$.

(b) $p \ge \lceil \frac{3}{2} \ln(16Kn\ln(m)) + 3\ln(\frac{1}{\gamma}) \rceil \doteq \bar{p}$, and

(c) $t \ge \frac{\sqrt{16Kn \ln m}}{\gamma}$.

Next we describe how to achieve (a)-(c) in the particular framework of the maximum concurrent flow problem. Note that in terms of the initial flow variables $f_{k,e}$, we have $x_{k,e} = \frac{1}{\lambda_u u_e} f_{k,e}$. We apply the framework developed in section 2.3 to the concurrent flow problem as follows:

(1) We set $p = \bar{p} + \lceil \log D \rceil$, where D is the sum of all demands, and

$$\sigma_{k,e} \;=\; \frac{2^{-p}}{u_e}, \quad \forall k, e. \tag{14}$$

This satisfies requirement (a) of Step 3.

(2) We modify BINARY SEARCH as follows. Every time a new upper bound λ_u is computed, we replace it by a *relaxed* bound $\hat{\lambda}_u \ge \lambda_u$, chosen so that $\frac{2^p}{\hat{\lambda}_u} = \left\lfloor \frac{2^p}{\lambda_u} \right\rfloor$. Note that $\lambda_u \le D$, and so $\hat{\lambda}_u \le \frac{\lambda_u}{1-\lambda_u 2^{-p}} \le \lambda_u(1 + O(2^{-\bar{p}}))$. Since $\bar{p} = \frac{3}{2}\ln(Kn\ln(m)) + 3\ln(\frac{1}{\gamma})$, where $\gamma = \delta/\lambda_u$, we have that

$$\frac{\hat{\lambda}^U - \lambda_l}{\lambda_u - \lambda_l} \le 1 + \frac{\lambda_u O\left(2^{-\bar{p}}\right)}{\lambda_u - \lambda_l} \le 1 + O\left(\frac{2^{-\bar{p}}}{3\gamma}\right) = 1 + o(1).$$

Thus, up to constants the the complexity bound in Corollary 2 remains unchanged. For simplicity, in what follows we will use the notation λ_u to refer to the relaxed upper bound.

3.1 Solving the piecewise-linear problems

In this section we show how the modifications (1) and (2) above allow one to efficiently solve the piecewise-linear problems encountered in algorithm QP$_\sigma$.

The generic piecewise-linear problem that we need to solve is of the form

$$\begin{array}{ll} \min & \sum_{k,e} \bar{\mathcal{L}}_{k,e}(x_{k,e}) \\ \text{s. t.} & x \in P(\lambda_u), \end{array} \tag{15}$$

where $\bar{\mathcal{L}}_{k,e}(\cdot) = \mathcal{L}_{\sigma_{k,e},\bar{x}_{k,e}}(\cdot)$ is a continuous convex piecewise-linear function with breakpoints at the integer multiples of $\frac{2^{-p}}{u_e}$ and with pieces of strictly increasing slope.

For every k and e, define $r_{k,e} = 2^p u_e x_{k,e}$. In terms of the initial flow variables $f_{k,e}$, we have $r_{k,e} = \frac{2^p}{\lambda_u} f_{k,e}$. Thus, after the change of variables the optimization problem is of the form

$$\begin{array}{ll} \min & \sum_{k,e} \mathcal{L}_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = \frac{2^p}{\lambda_u} d_k, \quad \forall k, \\ & r_{k,e} \le 2^p u_e, \quad \forall k, e, \end{array} \tag{16}$$

where $\mathcal{L}_{k,e}$ is continuous convex piecewise-linear function with breakpoints at the integers and pieces of strictly increasing slope. The optimization problem (16) just a min-cost flow problem, with integral demands and capacities. In summary, as discussed in the previous section, given λ_u we will have to solve $2t$ problems of the form (16), where t is as in (c).

We solve each such problem using an approach similar to that described in [17] and in [1] (Chapter 14). This approach is reminiscent of the cost-scaling (or capacity scaling) method for solving standard (linear) minimum-cost flow problems. Our algorithm is described in Appendix A.3. The algorithm assumes that a feasible, integral solution to (16) has been already computed (we will account for this work separately). Given such a solution, the algorithm solves problem (16) by performing

$$O\left(\sum_k \sum_e (p + \log u_e)\right) = O(K|E|p + KL_U)$$

shortest path computations, where L_U is the number of bits needed to represent the largest capacity. Since $p = \bar{p} + \lceil \log D \rceil$, over all iterations of algorithm QP_σ the total number of shortest path computations incurred in calls to the algorithm in Appendix A.3 is

$$O\left(\frac{\sqrt{Kn\ln(m)}}{\epsilon}\left(K|E|(\bar{p} + \lceil \log D \rceil) + KL_U\right)\right) = O^*\left(\epsilon^{-1}K^2|E|^{\frac{1}{2}}\left(L_U + |E|\lceil \log D \rceil + |E|\log(\frac{1}{\epsilon})\right)\right),$$

plus lower order complexity terms.

Finally, we account for the complexity of computing a feasible integral solution to (16). Note that all the problems (16) arising in a given call to algorithm QP_σ make use of the same value of λ_u and p – consequently, the feasible integral flows need be computed once per commodity, per call to QP_σ. In total, this will be $O^*(\epsilon^{-1})$ feasible flow computations per commodity. Each such computation essentially amounts to a maximum flow problem. We can solve such a problem using e.g. any augmenting path algorithm (which essentially relies on shortest path computations). The simplest such algorithm (see [1]) peforms $O(|V||E|)$ shortest path computations.

In summary, we now have the following result:

Theorem 9 *An ϵ-optimal solution to a maximum concurrent flow problem, on a graph $G = (V, E)$ with $|V|$ nodes, $|E|$ edges, and K commodities can be computed by solving*

$$O^*\left(\epsilon^{-1}K^2E^{\frac{1}{2}}\left(L_U + |E|\lceil \log D \rceil + |E|\log(\frac{1}{\epsilon})\right) + \epsilon^{-1}|V||E|\right)$$

shortest path problems, plus lower complexity steps where L_U denotes the number of bits needed to store the capacities and D is the sum of demands.

4 Maximum multicommodity flows

In this section we extend the techniques developed in the previous section to the maximum multicommodity flow problem (see [2]). See also [8, 5].

Let (s_k, t_k), $k = 1, \ldots, K$ denote a set of source-sink node pairs. The goal of the maximum multicommodity flow problem is to find a feasible multicommodity flow (one that satisfies the capacity constraints on any edge) that maximizes the sum, over all k, of the amount of flow sent from s_k to t_k. Our approach will be to reduce this problem into an equivalent maximum concurrent flow problem.

Given a multicommodity flow, we will denote by F_k denote the flow sent from s_k to t_k, for $k = 1, \ldots, K$. We denote by F^* the optimal value of the maximum multicommodity flow problem. Finally, let

$$F^L = \max_{k=1,\ldots,K} \{\text{maxflow for commodity } k \text{ alone }\} \tag{17}$$

Then $F^L \leq F^* \leq F^U = KF^L$. Moreover, for any feasible flow and any k we have $F_k \leq F^L$. Suppose we introduce new variables $G_k = F^L - F_k \geq 0$. Then the maximum multicommodity flow problem is equivalent to

$$\begin{aligned} G^* \doteq \ & \min \ \sum_{k=1}^{K} G_k \\ & \text{s.t.} \ \ F_k + G_k = F^L, \quad k = 1, \ldots, K, \qquad (18) \\ & \qquad \sum_{k=1}^{K} f_{k,e} \leq u_e, \quad e \in E, \\ & \qquad f_k \text{ nonnegative and satisfies flow conservation,} \quad k = 1, \ldots, K. \qquad (19) \end{aligned}$$

Clearly, $\sum_k G_k \leq G^U \doteq (K-1)F^L$, and $F^* + G^* = F^U$. We assume that $F^L > 0$, or else the problem is trivial. Further, let $\delta = \min\left\{\frac{\epsilon}{K}, \frac{1}{2}\right\}$, and suppose $(\hat{f}_k, \hat{F}_k, \hat{G}_k)$, $k = 1, \ldots, K$, is a vector satisfying:

(i) $\sum_{k=1}^{K} f_{k,e} \leq (1+\delta)u_e$, $\forall e$,

(ii) constraints (18) and (19), and

(iii) $\sum_k \hat{G}_k \leq (1+\delta)G^*$.

Then $\sum_k \hat{F}_k = F^U - \sum_k \hat{G}_k = F^* + G^* - \sum_k \hat{G}_k$, and consequently

$$\frac{\sum_k \hat{F}_k}{F^*} \geq 1 - \delta\frac{G^*}{F^*} \geq 1 - \delta\frac{G^U}{F^L} = 1 - (K-1)\delta.$$

As a result, the vector $((1+\delta)^{-1}\hat{f}_k)$, $k = 1, \ldots, K$, is an ϵ-optimal solution to the maximum multicommodity flow problem.

In order to find a vector satisfying conditions (i)-(iii), we first establish a non-zero lower bound for G^*. Consider the packing problem

$$\begin{aligned} \min \ & \max_{e\in E}\left\{\frac{\sum_{k=1}^{K} f_{k,e}}{u_e}\right\} \qquad (20) \\ \text{s.t.} \ & F_k = (1-\delta/2)F^L, \quad \forall k, \\ & f_k \text{ satisfies flow conservation,} \quad \forall k. \end{aligned}$$

Note that (20) is a maximum concurrent flow problem. Let $\hat{f}$ denote any $(\delta/4)$-optimal solution for (20). Consider the following two cases:

(a) $\lambda(\hat{f}) \leq 1 + \delta/2$. In this case, $\hat{f}/(1+\delta/2)$ is a feasible flow with total flow $(\sum_k F_k)/(1+\delta/2) \geq (1-\delta)KF^L = (1-\delta)F^U$, i.e. $\hat{f}/(1+\delta/2)$ is a δ-optimal multicommodity flow.

(b) $\lambda(\hat{f}) > 1 + \delta/2$. In this case, the packing problem (20) is infeasible. Thus, for any multicommodity flow satisfying the capacity constraints and flow conservation we have $F_k < (1-\delta/2)F^L$, i.e. $G^k > \delta F^L/2$ for at least one k. Hence, $G^L \doteq \delta F^L/2$ is a valid lower bound for G^*.

We next describe the procedure that computes a vector satisfying (i)-(iii). The procedure maintains two bounds, $0 < G^L \leq G^* \leq G^U$, and a flow vector $\hat{f}$. The bound G^L is initialized as indicated above and $G^U = (K-1)F^L$. The flow vector $\hat{f}$ is initialized as follows. Let $\bar{k}$ denote the commodity attaining the maximum in (17). Then $\hat{f}_{\bar{k}}$ is set equal to any flow vector sending F^L units from $s_{\bar{k}}$ to $t_{\bar{k}}$, while for all other k, the flow vector $\hat{f}_k$ is the zero vector. Thus, the flow $\hat{f}$ attains G^U.

In any interation of the procedure, we set $G = \frac{G^L + G^U}{2}$ and compute an $(\delta/8)$-approximate solution to the packing problem

$$\begin{array}{rll} \lambda^*_G = & \min & \max\left\{\max_{e\in E}\left\{\frac{\sum_{k=1}^K f_{k,e}}{u_e}\right\}, \frac{\sum_k G_k}{G}\right\} \\ & \text{s.t.} & F_k + G_k = F^L, \quad \forall k, \\ & & f_k \text{ satisfies flow conservation}, \quad \forall k. \end{array} \tag{21}$$

Note that (21) is a concurrent flow problem in the graph $G' = (V', E')$, $V' = V \cup \{S, T\}$, $E' = E \cup \{(S,T)\} \cup \{(s_k, S), (T, t_k) : k = 1, \ldots, K\}$, $u_{(s_k,S)} = u_{(T,t_k)} = \infty$, and $u_{(S,T)} = G$ (the flow on edges (s_k, S) and (T, t_k) are both equal to G_k). Consequently, this task can be accomplished in $O^*(\frac{1}{\delta}\log(\frac{1}{\delta}))$ iterations of ALGORITHM QP$_\sigma$ where each iteration solves a piece-wise linear program that reduces to a sequence of shortest path problems.

Let λ_G denote the congestion of any $(\delta/8)$-optimal solution. There are two cases:

(a) $\lambda_G > (1 + \delta/8)$. Then we reset to $G^L \leftarrow G$.

(b) $\lambda_G \leq (1 + \delta/8)$. In this case we reset $G^U \leftarrow (1+\delta/8)G$ and update $\hat{f}$ to be the solution we computed to the packing problem.

The above procedure is repeated until $G^U - G^L \leq (\delta/2)G^L$. At the start of the procedure, $G^L = \delta F^L/2 \geq \delta G^U/2K$, and thus the number of bisection steps is bounded above by $O(\log(\frac{1}{\delta}))$ (neglecting polynomial factors in n, m and K). To validate the procedure, note that each time that G^L is reset, i.e. case (a) holds, we have that $\lambda^*_G > 1$, which implies that $G \leq G^*$, and thus G^L is always a lower bound for G^*. When case (b) applies the flow vector obtained in that iteration satisfies (i), (ii) and $\sum_k \hat{G}_k \leq (1+\delta)G \leq (1+\delta/8)G^U$. We conclude that at termination the vector $\hat{f}$ is as desired.

In summary, the number of iterations of ALGORITHM QP$_\sigma$ needed to find an ϵ-optimal maximum multicommodity flow is bounded above by $O\left(\frac{1}{\epsilon}\log^2(\frac{1}{\epsilon})\right)$, again neglecting polynomial factors in n, m and K.

5 Pure covering problems

The basic covering problem has the following structure:

$$\mu^* = \max_{x \in Q} \min_{1 \le i \le m} \{a_i^T x\}, \tag{22}$$

where $Q \subseteq \mathbf{R}_+^n$ is a compact, convex set, $A = [a_1, \dots, a_m]^T \in \mathbf{R}_+^{m \times n}$ and each positive a_{ij} has value at least 1. For a given $x \in Q$, let $\mu(x) = \min_{1 \le i \le m}\{a_i^T x\}$.

5.1 Upper and lower bounds

In order to start our procedure, we need polynomially separated upper and lower bounds for μ^*. The following technique is used in [21]. For $i = 1, \dots, m$, define

$$\mu_i = \max_{x \in Q}\{a_i^T x\}, \quad \bar{x}_i = \text{argmax}_{x \in Q}\{a_i^T x\}. \tag{23}$$

Then, it is clear that

$$\mu^* \le \mu_u \doteq \min_{1 \le i \le m}\{\mu_i\}. \tag{24}$$

Let $\bar{x} = \frac{1}{m}\sum_{k=1}^m \bar{x}_k$. Then

$$\mu(\bar{x}) = \min_{1 \le i \le m}\{a_i^T \bar{x}\} = \frac{1}{m}\min_{1 \le i \le m}\left\{\sum_{k=1}^m a_i^T \bar{x}_k\right\} \ge \frac{1}{m}\min_{1 \le i \le m}\left\{a_i^T \bar{x}_i\right\} = \frac{1}{m}\mu_u.$$

$$\mu_l \doteq \frac{1}{m}\mu_u \le \mu^* \le \mu_u. \tag{25}$$

5.2 Refinement of bounds

In order to approximately compute μ^* we would like to use a binary search procedure similar to that described in section 2.1. In the covering case, however, we need a further elaboration. Given any $\mu_u > 0$ define

$$C(\mu_u) \doteq \{y \in [0,1]^n : \exists x \in Q \text{ s.t. } x \ge \mu_u y\}, \tag{26}$$

and

$$\mu_u^* = \max_{y \in C(\mu_u)} \min_{1 \le i \le m}\{a_i^T y\}. \tag{27}$$

Then we have the following result.

Lemma 10 *For any upper bound $\mu_u \ge \mu^*$, we have that $\mu_u^* = \frac{\mu^*}{\mu_u}$.*

Proof: Let $y^* = \text{argmax}\{\mu(y) : y \in C(\mu_u)\}$. Since $y \in C(\mu_u)$, there exists $\hat{x} \in Q$ such that $\hat{x} \ge \mu_u y^*$. Thus,

$$\mu_u^* = \mu(y^*) \le \frac{\mu(\hat{x})}{\mu_u} \le \frac{\mu^*}{\mu_u}.$$

To prove the bound in the other direction, let $x^* = \text{argmax}_{x \in Q}\, \mu(x)$. Define

$$\hat{y}_j = \frac{\min\{x_j^*, \mu_u\}}{\mu_u}, \quad j = 1, \dots, n.$$

Then $\hat{y} \in C(\mu_u)$ and to complete the proof it will suffice to show that, for any $1 \leq i \leq m$, $\sum_j a_{ij}\hat{y}_j \geq \frac{\mu^*}{\mu_u}$. To see that this is the case, fix i, and note that if $x^*_h > \mu_u$ for some h with $a_{ih} > 0$, then by the assumption on A, $\sum_j a_{ij}\hat{y}_j \geq \hat{y}_h = 1 \geq \frac{\mu^*}{\mu_u}$ (since $\mu_u \geq \mu^*$). If on the other hand $x^*_j \leq \mu_u$ for all j with $a_{ij} > 0$, then

$$\sum_j a_{ij}\hat{y}_j = \frac{\sum_j a_{ij}x^*_j}{\mu_u} \geq \frac{\mu^*}{\mu_u},$$

as desired. ■

The rest of the algorithm mirrors that for the packing case, in that we will use a potential function method to implement each step of the binary search. However, unlike in the packing case the quadratic optimization problems solved in the course of approximately minimizing the potential function are defined over the sets $C(\mu_u)$, rather than over Q itself. At the end of this section we will comment on why our approach needs the sets $C(\mu_u)$.

Consider a typical step of the binary search. We are given a lower bound μ_l and an upper bound μ_u on μ^*. Setting $\delta = \frac{\mu_u - \mu_l}{3}$, the outcome of the binary search step is a $x \in Q$ with $\mu(x) \geq \mu^* - \delta$; if $\mu(x) \geq \mu_l + 2\delta$ then we reset $\mu_l \leftarrow \mu_l + \delta$, while in the other case we reset $\mu_u \leftarrow \mu_l + 2\delta$, thereby improving the gap between the two bounds by a factor of 2/3. In view of the above Lemma, we equivalently seek a $y \in C(\mu_u)$ with $\mu(y) \geq \mu^*_{P(\mu_u)} - \gamma$, where $\gamma = \frac{\delta}{\mu_u}$; and we will compare $\mu(y)$ with $\frac{\mu_l}{\mu_u} + 2\gamma$.

The computation of such a y is accomplished through the use of a potential function. For the covering case, we now use:

$$\Phi(x) = \frac{1}{\beta}\log\left(\sum_{i=1}^{m} e^{-\beta a_i^T x}\right)$$

for a suitably chosen $\beta > 0$. We have the following analogue of equation (5)

$$-\mu(x) \leq \Phi(x) \leq -\mu(x) + \frac{\ln m}{\beta}. \tag{28}$$

Thus, given $\gamma > 0$, in order to compute $y \in P(\mu_u)$ with $\mu(y) \geq \mu^*_u - \gamma$ we set $\beta = \frac{2\ln m}{\gamma}$ and compute $y \in C(\mu_u)$ with $\Phi(y) \leq \Phi^* + \gamma/2$, where $\Phi^* \doteq \min_{y \in P(\mu_u)} \Phi(y)$. For this purpose we can use Algorithm QP given in section 2, verbatim. To see that this is a valid approach, recall that the proofs in Section 2 and that of Lemma 5 are all valid for negative values of α.

Thus, neglecting polynomial factors in n and m, and polylog factors in γ, we will need $O^*(\gamma^{-1})$ iterations to compute y, and, in summary, we will compute an ϵ-optimal solution to the covering problem in $O^*(\epsilon^{-1})$ iterations.

Of course, these iterations are quadratic programs over sets $C(\mu_u)$, not the original constraint set Q. But note that if Q is block-angular, then so is $C(\mu_u)$. Also, $y \in C(\mu_u)$ if and only if there exists $s \geq 0$ such that $\mu_u y + s \in Q$. In other words, if we are given a linear inequality description of Q, then we obtain one for $C(\mu_u)$ with twice as many variables but with the same general structure.

Also, suppose that Q is given by a linear optimization oracle. Under appropriate assumptions, a convex quadratic program over $C(\mu_u)$ reduces to a polynomial number of linear programs over Q [24]. The key ingredient here is that that the separation problem over $C(\mu_u)$ can be solved by making a polynomial number of calls to the optimization oracle over Q. To see that this is the case, recall that we assumed $Q \subseteq \mathbf{R}^n_+$ and write

$$Q^{\leq}(\mu_u) \doteq \{y \geq 0 : \exists x \in Q \text{ s.t. } x \geq \mu_u y\}.$$

Then, for any $c \in R^n$, we have that

$$\max\Big\{\sum_j c_j y_j \,:\, y \in Q^{\leq}(\mu_u)\Big\} \;=\; \max\Big\{\sum_j c_j^{+} y_j \,:\, y \in Q^{\leq}(\mu_u)\Big\}, \tag{29}$$

where $r^+ \doteq \max\{r, 0\}$. Since the objective vector in the right-hand side of (29) is nonnegative, it follows that this term is equal to

$$(\mu_u)^{-1} \max\Big\{\sum_j c_j^{+} y_j \,:\, y \in Q\Big\}.$$

Thus, a linear optimization problem over $Q^{\leq}(\mu_u)$ reduces to an equivalent linear optimization problem over Q. Consequently, the separation problem over $Q^{\leq}(\mu_u)$ requires a polynomial number of oracle calls (to the optimization oracle over Q). Since

$$C(\mu_u) \;=\; Q^{\leq}(\mu_u) \cap \{y \in R^n \,:\, y_j \leq 1 \;\forall j\}\,,$$

we can conclude that the same for $C(\mu_u)$, as desired.

5.3 Why do covering problems require quadratic optimization over $C(\mu_u)$

In order to motivate our use of the sets $C(\mu_u)$, it is worth revisiting some of our techniques used for packing problems, in particular, the use of the potential function, which we adapted from the packing case in a straighforward manner in order to handle covering problems (essentially, by changing the sign of the exponent).

To understand the difference between packing and covering problems let us examine the role of the sets P in packing problems. In the context of a fractional packing problem $\textsc{Pack}(A, Q)$ with $A \in \{0,1\}^{m\times n}$, the critical observation is that $\lambda^* \leq \lambda_u$ implies that $x_j \leq \lambda_u$ for all $j = 1, \ldots, n$. In this setting the set $P = \lambda_u^{-1} Q \cap [0,1]^n$. Next, consider a generalized packing problem with a nonnegative matrix A. We show that by introducing new variables this problem can be reduced to a packing problem $\textsc{Pack}(A, Q)$ with $A \in \{0,1\}^{m\times n}$. The combined effect of adding variables and of scaling by λ_u^{-1} is the following: if $a_{ij} > 0$ then we generate a new variable of the form $\frac{a_{ij}}{\lambda_u} x_j$. The fact that $\lambda^* \leq \lambda_u$ implies that, without loss of generality, we can place an upper bound of 1 on this new variable; or equivalently, $\lambda^* \leq \lambda_u$ implies that the set $\bar{Q} = \{x \in Q \,:\, \frac{a_{ij}}{\lambda_u} x_j \leq 1 \;\forall\, a_{ij} > 0\}$ is nonempty, and $\lambda(x) \leq \lambda_u$ if and only if $x \in \bar{Q}$. For any $A \in \{0,1\}^{m\times n}$ and $x \in P$, we have $\sum_j a_{ij} x_j \leq K$ for all i, where K denotes the maximum number of nonzeros terms in any row of A. This bound is instrumental in the proof of Corollary 4 and Lemma 5, both of which are critical in the analysis of the algorithm. In particular, the iteration count depends on K, and not on the "width" [21] of the problem.

However, when dealing with a covering problem this approach fails. Given an upper bound μ_u on μ^*, and a coefficient $a_{ij} > 0$, we cannot assume that $\frac{\mu_u}{a_{ij}}$ is an upper bound on x_j. The reason for this is that there might be a different row i' with $0 < a_{i'j} < a_{ij}$. Instead, we can only assume that $x_j \leq \frac{\mu_u}{a_{kj}}$, where a_{kj} is the smallest positive coefficient in column j of A. However, when we use this bound the arguments in Corollary 4 and Lemma 5 do not go through – instead of getting a dependence on K we obtain a dependence on the maximum ratio between positive entries in any given column j of A. Finally, we may be unable to place a (simple) upper bound on some x_j, even when $A \in \{0,1\}^{m\times n}$ and there is a known upper bound $\mu_u \geq \mu^*$, because the structure of the set Q may simply prevent us from doing so – in other words, a set of the form $Q \cap \{x_j \leq \mu_u\}$ may be empty, in contrast with what happens in the packing case.

5.4 A special case

Consider the case where all positive entries of A have a common value. Without loss of generality, $A \in \{0,1\}^{m \times n}$. In addition suppose that Q is "downwards closed", that is to say for any $x \in Q$, if $0 \leq x' \leq x$ then $x' \in Q$ as well. The combination of the two assumptions overcomes the two difficulties listed above; and in this case the sets P have a particularly simple structure. Suppose we are given an upper bound μ_u on μ^*. Then $\mu_u P \subseteq Q$. In fact, optimizing a separable quadratic program over P is equivalent to optimizing a separable quadratic program over the set $\{x \in Q : 0 \leq x_j \leq \mu_u \ \forall j\}$.

6 Algorithms for mixed packing and covering

In this section consider the mixed packing-covering problem

$$\begin{array}{rcll} \lambda^* & \doteq & \min & \lambda(x) \doteq \max_{1 \leq i \leq m_p}\{p_i^T x\}, \\ & & \text{s.t.} & \min_{1 \leq i \leq m_c}\{c_i^T x\} \geq 1, \\ & & & 0 \leq x \in R^n. \end{array} \tag{30}$$

Here, the vectors p_i, $i = 1, \ldots, m_p$ and c_i, $i = 1, \ldots, m_c$, are assumed to be nonnegative vectors. We will present an algorithm that computes for any $\epsilon > 0$, an ϵ-approximation to λ^* in $O^*(\frac{1}{\epsilon})$ steps. As before, each step consists of solving a separable, convex quadratic program, but now in addition these quadratic programs are solved over sets of the form $l_j \leq x_j \leq u_j$, $1 \leq j \leq n$. Thus, each step can be solved in $O(n)$ time while using $O(n)$ space.

6.1 Polynomially separated upper and lower bounds

In this section, we compute upper and lower bounds on λ^*. The analysis in this section is identical to the analysis in [28].

Let $p = \frac{1}{m_p}\sum_{i=1}^{m_p} p_i$. For each $k = 1, \ldots, m_c$ let

$$\begin{aligned} \lambda_k &= \min\left\{p^T x : c_k^T x \geq 1, x \geq 0\right\}, \\ &= \min_{1 \leq j \leq n}\left\{\frac{p_j}{c_{kj}}\right\}. \end{aligned} \tag{31}$$

Let $x^{(k)}$ denote the vector that achieves the minimum in (31), i.e.

$$x_j^{(k)} = \begin{cases} \frac{1}{c_{kj}}, & j = \operatorname{argmin}_{1 \leq j' \leq n}\left\{\frac{p_{j'}}{c_{kj'}}\right\}, \\ 0, & \text{otherwise.} \end{cases}$$

Then, it is clear that

$$\lambda^* \geq \lambda_l = \max_{1 \leq k \leq m_c}\{\lambda_k\}.$$

Let $\bar{x} = \sum_{k=1}^{m_c} x^{(k)}$. Then

$$c_i^T \bar{x} = \sum_{k=1}^{m_c} c_i^T x^{(k)} \geq c_i^T x^{(i)} \geq 1,$$

i.e. $\bar{x}$ is feasible for (30), and

$$\max_{1\le i\le m_p}\{p_i^T\bar{x}\} \le \sum_{i=1}^{m_p} p_i^T\bar{x} = m_p p^T\bar{x} = m_p\sum_{k=1}^{m_c}\lambda_k = m_c m_p \lambda_l = \lambda_u$$

Thus, we have a lower bound λ_l and an upper bound λ_u such that $\lambda_u \le m_c m_p \lambda_l$.

As was the case with all other problems considered in this paper, the bounds will be refined using a binary search technique. In the general step of the binary search we have a lower bound $0 \le \lambda_l$ and an upper bound λ_u on λ^*. Let $\delta \doteq \frac{\lambda_u-\lambda_l}{2}$, and let $\gamma \doteq \frac{\delta}{\lambda_u}$. Thus, the relative gap between the bounds is 2γ. We achieve this by converting the problem into an equivalent pure covering problem and computing a good solution for the covering problem.

Since λ_u is an upper bound on λ^*, it follows that for $1 \le i \le m_p$, if $p_{ij} > 0$ we can assume that $x_j \le \frac{\lambda_u}{p_{ij}}$. For $1 \le i \le m_p$, write $S_i \doteq \{j : p_{ij} > 0\}$. We then have:

$$\frac{\lambda^*}{\lambda_u} \quad = \quad \begin{array}{rl} \min & \max_{1\le i\le m_p}\{\sum_{j\in S_i} y_{ij}\}, \\ \text{s.t.} & (x,y)\in T. \end{array} \tag{32}$$

where $T \subseteq R_+^n \times R_+^n$ is the set of pairs (x,y), such that

$$\begin{array}{rcll} \sum_j c_{ij}x_j & \ge & 1, & i=1,\ldots,m_c, \\ y_{ij} - \frac{p_{ij}}{\lambda_u}x_j & = & 0, & j\in S_i, \quad i=1,\ldots,m_p, \\ y_{ij} & \le & 1, & j\in S_i, \quad i=1,\ldots,m_p, \\ x & \ge & 0. & \end{array}$$

Note that $\frac{\lambda_l}{\lambda_u} \le \frac{\lambda^*}{\lambda_u} \le 1$, and $1 - \frac{\lambda_l}{\lambda_u} = 2\gamma$. Rather than work with T itself, we change variables once again, by defining variables $z_{ij} = 1 - y_{ij}$. As we show below, this reformulation allows us to refine the bounds on $\frac{\lambda^*}{\lambda_u}$. To this end, consider the following covering problem:

$$\mu^* \quad = \quad \begin{array}{lll} \max & \mu & \\ \text{s.t.} & \sum_{j\in S_i} z_{ij} \ge \mu(|S_i| - (1-\gamma)), & i=1,\ldots,m_p, \\ & \sum_j c_{ij}x_j \ge \mu, & i=1,\ldots,m_c, \\ & (x,z)\in Q, & \end{array} \tag{33}$$

where $Q \subseteq R_+^n \times R_+^n$ is the set of pairs (x,z), such that

$$\begin{array}{rcll} z_{ij} + \frac{p_{ij}}{\lambda_u}x_j & = & 1, & j\in S_i, \quad i=1,\ldots,m_p, \\ z_{ij} & \ge & 0, & j\in S_i, \quad i=1,\ldots,m_p, \\ x & \ge & 0. & \end{array}$$

Let S denotes the largest $|S_i|$ and let $(\bar{x},\bar{y})$ denote any $\frac{\gamma}{3S}$-optimal solution of (33). Consider the following two cases:

(a) $\mu(\bar{x},\bar{z}) < (1-\frac{\gamma}{3S})$. In this case we claim that that $1-\gamma$ is a lower bound on $\frac{\lambda^*}{\lambda_u}$. Otherwise, there exists $(\hat{x},\hat{y}) \in T$ with $\sum_{j\in S_i} y_{ij} \le 1-\gamma$ for each $1 \le i \le m_p$. Then it follows that $\mu^* \ge 1$, and a $\frac{\gamma}{3S}$-optimal solution to (33) must have $\mu \ge (1-\frac{\gamma}{3S})$, a contradiction.

(b) $\mu(\bar{x}, \bar{z}) \geq (1 - \frac{\gamma}{3S})$. Then $\sum_j c_{ij}\bar{x}_{ij} \geq 1 - \frac{\gamma}{3S}$, and

$$\sum_{j \in S_i} \bar{z}_{ij} \geq \left(1 - \frac{\gamma}{3S}\right)(|S_i| - (1-\gamma)) \geq |S_i| - (1 - 2\gamma/3), \quad i = 1, \ldots, m_p.$$

Define $\bar{y}_{ij} = 1 - \bar{z}_{ij}$, $j \in S_i$, $i = 1, \ldots, m_p$ and $(\tilde{x}, \tilde{y}) = \frac{1}{1-\frac{\gamma}{3S}}(\bar{x}, \bar{x})$. Then it follows that $\sum_j c_{ij}\tilde{x}_{ij} \geq 1$, for all $i = 1, \ldots, m_c$. Thus, $(\tilde{x}, \tilde{y}) \in T$. The objective

$$\sum_{j \in S_i} \tilde{y}_{ij} \leq \frac{1 - 2\gamma/3}{1 - \frac{\gamma}{3S}} \leq 1 - \frac{\gamma}{3}, \quad i = 1, \ldots, m_p.$$

Consequently, we jave improved upper bound on $\frac{\lambda^*}{\lambda_u}$.

In conclusion, in either case the gap is decreased by at least a factor of 5/6.

References

[1] R. Ahuja, T. L. Magnanti, and J. Orlin. *Networks Flows: Theory, Algorithms, and Practice*, Prentice Hall. 1993.

[2] B. Awerbuch and F. Leighton, A simple local-control approximation algorithm for multicommodity flow, *Proc. 34th Ann. Symp. on Foundations of Comp. Sci. (FOCS)*, 459–468, 1993.

[3] D. Bienstock. *Potential Function Methods for Approximately Solving Linear Programming Problems, Theory and Practice*, Kluwer, Boston. 2002. An early version downloadable from `www.core.ucl.ac.be`.

[4] D. Bienstock and O. Raskina. Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. *Math. Prog.* **91**:379–492, 2002.

[5] L.K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Disc. Math.*, **13**:505-520, 2000.

[6] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Res. Log. Quart.* **3**:149–154, 1956.

[7] L. Fratta, M. Gerla and L. Kleinrock. The flow deviation method: an approach to store-and-forward communication network design. *Networks* **3**:97-133, 1971.

[8] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proc. 39th Ann. Symp. on Foundations of Comp. Sci. (FOCS)*, 300-309, 1998.

[9] M.D. Grigoriadis and L.G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM J. Optim.*, **4**:86-107, 1994.

[10] M.D. Grigoriadis and L.G. Khachiyan. An exponential-function reduction method for block-angular convex programs. *Networks* **26**:59-68, 1995.

[11] M.D. Grigoriadis and L.G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}KNM)$ time. *Math. Prog.*, **75**:477-482, 1996.

[12] M@. Grotschel, L@. Lovasz and A@. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-VErlag. 1993.

[13] S.Kapoor and P.M.Vaidya. Fast algorithms for convex quadratic programming and multicommodity flows, *Proc. 18th Ann. ACM Symp. on Theory of Computing*, 147-159, 1986.

[14] 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. *Proc. 22nd Ann. ACM Symp. on Theory of Computing*, 310-321, 1990.

[15] P. Klein and N. Young. On the number of iterations for Dantzig-Wolfe optimization and packing-covering approximation algorithms, Proceedings of IPCO VII, 320-327, 1999.

[16] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems, *Proc. 23nd Ann. ACM Symp. on Theory of Computing*, 101-111, 1999.

[17] M. Minoux. A polynomial algorithm for minimum quadratic cost flows. *European J. Oper. Res.* **18**:377-387, 1984.

[18] A. Nemirovski. Prox-method with rate of convergence $O(\frac{1}{t})$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. Under review in *SIAM J. Optim.* 2003.

[19] Y. Nesterov. Smooth minimization of non-smooth functions. CORE Discussion Paper, CORE, UCL, Belgium. 2003.

[20] S. Plotkin and D. Karger. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 18-25, 1995.

[21] S. Plotkin, D.B. Shmoys and E. Tardos. Fast approximation algorithms for fractional packing and covering problems, *Math. Oper. Res.* **20**:495-504, 1995.

[22] T. Radzik. Fast deterministic approximation for the multicommodity flow problem. *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 1995.

[23] F. Shahrokhi and D.W. Matula. The maximum concurrent flow problem. *J. ACM* **37**:318-334, 1991.

[24] L. Tunçel, personal communication.

[25] P.M. Vaidya. Speeding up linear programming using fast matrix multiplication, *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science*, 332-337, 1989.

[26] P.M. Vaidya. A new algorithm for minimizing convex functions over convex sets, *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science*, 338-343, 1989.

[27] J. Villavicencio and M.D. Grigoriadis. Approximate Lagrangian decomposition using a modified Karmarkar logarithmic potential. In *Network Optimization* (Pardalos and Hager, eds.) Lecture Notes in Economics and Mathematical Systems **450** Springer-Verlag, Berlin, 471 – 485, 1995.

[28] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. of 42nd IEEE FOCS*, pages 538–546, 2001.

A Appendix

A.1 Improved Taylor expansion

The derivative of $\nabla\Phi(x) = A^T\pi(x)$, where

$$\pi_i(x) = \frac{e^{\alpha a_i^T x}}{\sum_{j=1}^m e^{\alpha a_i^T x}}, \quad i = 1, \ldots, m.$$

We want to show that $\|A^T(\pi(x) - \pi(y))\|_2 \leq K|\alpha|\|x - y\|_2$.

The construction of the bound will proceed in two steps.

(a) $\mathcal{L}_1$-bound on A^T: Since $\pi(x) \geq 0$ and $\sum_{i=1}^m \pi_i(x) = 1$ for all x, we endow the π-space with the $\mathcal{L}_1$-norm. Define $\|A\|_{1,2}$ as follows.

$$\begin{aligned}\|A\|_{1,2} &= \max\Big\{\|A^T\pi\|_2 : \|\pi\|_1 = 1\Big\}, \\ &= \max\left\{\|\sum_{i=1}^m \pi_i a_i\|_2 : \|\pi\|_1 = 1\right\}, \\ &= \max_{1\leq i\leq m}\{\|a_i\|_2\} = \sqrt{K}.\end{aligned}$$

where the first equality in the last step follows from the fact that $\|a\|_2$ is a convex function and achieves its maximum at the extreme points of the feasible set and the final bound follows from the fact that number of 1's in the vectors a_i is bounded above by K. Thus, we have that

$$\begin{aligned}\|A^T(\pi(x) - \pi(y))\|_2^2 &\leq \|A\|_{1,2}^2\|\pi(x) - \pi(y)\|^2, \\ &\leq K\|\pi(x) - \pi(y)\|_1^2. \end{aligned} \tag{34}$$

(b) Strong convexity of the entropy function: The next step is to show an upper bound of the form $\|\pi(x) - \pi(y)\|_1 \leq C\|x - y\|_2$ for an appropriate C. Since we need an upper bound on a norm, one way to achieve this is to bound it above by a strongly convex function.

Define $H(\pi) = \sum_{i=1}^m \pi_i \log(\pi_i)$ for π such that $\pi \geq 0$ and $\sum_{i=1}^m \pi_i = 1$. Then, $\nabla^2 H = \mathbf{diag}(1/\pi_i)$, and

$$y^T(\nabla^2 H)y = \sum_{i=1}^m \frac{y_i^2}{\pi_i}.$$

The RHS is a convex function and on minimizing this function over π in the simplex, we obtain that

$$y^T(\nabla^2 H)y \geq \Big(\sum_{i=1}^m |y_i|\Big)^2 \tag{35}$$

Thus, we have that

$$\begin{aligned}
(\nabla H(\pi(x) - \nabla H(\pi(y)))^T(\pi(x) - \pi(y)) &= (\nabla^2 H(\pi_\theta)(\pi(x) - \pi(y)))^T(\pi(x) - \pi(y)),\\
&\geq \|\pi(x) - \pi(y)\|_1^2,
\end{aligned}$$

where $\pi_\theta = \theta\pi(x) + (1-\theta)\pi(y)$, $\theta \in [0,1]$, the first equality follows from the mean value theorem, and the second inequality follows form (35). Substituting the value of $\nabla H(\pi(x))$ and $\nabla H(\pi(x))$, it follows that

$$\begin{aligned}
\|\pi(x) - \pi(y)\|_1^2 &\leq \sum_{i=1}^m (\log(\pi_i(x)) - \log(\pi_i(y)))(\pi_i(x) - \pi_i(y)),\\
&= \sum_{i=1}^m \Big(\alpha(a_i^T x) - \log\Big(\sum_j e^{\alpha a_j^T x}\Big)\Big)(\pi_i(x) - \pi_i(y))\\
&\quad - \sum_{i=1}^m \Big(\alpha(a_i^T y) + \log\Big(\sum_j e^{\alpha a_j^T y}\Big)\Big)(\pi_i(x) - \pi_i(y)),\\
&= \alpha \sum_{i=1}^m \Big(a_i^T(x-y)\Big)(\pi_i(x) - \pi_i(y))\\
&\quad - \log\big(\sum_j e^{\alpha a_j^T y}\big)\big) \sum_{i=1}^m (\pi_i(x) - \pi_i(y))\\
&\quad + \log\big(\sum_j e^{\alpha a_j^T y}\big)\big) \sum_{i=1}^m (\pi_i(x) - \pi_i(y)), \qquad (36)\\
&= \alpha \sum_{i=1}^m \Big(a_i^T(x-y)\Big)(\pi_i(x) - \pi_i(y)),\\
&= \alpha(x-y)^T(A^T(\pi(x) - \pi(y))),\\
&\leq |\alpha|\|x-y\|_2\|A^T(\pi(x) - \pi(y))\|_2, \qquad (37)
\end{aligned}$$

where (36) follows from the fact that $\pi(x)$, $\pi(y)$ are both elements of the simplex and (37) follows from the Cauchy-Schwartz inequality.

Combining (34) and (37) we get

$$\begin{aligned}
\|A^T(\pi(x) - \pi(y))\|_2 &\leq |\alpha|\|A\|_{1,2}^2\|x-y\|_2,\\
&\leq K|\alpha|\|x-y\|_2.
\end{aligned}$$

A.2 Proof of Theorem 7

We begin with the approximate version of the Taylor expansion (see Lemma 5).

Lemma 11 *At any iteration t of Algorithm* QP_σ*, we have that for any $x \in P$*

$$\hat{S}_t(x) - \hat{S}_t(\hat{z}^{(t)}) \geq \frac{2K|\alpha|}{(t+1)(t+2)}\Big\{\sum_{j=1}^n (x_j - \hat{z}_j^{(t)})^2 - \sum_{j=1}^n \Big(\sigma_j + \frac{1}{4}\sigma_j^2\Big)\Big\}.$$

Proof: Fix $x \in P$. Recall that the function $\hat{S}_t(x)$ is defined as follows.

$$\hat{S}_t(x) \quad = \quad \frac{2}{(t+1)(t+2)} \Big\{ 2K|\alpha| \sum_{j=1}^{n} \mathcal{L}_{\sigma_j, \hat{x}_j^{(0)}}(x_j) + \sum_{h=0}^{t} (h+1)[\Phi(\hat{x}^h) + \langle g_h, x - \hat{x}^h \rangle] \Big\}.$$

Define

$$\bar{S}_t(x) = \frac{2}{(t+1)(t+2)} \Big\{ K|\alpha| \sum_{j=1}^{n} (x_j - \hat{x}_j^{(0)})^2 + \sum_{h=0}^{t} (h+1)[\Phi(\hat{x}^{(h)}) + \langle \hat{g}^{(h)}, x - \hat{x}^{(h)} \rangle] \Big\}.$$

Then, by Lemma 6 (ii),

$$\hat{S}_t(x) - \hat{S}_t(\hat{z}^{(t)}) \geq \bar{S}_t(x) - \bar{S}_t(\hat{z}^{(t)}) - \frac{2K|\alpha|}{(t+1)(t+2)} \Big\{ \sum_{j=1}^{n} \frac{1}{4} \sigma_j^2 \Big\}. \tag{38}$$

Since $\bar{S}_t$ is a quadratic function, it follows that

$$\bar{S}_t(x) - \bar{S}_t(\hat{z}^{(t)}) \quad = \quad \frac{2K|\alpha|}{(t+1)(t+2)} \sum_{j=1}^{n} (x_j - \hat{z}_j^{(t)})^2 + \langle \nabla \bar{S}_t(\hat{z}^{(t)}), x - \hat{z}^{(t)} \rangle. \tag{39}$$

Consider the function $\hat{S}_t$ restricted to the one-dimensional segment between $\hat{z}^{(t)}$ and x. $\hat{S}_t$ is convex, piecewise-linear, and is minimized at $\hat{z}^{(t)}$ (by definition of $\hat{z}^{(t)}$). Hence, as we traverse the segment from $\hat{z}^{(t)}$ to x, the slope of the first piece of the piecewise-linear function must be nonnegative. In other words,

$$\frac{2}{(t+1)(t+2)} \Big\{ \sum_{j=1}^{n} [K|\alpha| \lambda_j + \sum_{h=0}^{t} (h+1) g_j^{(h)}](x_j - \hat{z}_j^{(t)}) \Big\} \geq 0, \tag{40}$$

where for $1 \leq j \leq n$,

$$\lambda_j = \begin{cases} \mathcal{L}^{+}_{\sigma_j, \hat{x}_j^{(0)}}(\hat{z}_j^{(t)}), & x_j \geq \hat{z}_j^{(t)}, \\ \mathcal{L}^{-}_{\sigma_j, \hat{x}_j^{(0)}}(\hat{z}_j^{(t)}), & \text{otherwise}, \end{cases}$$

and $g_j^{(h)}$ is the j-th coordinate of $g^{(h)}$, $j = 0, \ldots, t$. Since $P \subseteq [0,1]^n$, by Lemma 6 (iii) the second term in the right-hand side of (39) is at least $-\frac{2K|\alpha|}{(t+1)(t+2)} \sum_{j=1}^{n} \sigma_j$; consequently,

$$\bar{S}_t(x) - \bar{S}_t(\hat{z}^{(t)}) \geq \frac{2K|\alpha|}{(t+1)(t+2)} \Big\{ \sum_{j=1}^{n} (x_j - \hat{z}_j^{(t)})^2 - \sum_{j=1}^{n} \sigma_j \Big\}.$$

Together with equation (38) this implies the desired result. ■

Theorem 7 is established by induction on t. By definition, we have that

$$\begin{aligned} \hat{S}_0(\hat{z}^{(0)}) \quad &= \quad 2K|\alpha| \sum_{j=1}^{n} \mathcal{L}_{\sigma_j, \hat{x}_j^{(0)}}(\hat{z}_j^{(0)}) \; + \; \Phi(\hat{x}^0) + \langle \hat{g}^{(0)}, z^{(0)} - x^0 \rangle, \\ &\geq \quad K|\alpha| \sum_{j=1}^{n} \mathcal{L}_{\sigma_j, \hat{x}_j^0}(\hat{z}_j^{(0)}) \; + \; \Phi(\hat{x}^0) + \langle \hat{g}^{(0)}, z^{(0)} - x^0 \rangle, \end{aligned}$$

$$
\begin{aligned}
&\geq\; K|\alpha| \sum_{j=1}^{n} \mathcal{L}_{\sigma_j,\hat{x}_j^0}(\hat{y}_j^{(0)}) \;+\; \Phi(\hat{x}^0) + \langle \hat{g}^{(0)}, y^{(0)} - x^0 \rangle, \\
&\geq\; \frac{K|\alpha|}{2} \sum_{j=1}^{n} (y_j^{(0)} - x_j^0)^2 \;+\; \Phi(x^0) + \langle g^{(0)}, y^{(0)} - x^0 \rangle, \qquad (41)\\
&\geq\; \Phi(y^{(0)}), \qquad (42)
\end{aligned}
$$

where (41) follows from Lemma 6(ii) and (42) follows from the definition of $y^{(0)}$.

Next, the inductive step. Let $x \in P$. By Lemma 11, we have

$$
\begin{aligned}
\hat{S}_t(x) &\geq \hat{S}_t(\hat{z}^{(t)}) + \frac{2K|\alpha|}{(t+1)(t+2)} \Big\{ \sum_{j=1}^{n} (\hat{z}_j^{(t)} - \hat{x}_j^0)^2 - \sum_{j=1}^{n} \Big(\sigma_j + \frac{1}{4}\sigma_j^2 \Big) \Big\}, \\
&\geq \hat{S}_t(\hat{z}^{(t)}) + \frac{2K|\alpha|}{(t+1)(t+2)} \sum_{j=1}^{n} (\hat{z}_j^{(t)} - \hat{x}_j^0)^2 - \frac{5K|\alpha|}{2(t+1)^2} \Big(\sum_{j=1}^{n} \sigma_j \Big).
\end{aligned}
$$

Applying the induction hypothesis, and continuing as in the proof of Theorem 3, we obtain the following analog of the inequality following (11):

$$
\hat{S}_{t+1}(x) \geq \Phi(\hat{x}^{(t+1)}) + \min_{y \in P} \Big\{ \frac{K|\alpha|}{2} \sum_{j} \Big(y_j - \hat{x}_j^{(t+1)} \Big)^2 + \langle \hat{g}^{(t+1)}, y - \hat{x}^{(t+1)} \rangle \Big\} - \Big(\frac{5K|\alpha|}{2} \Big) \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big).
$$

Applying Lemma 6 again, we obtain

$$
\begin{aligned}
\hat{S}_{t+1}(x) \;\geq\;& \Phi(\hat{x}^{(t+1)}) + K|\alpha| \sum_{j} \mathcal{L}_{\sigma_j,\hat{x}_j^{(t+1)}} \Big(\hat{y}_j^{t+1} \Big) + \langle \hat{g}^{(t+1)}, \hat{y}^{t+1} - \hat{x}^{(t+1)} \rangle \\
& - \Big(\frac{5K|\alpha|}{2} \Big) \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big), \\
\geq\;& \Phi(\hat{x}^{(t+1)}) + \frac{K|\alpha|}{2} \sum_{j} \Big(\hat{y}_j^{t+1} - \hat{x}_j^{(t+1)} \Big)^2 + \langle \hat{g}^{(t+1)}, \hat{y}^{t+1} - \hat{x}^{(t+1)} \rangle \\
& - \Big(\frac{5K|\alpha|}{2} \Big) \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big), \\
\geq\;& \Phi(\hat{y}^{t+1}) - \Big(\frac{5K|\alpha|}{2} \Big) \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big),
\end{aligned}
$$

where the last inequality follows from Lemma 5. ■

A.3 Piecewise-linear min-cost flow problems

We are given an optimization problem of the form described in section 3.1,

$$
\begin{array}{ll}
\min & \sum_{k,e} \mathcal{L}_{k,e}(r_{k,e}) \\
\text{s. t.} & N r_k = \hat{d}_k, \quad 0 \leq r_k \leq \hat{u}_k, \qquad k = 1, \ldots, K,
\end{array} \qquad (43)
$$

where for every k and e, $\mathcal{L}_{k,e}$ is a continuous, convex, piecewise-linear function with breakpoints at the integers and with pieces of strictly increasing slope, $\hat{d}_k$ and $\hat{u}_k$ are integral, and $u_{k,e} \leq 2^q$ for an appropriate integer $q > 0$. For each k and each vertex i, denote by $\hat{d}_{k,i}$ the i^{th} component of $\hat{d}_k$.

We assume that for each k we have an integral flow that satisfies the constraints of (43). Thus, we can convert (43) into an equivalent *circulation form*. Then we will have a problem of the form:

$$\begin{array}{ll} \min & \sum_{k,e} \mathcal{L}_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = 0, \quad -a_k \le r_k \le b_k, \qquad k = 1, \ldots, K, \end{array} \tag{44}$$

where for every k and e, $0 \le a_{k,e}, b_{k,e} \le 2^q$ and integral, and again $\mathcal{L}_{k,e}$ is a convex, continuous piecewise-linear function with breakpoints at the integers (the $\mathcal{L}_{k,e}$ in (44) equal those in (43), shifted by appropriate amounts).

We solve (44) by solving a sequence of circulation problems – our approach is similar to [17]. For all integer $h \in [0, q]$, $k = 1, \ldots, K$ and $e \in E$, let $\mathcal{L}^{(h)}_{k,e}$ denote the continuous, convex, piecewise-linear function with breakpoints at the integer multiples of 2^h, and which agrees with $\mathcal{L}_{k,e}$ at each breakpoint.

Further, define $a^{(h)}_{k,e} = \lceil 2^{-h} a_{k,e} \rceil$ and $b^{(h)}_{k,e} = \lceil 2^{-h} b_{k,e} \rceil$. Then, the *level-h* problem is:

$$\begin{array}{ll} \min & \sum_{k,e} \mathcal{L}^{(h)}_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = 0, -b^{(h)}_k \le r_k \le a^{(h)}_k, \quad \forall k. \end{array} \tag{45}$$

Thus, the level-0 problem is (44). We solve it by first solving the level-q problem, then the level-$(q-1)$ problem, and so on. Note that for $0 \le h \le q$, the function $\mathcal{L}^{(h)}_{k,e}$ has 2^{q-h} breakpoints in the range of the level-h problem. Hence, the level-h problem can be seen as an ordinary (e.g., linear) minimum-cost circulation problem, on the graph $\hat{G}^{(h)}$ obtained from the original graph G by replacing each edge e with 2^{q-h} parallel arcs, each with capacity 2^h and appropriate cost. (To avoid confusion, we use the term arc, rather than edge, which we reserve for G). We stress that our algorithm will *only implicitly* work with $\hat{G}^h$.

Inductively, suppose we have solved the level-h problem. We can assume, without loss of generality, that all the entries of optimal circulation $r^{(h)}_k$ are integer multiple of 2^h. Let π^h_k denote the node potentials (see [17] or [1] for details). Our task is to refine $r^{(h)}_k$ into an optimal circulation for the level-$(h-1)$ problem.

Note that by definition, for any k and e,

(a) The functions $\mathcal{L}^{(h)}_{k,e}$ and $\mathcal{L}^{(h-1)}_{k,e}$ agree at integer multiples of 2^{h-1}.

(b) Let $q \in \mathbf{Z}_+$. Then the slope of $\mathcal{L}^{(h-1)}_{k,e}$ is less (resp. more) than the slope of $\mathcal{L}^{(h)}_{k,e}$ in the interval $\left[2^h q, 2^h q + 2^{h-1}\right)$ (resp. in the interval $\left[2^h q + 2^{h-1},\, 2^h(q+1)\right)$).

(c) Either $a^{(h-1)}_{k,e} = a^{(h)}_{k,e}$ or $a^{(h-1)}_{k,e} = a^{(h)}_{k,e} - 2^{h-1}$, and similarly with the pair $(b^{(h-1)}_{k,e}, b^h_{k,e})$.

Thus, it is easy to see that $r^{(h)}_k$, together with the potentials $\pi^{(h)}_k$, *nearly* satisfies the optimality conditions for the level-$(h-1)$ problem. More precisely, suppose we convert $r^h_{k,e}$ into a circulation on the graph $\hat{G}^{(h-1)}$ by following the following "greedy" rule: for any k and e, we "fill" the parallel arcs corresponding to k, e in increasing order of cost (and thus, at most one arc will have flows strictly between bounds). We may need an additional, "overflow" arc, also of capacity 2^{h-1} in the case that $r^{(h)}_{k,e} = a^{(h)}_{k,e} > a^{(h-1)}_{k,e}$ and similarly for $b^{(h)}_{k,e}$.

Denote by $\hat{r}^{(h)}_{k,e}$ the resulting circulation in $\hat{G}^{h-1}$. Then by properties (a)-(c) above, it follows that at most *one* of the parallel arcs corresponding to a pair (k, e) either fails to satisfy the optimality conditions with respect to the potentials $\pi^{(h)}_k$ or is an overflow arc. Consequently, we can obtain an optimal circulation in $\hat{G}^{(h-1)}$ in at most $O(|E|)$ flow pushes (each pushing $2^{(h-1)}$ units of flow) or computation of node potentials; and each such step requires the solution of a shortest path problem. It is clear that (again because of (a)-(c) above) all of this can be done without explicitly considering the graph $\hat{G}^{(h-1)}$: instead, we always keep a single flow value for commodity k on any edge e, which is always an integral multiple of $2^{(h-1)}$ – if we wish to use one of the parallel arcs corresponding to k, e in a push (or when searching for an augmenting path), then it takes $O(1)$ time to determine *which* of the arcs we will use. This completes the description of the inductive step.

In summary, we have:

Lemma 12 *Problem (43) can be solved by performing $O(\sum_k \sum_e \log u_{k,e})$ shortest path computations.* $\blacksquare$