CORC Report TR-2003-03
Concurrent Flows in $O^*(\frac{1}{\epsilon})$ iterations

Daniel Bienstock (dano@columbia.edu) and Garud Iyengar (garud@columbia.edu)
Columbia University
New York, NY 10027

August 15, 2003 – Version 2004-04-08

Abstract

We adapt a method proposed by Nesterov [N03] to obtain an algorithm that computes ϵ-optimal solutions to packing problems by solving $O^*(\epsilon^{-1}\sqrt{Kn})$ separable convex quadratic programs, where K is the maximum number of nonzeros per row and n is the number of variables. We also show one can approximate the solution of the quadratic program to any degree of accuracy by the solution of an appropriately defined piecewise-linear program. For the special case of the maximum concurrent flow problem with rational capacities and demands we obtain an algorithm that computes an ϵ-optimal flow by solving $O^*(\epsilon^{-1}K^{3/2}\mathcal{M}\sqrt{\mathcal{N}}(\log\frac{1}{\epsilon}+L_U+L_D))$ shortest path problems, where K is the number of commodities, $\mathcal{M}$ is the number of arcs, $\mathcal{N}$ is the number of nodes, and L_U and L_D are, respectively, the number of bits needed to store the capacities and demands. In contrast, previous algorithms required $\Omega(\epsilon^{-2})$ iterations.

1 Introduction

Let $A = [a_1, \ldots, a_m^T]^T$ be an $m \times n$ 0-1 matrix. Let K denote the maximum number of non-zeros in any row of A. Let $Q \subseteq \Re^n_+$ be a closed convex set. We are interested in approximately solving the min-max problem

$$\lambda^*_{A,Q} = \min\left\{\lambda(x) \doteq \max_{1\le i\le m}\{a_i^T x\} \,:\, x \in Q\right\} \tag{1}$$

We show how to employ a technique due to Nesterov [N03] to obtain, for any given $\epsilon \in (0,1)$, a point $\hat{x} \in Q$ with $\lambda(\hat{x}) \le (1+\epsilon)\lambda^*_{A,Q}$, by solving at most

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

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 $\lambda^U > 0$.

The *maximum concurrent flow problem* can be stated in the form (1). Here we are given a graph G with $\mathcal{N}$ nodes and $\mathcal{M}$ edges, where each edge e has a positive capacity u_e, and a list of K commodities that need to be routed. The objective of the problem is to find a routing that minimizes the maximum *load* on any edge. Let $f_{k,e}$ denote the amount of flow of commodity k

on edge e. Then the load on edge e is given by $(\sum_k f_{k,e})/u_e$. We show that we can compute an ϵ-optimal flow by solving

$$O^*\left(\epsilon^{-1}K^{3/2}\mathcal{M}\sqrt{\mathcal{N}}\left(\log\frac{1}{\epsilon}+L_U+L_D\right)\right)$$

shortest path problems, where L_U and L_D denote, respectively, the number of bits needed to store the capacities and demands. In contrast, previous algorithms required $\Omega(\epsilon^{-2})$ iterations.

Shahrokhi and Matula [SM91] presented an algorithm for the case of uniform capacities (i.e., equal capacity for all edges). Their method seeks to approximately minimize an exponential potential function of the form

$$\sum_{ij} e^{\alpha\left(\sum_k g_e^k\right)}.$$

It is shown in [SM91] that, given $\epsilon \in (0,1)$, one can choose α so that (approximately) minimizing the potential function will yield a flow whose maximum load is at most $1+\epsilon$ times the optimum (i.e., an ϵ-optimal flow). The algorithm given in [SM91] is, roughly, a first-order procedure to minimize the potential function, and the number of iterations required to compute an ϵ-optimal flow is most $O(\epsilon^{-7})$ times a polynomial in the number of nodes and edges. Each iteration consists of a shortest-path computation, which is used to partially reroute a commodity.

[SM91] spurred a great deal of research, which generalized the techniques to broader packing and covering problems, gradually reduced the dependence of the iteration count on ϵ to finally obtain ϵ^{-2}, and simplified the overall approach. See [KPST90, LMPSTT91, GK94, GK95, PST91, R95, GaKo98, F00] for details. All of these algorithms rely, sometimes implicitly, on the exponential potential function, and can be viewed as first-order methods. [VG97] uses a logarithmic potential function. [BR02] shows that the flow deviation algorithm in [FGK71] yields an $O(\epsilon^{-2})$ algorithm; this time using a rational barrier function. For an overview of potential function approaches, see [B02].

Many of the methods in the prior literature can be applied to general *packing problems*. These are of the form:

$$\lambda^*_{A,Q} \;=\; \min_{x\in Q}\max_{1\le i\le m}\{a_i^T x\},$$

where as before $Q \subseteq \Re^n_+$ is a closed convex set, but now A is an $m\times n$ matrix with nonnegative entries. The packing problem can be reduced to one of the form (1), as follows. Let N be the number of nonzeros in A. For each entry $a_{ij} > 0$ we introduce a new variable y_{ij} and a new constraint:

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

Let

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

It follows that

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

Furthermore, P is closed convex, and because of (3), a convex separable quadratic program over P reduces to one over Q. Hence, for any $\epsilon \in (0,1)$ one can compute a point $\hat{x} \in Q$ with $\lambda(\hat{x}) \le (1+\epsilon)\lambda^*_{A,Q}$, in at most $O(\epsilon^{-1}\sqrt{N\log m})$ iterations.

A common feature to all of the prior algorithms is that they can be viewed, as (sometimes implicit) *Frank-Wolfe* [FW56] algorithms, in that they iterate by solving linear optimization problems over Q

(a shortest path problem or a min-cost flow problem in the case of multicommodity flow problems), and take convex combinations of iterates. [KY98] proved an $O(\epsilon^{-2})$ *lower* bound for Frank-Wolfe algorithms for problem (1), under appropriate conditions. The algorithms described in this paper bypass the requirements needed for the analysis in [KY98] 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. This is the particular detail that renders the bound in [KY98] invalid.
While some of the key ideas in this paper are derived from those in [N03], the paper is self-contained. 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.

2 Outer loop: binary search

In this section we abbreviate $\lambda^*_{A,Q}$ as λ^*. First, one can obtain, in polynomial time, an upper bound λ^U and a lower bound λ^L for λ^* that differ by at most a factor of $O(\min\{m, K\})$ (see [B02, GK94] for details). Next, the bounds are refined using a binary search procedure introduced in [GK94] (see [B02] for an alternate derivation). Here, **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 (λ^L, λ^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 $(\lambda^U - \lambda^L) \leq \epsilon\lambda^L$ **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 $\hat{x}$

Thus, our task is to supply the appropriate algorithm **ABSOLUTE**. In Section 3 we show:
Theorem: *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 2.1 *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*.*

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 of iterations $H = O(\log(\epsilon/mK))$ 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. ■

3 Inner loop: ABSOLUTE$(A, Q, \lambda^U, \delta)$ algorithm

In this section we describe the **ABSOLUTE**$(A, Q, \lambda^U, \delta)$ algorithm. We construct this algorithm using techniques from [N03].
Define $\bar{Q} \in \Re^{2n}$ as follows.

$$\bar{Q} = \left\{ (x, y) \in \Re^{2n} \,:\, x \in Q,\ \ y_j = \frac{x_j}{\lambda^U} \ \text{and}\ \ y_j \leq 1 \ (1 \leq j \leq n) \right\}.$$

Let P denote the *projection* of $\bar{Q}$ to the space of the y variables, that is

$$P = \{ y \in \Re^n \,:\, (x, y) \in \bar{Q} \text{ for some } x \}.$$

If Q is a polyhedron then so are $\bar{Q}$ and P. Moreover, $P \subseteq [0, 1]^n$. Let (with a slight abuse of notation)

$$\lambda^*_P = \min\{\lambda(y) = \max_{1 \leq i \leq m} \{a_i^T y\} \,:\, y \in P\}.$$

Then it follows that

$$\lambda^*_P = \frac{\lambda^*}{\lambda^U} \leq 1.$$

In this section we describe an algorithm that, for any $\gamma \in (0, 1)$, computes $\hat{y} \in P$, with $\lambda(\hat{y}) \leq \lambda^*(P) + \gamma$, 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.

3.1 Potential reduction algorithm

For the purposes of this section, we assume that we are given a 0-1, $m \times n$ matrix A with at most K nonzeros per row, and a (nonempty) closed convex set $P \subseteq [0, 1]^n$, and a constant $\gamma \in (0, 1)$. Furthermore, it is known that $\lambda^*_P \leq 1$.
Define the **potential function** ([GK94], [PST91]) $\Phi(x)$ as follows:

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

where $\alpha = \frac{2 \ln m}{\gamma}$. Let $\Phi^* = \min\{\Phi(x) \,:\, x \in P\}$. It is easy to show that for all $x \in P$

$$\lambda(x) \leq \Phi(x) \leq \lambda(x) + \frac{\ln m}{\alpha}. \tag{4}$$

(See [GK94] for details). Consequently, we only need to compute $x \in P$ with $\Phi(x) \leq \Phi^* + \gamma/2$.
Notation: In what follows, to avoid confusion, we will occasionally use the $\langle , \rangle$ notation to indicate inner product.

ALGORITHM N

Input: $P \subseteq [0,1]^n$, A, $\alpha = \frac{\ln m}{\gamma}$, $L = \gamma^{-1}\sqrt{8Knln(m)}$ **Output**: $\hat{y} \in P$ such that $\lambda(\hat{y}) \le \lambda_P^* + \gamma$ **choose** $x^0 \in P$. **set** $t \leftarrow 0$. **while** $(t \le L)$ **do** **set** $g_t \;=\; \nabla(\Phi(x^t))$. **set** $y^t \leftarrow \text{argmin}\left\{ \frac{K\alpha}{2}\sum_{j=1}^n (x_j - x_j^t)^2 \;+\; \langle g_t, x - x^t\rangle \;:\; x \in P\right\}$ **set** Let $z^t \leftarrow \text{argmin}\{S_t(x) \;:\; x \in P\}$ where $$S_t(x) \;=\; \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^n (x_j - x_j^0)^2 \;+\; \frac{2}{(t+1)(t+2)}\sum_{h=0}^t (h+1)\left[\Phi(x^h) + \left\langle g_h, x - x^h\right\rangle\right].$$ **set** $x^{t+1} \leftarrow \frac{2}{t+3}z^t \;+\; \frac{t+1}{t+3}y^t, \quad t \leftarrow t+1.$ **return** y^L

The following result is established in Section 3.2.

Theorem 3.1 *For any $t \ge 0$, $\Phi(y^t) \;\le\; S_t(z^t)$.*

Theorem 3.1 implies the following corollary.

Corollary 3.2 $\Phi(y^L) \;\le\; \Phi^* \;+\; \gamma/2$.

Proof: Fix $t \ge 0$. Let $x^* = \text{argmin}\{\Phi(x) : x \in P\}$. Theorem 3.1 implies that

$$\Phi(y^t) \le S_t(z^t) \le S_t(x^*) = \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^n (x_j^* - x_j^0)^2 + \frac{2}{(t+1)(t+2)}\sum_{h=0}^t (h+1)\left[\Phi(x^h) + \left\langle g_h, x^* - x^h\right\rangle\right].$$

Since Φ is convex, it follows that $\Phi(x^h) + \left\langle g_h, x^* - x^h\right\rangle \;\le \Phi^*$, $\forall 0 \le h \le t$. Thus, we have that

$$\begin{aligned}\Phi(y^t) \;&\le\; \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^n (x_j^* - x_j^0)^2 \;+\; \Phi^*\left(\frac{2}{(t+1)(t+2)}\sum_{h=0}^t (h+1)\right), \\ &\le\; \frac{2Kn\alpha}{(t+1)(t+2)} \;+\; \Phi^* \;=\; \frac{4Kn\ln m}{\gamma(t+1)(t+2)} \;+\; \Phi^*.\end{aligned}$$

Consequently, $\Phi(y^t) - \Phi^* \;\le\; \gamma/2$ for $t \;\ge\; L = \frac{\sqrt{8Kn\ln m}}{\gamma}$. ■

3.2 Proof of Theorem 3.1

The following lemma follows from considering the second-order Taylor expansion of Φ restricted to the line-segment from x to y.

Lemma 3.3 *For $x, y \in P$, $\Phi(y) \;\le\; \Phi(x) \;+\; [\nabla(\Phi(x))]^T(y-x) \;+\; \frac{K\alpha}{2}\sum_j (y_j - x_j)^2$.*

The rest of the proof of Theorem 3.1 closely mirrors the development in Section 3 of [N03]. The proof is by induction on t. For $t = 0$, note that

$$\begin{aligned} S_0(z^0) &= K\alpha \sum_{j=1}^{n} (z_j^0 - x_j^0)^2 \; + \; \Phi(x^0) + \left\langle g_0, z^0 - x^0 \right\rangle \\ &\geq \frac{K\alpha}{2} \sum_{j=1}^{n} (z_j^0 - x_j^0)^2 \; + \; \Phi(x^0) + \left\langle g_0, z^0 - x^0 \right\rangle \\ &\geq \frac{K\alpha}{2} \sum_{j=1}^{n} (y_j^0 - x_j^0)^2 \; + \; \Phi(x^0) + \left\langle g_0, y^0 - x^0 \right\rangle \; \geq \; \Phi(y^0). \end{aligned} \tag{5}$$

The first inequality in (5) follows by definition of y^0, and the second by Lemma 3.3.
Suppose now that we have proved the result for t, and we wish to prove it for $t+1$. Note that

$$S_{t+1}(x) \; = \; \Big(\frac{t+1}{t+3}\Big) S_t(x) \; + \; \Big(\frac{2}{t+3}\Big) \left[\Phi(x^{t+1}) + \left\langle g_{t+1}, x - x^{t+1} \right\rangle\right].$$

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

$$S_{t+1}(x) \geq \Big(\frac{t+1}{t+3}\Big) S_t(z_t) \; + \; \frac{2K\alpha}{(t+2)(t+3)} \sum_j (x_j - z_j^t)^2 \; + \; \frac{2}{t+3} \left[\Phi(x^{t+1}) + \left\langle g_{t+1}, x - x^{t+1} \right\rangle\right]. \tag{6}$$

By induction, $S_t(z^t) \geq \Phi(y^t) \geq \Phi(x^{t+1}) + \langle g_{t+1}, y^t - x^{t+1} \rangle$ (by convexity of Φ). Substituting in (6),

$$\begin{aligned} S_{t+1}(x) \; &\geq \; \Phi(x^{t+1}) \; + \; \frac{2K\alpha}{(t+2)(t+3)} \sum_j (x_j - z_j^t)^2 \; + \; \left\langle g_{t+1}, \frac{2}{t+3} x + \frac{t+1}{t+3} y^t - x^{t+1} \right\rangle \\ &\geq \; \Phi(x^{t+1}) \; + \; \frac{2K\alpha}{(t+3)^2} \sum_j (x_j - z_j^t)^2 \; + \; \left\langle g_{t+1}, \Big(\frac{2}{t+3}\Big) x + \Big(\frac{t+1}{t+3}\Big) y^t - x^{t+1} \right\rangle \\ &= \; \Phi(x^{t+1}) + \frac{K\alpha}{2} \sum_j \left(\Big(\frac{2}{t+3}\Big) x_j + \Big(\frac{t+1}{t+3}\Big) y_j^t - x_j^{t+1} \right)^2 \\ &\quad + \left\langle g_{t+1}, \Big(\frac{2}{t+3}\Big) x + \Big(\frac{t+1}{t+3}\Big) y^t - x^{t+1} \right\rangle, \end{aligned} \tag{7}$$

where (7) 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 (7) is lower bounded by:

$$\begin{aligned} &\Phi(x^{t+1}) \; + \; \min_{y \in P} \left\{ \frac{K\alpha}{2} \sum_j \left(y_j - x_j^{t+1}\right)^2 \; + \; \left\langle g_{t+1}, y - x^{t+1} \right\rangle \right\} \quad = \\ &\Phi(x^{t+1}) \; + \; \frac{K\alpha}{2} \sum_j \left(y_j^{t+1} - x_j^{t+1}\right)^2 \; + \; \left\langle g_{t+1}, y^{t+1} - x^{t+1} \right\rangle, \end{aligned} \tag{8}$$

by definition of y^{t+1}. By Lemma 3.3, the quantity in (8) is at least $\Phi(y^{t+1})$, as desired. This concludes the proof.

4 Piecewise-linear approximations

Algorithm N requires that we solve a separable quadratic programs over P. In this section we describe a general method for using piecewise-linear functions to approximate separable convex quadratics with arbitrarily small error. This method is derived from one given in Minoux [M84], Let $0 < \sigma$ and $w \in R$. Define a continuous piecewise-linear approximation $\mathcal{L}_{\sigma,w}(v)$ to the function $\frac{1}{2}(v-w)^2$ (valid for $v \geq 0$) as follows.

$$\mathcal{L}_{\sigma,w}(v) \doteq \frac{1}{2}q^2\sigma^2 + \frac{w^2}{2} - wv + \left(q + \frac{1}{2}\right)\sigma(v - q\sigma), \quad \forall v \in [q\sigma, (q+1)\sigma), \; q \in Z_+.$$

Note that for $v = q\sigma$, $q \in Z$, the derivative $\mathcal{L}'_{\sigma,w}(v)$ is not defined. Instead, we can define the *left-derivative* $\mathcal{L}^-_{\sigma,w}(v) = \left(q - \frac{1}{2}\right)\sigma$ and the *right-derivative* $\mathcal{L}^+_{\sigma,w}(v) = \left(q + \frac{1}{2}\right)\sigma$. On the other hand, $v \in (q\sigma, (q+1)\sigma)$, $q \in Z$, the derivative of $\mathcal{L}_{\sigma,w}(v)$ equals $\left(q + \frac{1}{2}\right)\sigma$, and for convenience we say this is the common value of the left- and right-derivatives. The following properties are easy to obtain.

Lemma 4.1 *For any $\sigma > 0$ and w,*

(i) For any integer $q \geq 0$, $\mathcal{L}_{\sigma,w}(q\sigma) = \frac{1}{2}(q\sigma - w)^2$,

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

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

4.1 Approximate Algorithm N

The following approximate version of Algorithm N uses a separate linear approximation for each quadratic term and possibly different $\sigma_j \in (0,1)$ for each variable x_j.

ALGORITHM N_σ

Input: $P \subseteq [0,1]^n$, A, $\alpha = \frac{\ln m}{\gamma}$, $L = \gamma^{-1}\sqrt{10Knln(m)}$, $\sigma_j \leq 2^{-p}$, $p > \frac{3}{2}\ln(Kn\ln(m)) + 3\ln(\frac{1}{\gamma})$
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**

set $g_t = \nabla(\Phi(\hat{x}^t))$.
set $\hat{y}^t \leftarrow \text{argmin}\left\{ K\alpha \sum_{j=1}^n \mathcal{L}_{\sigma_j, \hat{x}^t_j}(x_j) + \langle g_t, x - \hat{x}^t \rangle \, : \, x \in P \right\}$
set $\hat{z}^t \leftarrow \text{argmin}\{\hat{S}_t(x) \, : x \in P\}$ where

$$\hat{S}_t(x) = \frac{4K\alpha}{(t+1)(t+2)} \sum_{j=1}^n \mathcal{L}_{\sigma_j, x^0_j}(x_j) + \frac{2}{(t+1)(t+2)} \sum_{h=0}^t (h+1)\left[\Phi(\hat{x}^h) + \left\langle g_h, x - \hat{x}^h \right\rangle\right].$$

set $\hat{x}^{t+1} \leftarrow \frac{2}{t+3}\hat{z}^t + \frac{t+1}{t+3}\hat{y}^t$, $t \leftarrow t+1$.

return y^L

Algorithm N_σ replaces the quadratic objectives in Algorithm N with appropriate piecewise-linear approximations $\mathcal{L}_{\sigma_j,\hat{x}^t_j}$. In view of Lemma 4.1, we would expect that Algorithm N_σ successfully emulates Algorithm N if the σ_j are small enough. In the Appendix we provide a proof of the following fact:

Theorem 4.2 *For any $t \geq 0$, $\Phi(\hat{y}^t) \ \leq \ \hat{S}_t(\hat{z}^t) \, + \, 3K\alpha\Big(\sum_{h=1}^t \frac{1}{h^2} + t\Big)\big(\sum_k \sigma_j\big)$.*

Here we use this result to prove the correctness of Algorithm N_σ.

Corollary 4.3 *Algorithm N_σ $\Phi(y^L) \ \leq \ \Phi^* \, + \, \gamma/2$.*

Proof: Suppose $\sigma_j \leq 2^{-p}$. Using Theorem 4.2 and Lemma 4.1(ii), we obtain

$$\begin{aligned}
\Phi(\hat{y}^t) \quad &\leq \quad \Phi^* + \frac{2Kn\alpha}{(t+1)(t+2)} + 3K\alpha\left(\sum_{h=1}^t \frac{1}{h^2} + t\right) n2^{-p}, && (9)\\
&< \quad K\alpha n \left(2 + 3(2+t)2^{-p}\right). && (10)
\end{aligned}$$

Suppose $t \geq 2$ and choose $p \geq 3\ln t$. Then $\Phi(\hat{y}^t) - \Phi^* \leq \frac{5K\alpha n}{t^2}$. A simple calculation now establishes the result. ■

5 Concurrent flows with rational capacities and demands

In this section we focus on the maximum concurrent flow with rational capacities and show that the piecewise linear approximation introduced in Section 4 can be solved efficiently for this special case.

Suppose we have a network G with $\mathcal{N}$ nodes, $\mathcal{M}$ 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. Scaling capacities and demands by a common positive constant does not change the value of the problem, and therefore we can 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 $\mathcal{M}$-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 $F = \{f : Nf_k = d_k, f_k \geq 0, k = 1, \ldots, K\}$ 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. The constraints of the problem become

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

The problem is now in the canonical form described in Section 1, with $m = \mathcal{M}$ and $n = K\mathcal{M}$.

Step 2: After i iterations of **BINARY SEARCH** introduced in Section 2, we have lower and upper bounds λ^L and λ^U to λ^*, with

$$\frac{\lambda^U - \lambda^L}{\lambda^L} \quad \leq \quad \left(\frac{2}{3}\right)^i O(\min\{m, k\}). \tag{11}$$

Then, writing $\delta = \frac{1}{3}(\lambda^U - \lambda^L)$, we seek a vector g with $\max_e \sum_k g_{k,e} \leq \lambda^* + \delta$ – having computed this vector we either reset $\lambda^L \leftarrow \lambda^L + \delta$ or $\lambda^U \leftarrow \lambda^U - \delta$.

Step 3: Procedure **ABSOLUTE** computes the vector g needed in Step 2 by solving the scaled optimization problem

$$\begin{array}{rl} \min & \lambda, \\ \text{s.t.} & \sum_{k=1}^K x_{k,e} \leq \lambda, \quad e = 1, \ldots, \mathcal{M}, \\ & x \in P(\lambda^U) \doteq \{z : \exists g \in Q \text{ with } z = g/\lambda^U,\ 0 \leq z \leq 1\}. \end{array}$$

The value of this problem is $\lambda^*/\lambda^U \leq 1$, and, we seek a vector x feasible for this problem, and such that $\max_e \sum_k x_{k,e} \leq \lambda^*/\lambda^U + \gamma$, where $\gamma = \delta/\lambda^U$.

In Section 3.1 we show that in order to achieve the goal of procedure **ABSOLUTE** it is sufficient to produce a vector x such that $\Phi(x) \leq \Phi^* + \gamma/2$. Corollary 4.3 in Section 4 establishes that the output $\hat{y}^t$ produced by Algorithm N_σ satisfies this condition provided:

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

(b) $p > \frac{3}{2}\ln(Kn\ln(m)) + 3\ln(\frac{1}{\gamma}) \doteq \bar{p}(n, m, K, \gamma)$, and

(c) $t \geq \frac{\sqrt{10Kn\ln m}}{\gamma}$.

Note that in terms of the initial flow variables $f_{k,e}$, we have $x_{k,e} = \frac{1}{\lambda^U u_{k,e}} f_{k,e}$.
To adjust this framework to the concurrent flow problem, we set

$$\sigma_{k,e} \quad = \quad \frac{2^{-p}}{u_e}, \forall k,\, e, \ldots, \mathcal{M}, \tag{12}$$

where $p = \bar{p} + \lceil \log D \rceil$ and D is the sum of the demands. This satisfies requirement (a) of Step 3. Further, we modify **BINARY SEARCH**: every time a new upper bound λ^U is computed, we *relax* it, by replacing it with $\hat{\lambda}^U \geq \lambda^U$, chosen so that $\frac{2^p}{\hat{\lambda}^U} = \left\lfloor \frac{2^p}{\lambda^U} \right\rfloor$. Note that $\lambda^U \leq D$, and so $\hat{\lambda}^U \leq \frac{\lambda^U}{1 - \lambda^U 2^{-p}} \leq \lambda^U(1 + O(2^{-\bar{p}})$. Since $\bar{p} = \frac{3}{2}\ln(Kn\ln(m)) + 3\ln(\frac{1}{\gamma})$, where $\gamma = \delta/\lambda^U$, we have that

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

Thus, up to constants the the complexity bound in Corollary 2.1 remains unchanged. For simplicity, in what follows we will use the notation λ^U to refer to the relaxed upper bound.

5.1 Solving the piecewise-linear problems

In this section we show how to efficiently solve the piecewise-linear problems encountered in algorithm N_σ, using the fact that $\sigma_{k,e}$ is defined as in (12) and that, at any iteration, $\frac{1}{\lambda^U}$ is an integer multiple of 2^{-p} (which is what our modification to **BINARY SEARCH** achieves).
The generic piecewise-linear problem that we need to solve is of the form

$$\begin{array}{ll} \min & \sum_{k,e} \bar{\mathcal{L}}_{k,e}(x_{k,e}) \\ \text{s. t.} & x \in P(\lambda^U), \end{array} \tag{13}$$

where $b \in \Re^{K\mathcal{M}}$ is a fixed vector, and for any k and e, $\bar{L}_{k,e}(\cdot)$ is a continuous piecewise-linear function with breakpoints at the integer multiples of $\frac{2^{-p}}{u_{k,e}}$ and with pieces of strictly increasing slope. Suppose we change variables, by setting, for every k and e, $r_{k,e} = 2^p u_{k,e} x_{k,e}$. 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

$$\begin{array}{ll} \min & \sum_{k,e} \mathcal{L}_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k \, = \, \frac{2^p}{\lambda^U} d_k, \ \ \forall\, k \\ & r_{k,e} \, \leq \, 2^p u_{k,e} \ \ \forall\, k, e. \end{array} \tag{14}$$

This is just a min-cost flow problem, with integral demands and capacities. Further, for any k and e, $\mathcal{L}_{k,e}$ continuous, piecewise-linear, with breakpoints at the integers and with pieces of strictly increasing slope.
To solve this problem we use an approach similar to that described in [M84] and in [AMO93] (Chapter 14), which essentially amounts to capacity-scaling. The primary computational overhead in this algorithm arises from shortest path computations, used to compute shortest augmenting paths. For completeness, the Appendix provides a more detailed description of the algorithm.
Using the definition of p, we have:

Theorem 5.1 *An ϵ-optimal solution to a maximum concurrent flow problem, on a graph with $\mathcal{N}$ nodes, $\mathcal{M}$ edges, and K commodities can be computed by solving*

$$O^*\left(\epsilon^{-1} K^{3/2} \mathcal{M} \sqrt{\mathcal{N}} \left(\log \frac{1}{\epsilon} + L_U + L_D\right)\right)$$

shortest path problems, plus lower complexity steps, where L_U and L_D, respectively, denote the number of bits needed to store the capacities and demands.

References

[AMO93] R. Ahuja, T. L. Magnanti, and J. Orlin, *Networks Flows: Theory, Algorithms, and Practice*, ISBN 013617549X. Prentice Hall. (1993).

[B02] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems, Theory and Practice*, ISBN 1-4020-7173-6. Kluwer Academic Publishers, Boston (2002). An early version also appears as *CORE Lecture Series Monograph* ISSN-0771 3894, Core, UCL, Belgium (2001) (download from www.core.ucl.ac.be).

[BR02] D. Bienstock and O. Raskina, Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem, *Math. Programming* **91** (2002), 379–492.

[F00] L.K. Fleischer, Approximating Fractional Multicommodity Flow Independent of the Number of Commodities, *SIAM J. Disc. Math.*, **13** (2000), 505 – 520.

[FW56] M. Frank and P. Wolfe, An algorithm for quadratic programming, *Naval Res. Logistics Quarterly* **3** (1956), 149 – 154.

[FGK71] L. Fratta, M. Gerla and L. Kleinrock, The flow deviation method: an approach to store-and-forward communication network design, *Networks* **3** (1971), 97 – 133.

[GaKo98] 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.* (1998) 300-309.

[GK94] M.D. Grigoriadis and L.G. Khachiyan, Fast approximation schemes for convex programs with many blocks and coupling constraints, *SIAM Journal on Optimization* **4** (1994) 86 – 107.

[GK95] M.D. Grigoriadis and L.G. Khachiyan, An exponential-function reduction method for block-angular convex programs, *Networks* **26** (1995) 59-68.

[GK96] M.D. Grigoriadis and L.G. Khachiyan, Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}KNM)$ time, *Mathematical Programming* **75** (1996), 477 – 482.

[M84] M. Minoux. A polynomial algorithm for minimum quadratic cost flows. *European J. Oper. Res.* **18** (1984) 377-387.

[KPST90] 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* (1990), 310 – 321.

[KY98] P. Klein and N. Young, On the number of iterations for Dantzig-Wolfe optimization and packing-covering approximation algorithms, Proceedings IPCO 1999, 320 – 327.

[LMPSTT91] 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* (1991), 101-111.

[N03] Y. Nesterov, Smooth minimization of non-smooth functions, CORE Discussion Paper, CORE, UCL (2003).

[PK95] 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,* (1995), 18-25.

[PST91] S. Plotkin, D.B. Shmoys and E. Tardos, Fast approximation algorithms for fractional packing and covering problems, *Math. of Oper. Res.* **20** (1995), 495 – 504.

[R95] T. Radzik, Fast deterministic approximation for the multicommodity flow problem, *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms* (1995).

[SM91] F. Shahrokhi and D.W. Matula, The maximum concurrent flow problem, *Journal of the ACM* **37** (1991), 318 – 334.

[VG97] 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 (1995), 471 – 485.

A Appendix

A.1 Proof of Theorem 4.2

Lemma A.1 *At any iteration t of Algorithm N_σ, we have that*

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

Proof: Define

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

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

$$\bar{S}_t(x) - \bar{S}_t(\hat{z}^t) = \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^{n}(x_j - \hat{z}_j^t)^2 + [\nabla(\bar{S}_t(\hat{z}^t))]^T(x - \hat{z}^t). \tag{15}$$

Consider the function $\hat{S}_t$, restricted to the one-dimensional segment between $\hat{z}^t$ and x. This function is piecewise-linear, and convex, 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. Since $P \subseteq [0,1]^n$, by Lemma 4.1 (iii), the second term in the right-hand side of (15) is at least $-\frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^{n}\sigma_j$, and consequently:

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

The result now follows by Lemma 4.1 (ii). ■

Theorem 4.2 is established by induction on t. By definition, we have that

$$\begin{aligned}
\hat{S}_0(\hat{z}^0) &= 2K\alpha\sum_{j=1}^{n}\mathcal{L}_{\sigma_j,\hat{x}_j^0}(\hat{z}_j^0) + \Phi(\hat{x}^0) + \left\langle \hat{g}_0, z^0 - x^0 \right\rangle, \\
&\geq K\alpha\sum_{j=1}^{n}\mathcal{L}_{\sigma_j,\hat{x}_j^0}(\hat{z}_j^0) + \Phi(\hat{x}^0) + \left\langle \hat{g}_0, z^0 - x^0 \right\rangle, \\
&\geq K\alpha\sum_{j=1}^{n}\mathcal{L}_{\sigma_j,\hat{x}_j^0}(\hat{y}_j^0) + \Phi(\hat{x}^0) + \left\langle \hat{g}_0, y^0 - x^0 \right\rangle, \\
&\geq \frac{K\alpha}{2}\sum_{j=1}^{n}(y_j^0 - x_j^0)^2 + \Phi(x^0) + \left\langle g_0, y^0 - x^0 \right\rangle \geq \Phi(y^0).
\end{aligned} \tag{16}$$

The first bound in (16) follows from Lemma 4.1(ii) and the second as in Theorem 3.1.
Next, the inductive step. Let $x \in P$. By Lemma A.1, we have

$$\begin{aligned}
\hat{S}_t(x) &\geq \hat{S}_t(\hat{z}^t) + \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^{n}(\hat{z}_j^t - \hat{x}_j^0)^2 - \frac{K\alpha}{(t+1)(t+2)}\sum_{j=1}^{n}\left(2\sigma_j + \sigma_j^2/2\right), \\
&\geq \hat{S}_t(\hat{z}^t) + \frac{2K\alpha}{(t+1)(t+2)}\sum_{j=1}^{n}(\hat{z}_j^t - \hat{x}_j^0)^2 - \frac{3K\alpha}{(t+1)^2}\Big(\sum_{j=1}^{n}\sigma_j\Big),
\end{aligned}$$

Applying the induction hypothesis, and continuing as in the proof of Theorem 3.1, we obtain the following analog of the inequality following (8):

$$\hat{S}_{t+1}(x) \geq \Phi(\hat{x}^{t+1}) + \min_{y \in P} \Big\{ \frac{K\alpha}{2} \sum_j \left(y_j - \hat{x}_j^{t+1} \right)^2 + \left\langle \hat{g}_{t+1}, y - \hat{x}^{t+1} \right\rangle \Big\} - 3K\alpha \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big).$$

Applying Lemma 4.1 again, we obtain

$$\begin{aligned}
\hat{S}_{t+1}(x) \;\; \geq \;\; & \Phi(\hat{x}^{t+1}) + K\alpha \sum_j \mathcal{L}_{\sigma_j, \hat{x}_j^{t+1}} \left(\hat{y}_j^{t+1} \right) + \left\langle \hat{g}_{t+1}, \hat{y}^{t+1} - \hat{x}^{t+1} \right\rangle \\
& -3K\alpha \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big), \\
\geq \;\; & \Phi(\hat{x}^{t+1}) + \frac{K\alpha}{2} \sum_j \left(\hat{y}_j^{t+1} - \hat{x}_j^{t+1} \right)^2 + \left\langle \hat{g}_{t+1}, \hat{y}^{t+1} - \hat{x}^{t+1} \right\rangle \\
& -3K\alpha \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big), \\
\geq \;\; & \Phi(\hat{y}^{t+1}) - 3K\alpha \Big(\sum_{h=1}^{t+1} \frac{1}{h^2} + t + 1 \Big) \Big(\sum_{j=1}^{n} \sigma_j \Big),
\end{aligned}$$

where the last inequality follows from Lemma 3.3. ■

A.2 Solving the piecewise-linear problems min-cost flow problems

We are given an optimizatin problem of the form described in Section 5.1,

$$\begin{array}{ll} \min & \sum_{k,e} L_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = \hat{d}_k, \quad 0 \leq r_k \leq \hat{u}_k, \qquad k = 1, \ldots, K, \end{array} \tag{17}$$

where for every k and e, $L_{k,e}$ is continuous, convex, piecewise-linear, with breakpoints at the integers and with pieces of strictly increasing slope. Also, $\hat{d}_k$ and $\hat{u}_k$ are integral and $u_{k,e} \leq 2^p$ for any k and e.

If we compute a feasible, integral flow (which can be done in strongly polynomial time), we can thus express the problem in *circulation form* the form

$$\begin{array}{ll} \min & \sum_{k,e} L_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = 0, \quad -\alpha_k \leq r_k \leq \beta_k, \qquad k = 1, \ldots, K, \end{array} \tag{18}$$

where for every k and e, $\alpha_{k,e}$ and $\beta_{k,e}$ are nonnegative and integral and of value $\leq 2^p$, and $L_{k,e}$ is a convex, continuous piecewise-linear function with breakpoints at the integers.

We will solve problem (18) by solving a sequence of problems – our approach is similar Let $0 \leq h \leq p$ be an integer. For any k and e, define $L^h_{k,e}(r_{k,e})$ to be the continuous, piecewise-linear function, with breakpoints at the integer multiples of 2^h, where it agrees with $L_{k,e}(r_{k,e})$. Thus, $L^h_{k,e}$ is convex. Further, define $\alpha^h_{k,e}$ to be the smallest integer multiple of 2^h that is at least as large as $\alpha_{k,e}$, and similarly define $\beta^h_{k,e}$. Then, our *level-h* problem is:

$$\begin{array}{ll} \min & \sum_{k,e} L^h_{k,e}(r_{k,e}) \\ \text{s. t.} & N r_k = 0, \quad -\beta^h_k \leq r_k \leq \alpha^h_{k,}, \qquad k = 1, \ldots, K, \end{array} \tag{19}$$

Thus, the level-0 problem is (18), and we will solve it by solving the level-p problem, then the level-$(p-1)$ problem, and so on inductively. Note that for $0 \le h \le p$, the function $L^h_{k,e}$ has 2^{p-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^{p-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 $\hat{G}^h$, as we will see.
The level-p problem is a standard minimum-cost circulation problem, and without loss of generality we can compute an optimal circulation, all of whose entries are integer multiples of 2^p, in strongly polynomial time, as well as a set of optimal node potentials.
Inductively, suppose we have solved the level-h problem, and we have an optimal circulation r^h_k $(k = 1, \cdots, K)$ for this problem, each of whose entries is an integral multiple of 2^h, as well as an optimal set of node potentials π^h_k. Our task is to refine r^h_k into an optimal (and feasible) circulation for the level-$(h-1)$ problem.
Note that by definition, for any k and e,

(a) $L^h_{k,e}$ and $L^{h-1}_{k,e}$ agree at the integer multiples of 2^{h-1}, and

(b) Let $q \in Z_+$. Then the slope of $L^{h-1}_{k,e}$ is less (resp., more) than the slope of $L^h_{k,e}$ 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 $\alpha^{h-1}_{k,e} = \alpha^h_{k,e}$ or $\alpha^{h-1}_{k,e} = \alpha^h_{k,e} - 2^{h-1}$, and similarly with $\beta^{h-1}_{k,e}$ and $\beta^h_{k,e}$.

Thus, it is easy to see that r^h_k, together with the potentials π^h_k, *nearly* satisfies the optimality conditions for the level-$(h-1)$ problem. More precisely, suppose we were to 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 $r^h_{k,e} = \alpha^h_{k,e} > \alpha^{h-1}_{k,e}$ or in the similar case for $\beta^h_{k,e}$.
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 together with the potentials π^h_k or is an overflow arc. Consequently, we can obtain an optimal circulation in $\hat{G}^{h-1}$ in at most $O(\mathcal{M})$ flow pushes (each pushing 2^{h-1} units of flow) or recomputations of node potentials; and each such step requires the solution of a shortest path problem. It is clear that (again because of (a) - (c)) 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.