An Accelerated Newton–Dinkelbach Method and its Application to Two Variables Per Inequality Systems

Daniel Dadush **Z.K. Koh** Bento Natura László A. Végh

CWI Centrum Wiskunde & Informatica

LSE The London School of Economics and Political Science

Linear fractional optimization

- Given a closed domain $\mathcal{D} \subseteq \mathbb{R}^m$ and $c, d \in \mathbb{R}^m$ where $d^\top x > 0$ for all $x \in \mathcal{D}$, solve

$$\inf_{x \in \mathcal{D}} \frac{c^\top x}{d^\top x}.$$

- If $\mathcal{D} \subseteq \{0, 1\}^m$, it is called linear fractional combinatorial optimization.

- E.g. Minimum cost-to-time ratio cycle, minimum ratio spanning tree.

- Megiddo invented parametric search to solve this problem.

Requires: An affine algorithm for the nonfractional problem $\min_{x \in \mathcal{D}} c^\top x$.

- Parametric search simulates this algorithm for the problem

$$\min_{x \in \mathcal{D}} (c - \delta d)^\top x$$

with the parameter δ being indeterminate.

Newton–Dinkelbach method

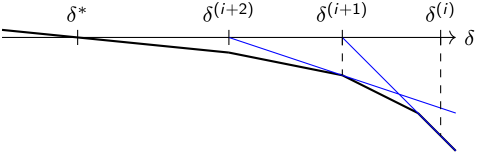
- The parametric function is concave, decreasing, piecewise-linear.

$$f(\delta) = \min_{x \in \mathcal{D}} (c - \delta d)^\top x.$$

- Let δ^* denote the optimal value. Then, $f(\delta) = 0 \iff \delta = \delta^*$.

- Use a root-finding technique like Newton's method.

Requires: An algorithm for the nonfractional problem $\min_{x \in \mathcal{D}} c^\top x$.



- Newton's method terminates in strongly polynomial number of iterations [Radzik '92].

Linear fractional programming

- If $\mathcal{D}$ is a polyhedron, then the problem is a linear fractional program.

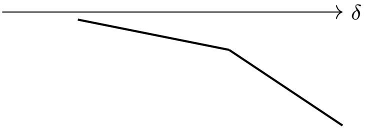
- Let us not assume $d^\top x > 0$ for all $x \in \mathcal{D}$. Instead, we solve

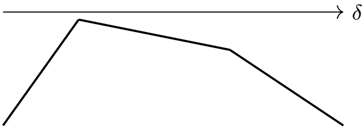
$$\inf_{x \in \mathcal{D}} \frac{c^\top x}{d^\top x} \quad \text{s.t. } d^\top x > 0.$$

- The parametric function $f : \mathbb{R} \to \mathbb{R} \cup \{-\infty\}$ is still concave and piecewise-linear, but no longer decreasing.

$$f(\delta) = \inf_{x \in \mathcal{D}} (c - \delta d)^\top x.$$

- f may not have a root.

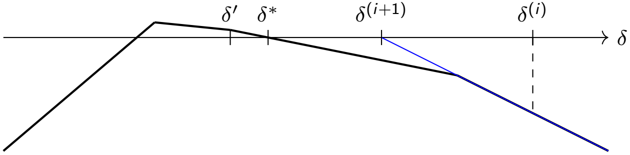




Accelerating Newton–Dinkelbach

Idea: At the end of each iteration, look-ahead to the point

$$\delta' := 2\delta^{(i+1)} - \delta^{(i)}$$



• Overwrite the next point $\delta^{(i+1)} \leftarrow \delta'$ if we did not overshoot ($\delta' \geq \delta^*$). This can be checked via

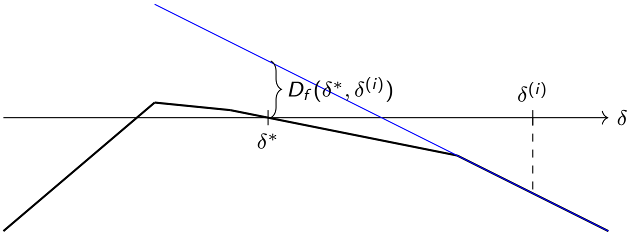
$$-\infty < f(\delta') < 0 \quad \text{and} \quad f'(\delta') < 0.$$

Motivation: Runtime bottleneck caused by consecutive iterations in which the gradient does not change by much. So, skip over them.

Bregman divergence

Def: For a concave function f, the Bregman divergence is

$$D_f(\delta^*, \delta^{(i)}) := f(\delta^{(i)}) + f'(\delta^{(i)})(\delta^* - \delta^{(i)}) - f(\delta^*)$$



- With acceleration, $D_f(\delta^*, \delta^{(i)}) \leq \frac{1}{2} D_f(\delta^*, \delta^{(i-2)})$ for all $i > 2$.

Intuition: If look-ahead succeeded ($\delta' \geq \delta^*$), then we made significant progress. Otherwise, we are not too far away from δ^*.

Linear fractional comb opt

Thm: For $\mathcal{D} \subseteq \{0,1\}^m$, the look-ahead Newton–Dinkelbach method terminates in $O(m \log m)$ iterations.

- $O(m^2 \log m)$ iterations without acceleration [Wang, Yang, Zhang '06].

Proof sketch:

- For each $i \geq 1$, $f'(\delta^{(i)}) = -d^\top x^{(i)}$ for some $x^{(i)} \in \mathcal{D}$.

- Bregman divergence is a modified cost of $x^{(i)}$

$$D_f(\delta^*, \delta^{(i)}) = (c - \delta^* d)^\top x^{(i)}.$$

- Bregman divergence halves every two iterations

$$0 \leq (c - \delta^* d)^\top x^{(i)} \leq \frac{1}{2}(c - \delta^* d)^\top x^{(i-2)}$$

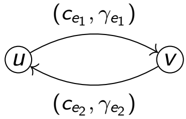
- Such a sequence has length $O(m \log m)$ [Goemans '92]. ■

Two variables per inequality (2VPI) system

Problem: Given $A \in \mathbb{R}^{m \times n}$ with at most 2 nonzero entries per row and $c \in \mathbb{R}^m$, find a feasible solution to $Ay \leq c$ or report infeasibility.

- WLOG, every inequality is of the form $y_u - \gamma_e y_v \leq c_e$, where $\gamma_e > 0$.

- Represent as a directed multigraph G on n nodes and m arcs, with arc costs $c \in \mathbb{R}^m$ and gain factors $\gamma \in \mathbb{R}^m_{>0}$.



- If the system is bounded and feasible, then it has a unique pointwise maximal solution y^*, i.e.

$$y^* \geq y \quad \text{for any feasible solution } y.$$

History of 2VPI

• A quasipolynomial Fourier–Motzkin elimination [Nelson '78].

• Characterization of feasibility in terms of cycles and bicycles in G [Shostak '81].

• The first weakly polynomial algorithm [Aspvall, Shiloach '79].

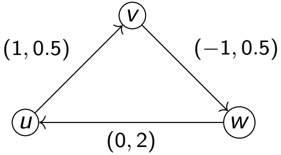
• Parametric search + AS algorithm $\Rightarrow$ strongly polynomial algorithm [Megiddo '83].

• Faster strongly polynomial algorithms [Cohen, Megiddo '94].

• The fastest strongly polynomial algorithm is also based on Fourier–Motzkin elimination, with a running time of $O(mn^2 \log m)$ [Hochbaum, Naor '94].

Pointwise maximal solution

• A directed cycle C is flow-absorbing if $\prod_{e \in E(C)} \gamma_e < 1$.



$$
\begin{aligned}
y_u - 0.5 y_v &\le 1 \\
y_v - 0.5 y_w &\le -1 \\
y_w - 2 y_u &\le 0
\end{aligned}
$$

• Flow-absorbing cycles induce upper bounds on the variables.

$$
\begin{aligned}
y_u - 0.5 y_u \le 0.5 &\implies y_u \le 1 \\
y_v - 0.5 y_v \le 0 &\implies y_v \le 0 \\
y_w - 0.5 y_w \le 1 &\implies y_w \le 2
\end{aligned}
$$

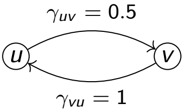
• Computing the pointwise maximal solution y^* amounts to finding the **best** cycle upper bounds.

Connection to linear fractional programming

- Primal-dual LPs for y_u^*

$$
\begin{aligned}
\min\ & c^\top x \\
\text{s.t.}\ & \text{netflow at } u = 1 \\
& \text{netflow at } v = 0 \quad \forall v \neq u \\
& x \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
\max\ & y_u \\
\text{s.t.}\ & y_v - \gamma_e y_w \leq c_e \quad \forall e = vw \in E.
\end{aligned}
$$

- Our domain is $\mathcal{D} := \{x \geq 0 : u \text{ has outflow } 1, v \text{ has netflow } 0\ \forall v \neq u\}$.



$x_{uv} = 2, x_{vu} = 1$

$\frac{1}{2} x \in \mathcal{D}$

- The primal LP is equivalent to the following linear fractional program

$$\inf_{x \in \mathcal{D}} \frac{c^\top x}{1 - \text{inflow at } u} \quad \text{s.t. } 1 - \text{inflow at } u > 0.$$

Connection to linear fractional programming

- Linear fractional program for y_u^*

$$\inf_{x \in \mathcal{D}} \frac{c^\top x}{1 - \text{inflow at } u} \quad \text{s.t. } 1 - \text{inflow at } u > 0.$$

- For any $\delta \in \mathbb{R}$, the value of the parametric function $f(\delta)$ is given by

$$\begin{array}{ll|lll} \min & c^\top x - \delta(1 - \text{inflow at } u) & \max & y_u - \delta & \\ \text{s.t.} & x \in \mathcal{D} & \text{s.t.} & y_v - \gamma_e \delta \le c_e & \forall e = vu \in \delta^-(u) \\ & & & y_v - \gamma_e y_w \le c_e & \forall e = vw \notin \delta^-(u). \end{array}$$

- $f(\delta)$ can be evaluated using a label-correcting algorithm.

$$y_v - \gamma_e y_w > c_e \quad \Longrightarrow \quad y_v \leftarrow c_e + \gamma_e y_v$$

- Loop over the arc set for n times à la Bellman–Ford.

Label-correcting algorithms

Shortest paths: Given a directed graph $G = (V, E)$ with arc costs $c \in \mathbb{R}^E$ and a target node t, find a shortest path from every node to t.

- Can be formulated as a 2VPI system:

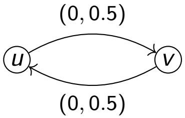
$$y_u - y_v \leq c_{uv} \qquad \forall uv \in E$$
$$y_t = 0$$

- The pointwise maximal solution gives shortest path distances to t.

Label-correcting: Start with high values of y, and repeatedly correct any violated contraints, i.e.

$$y_u - y_v > c_{uv} \quad \Longrightarrow \quad y_u \leftarrow c_{uv} + y_v$$

- Can this method be extended to general 2VPI systems?



$$y_u \leq 0.5 y_v$$
$$y_v \leq 0.5 y_u$$

Label-correcting algorithm for 2VPI

• Start with a subsystem for which the pointwise maximal solution y^* is trivial, and then progressively compute y^* for larger subsystems.

Input: A 2VPI system (G, c, γ).

Output: y^* if the system is feasible; `INFEASIBLE` otherwise.

1. Initialize node labels $y \in \mathbb{R}^n$
2. $k \leftarrow 0$, $G^{(0)} \leftarrow (V, \emptyset)$
3. **for each** $u \in V$:
   $G^{(k+1)} \leftarrow G^{(k)} \cup \delta^+(u)$
   $y^* \leftarrow$ pointwise maximal solution to the subsystem $(G^{(k+1)}, c, \gamma)$
   via accelerated Newton–Dinkelbach
   if f does not have a root:
   return `INFEASIBLE`
   $k \leftarrow k + 1$
4. **return** y

Strongly polynomial analysis

Thm: For each subsystem $(G^{(k)}, c, \gamma)$, the accelerated Newton's method terminates in $O(m)$ iterations.

Proof idea:

- For each $i \geq 1$, the Bregman divergence is the reduced cost of a path

$$D_f(\delta^*, \delta^{(i)}) = c^*(P^{(i)})$$

- The Bregman divergence halves every two iterations

$$0 \leq c^*(P^{(i)}) \leq \frac{1}{2} c^*(P^{(i-2)})$$

- This sequence of paths satisfy a certain subpath monotonicity property.

- After every 2 iterations, an arc ceases to appear in future paths. ■

$\Rightarrow$ Total running time $O(m^2 n^2)$.

Summary

- We accelerate the Newton–Dinkelbach method, and give an analysis using Bregman divergence.

- Applications:
  - A faster algorithm for linear fractional comb opt.
  - An iterative $O(m^2n^2)$ algorithm for 2VPI systems. This strengthens a weakly polynomial result for Newton's method on deterministic Markov Decision Process [Madani '02].

- Further questions:
  - Can we make our algorithm competitive with Hochbaum–Naor's $O(mn^2 \log m)$ algorithm? Is our analysis tight?
  - Apply the accelerated Newton–Dinkelbach method to other fractional optimization problems.
  - Are there better acceleration schemes for the Newton–Dinkelbach method?

Thank You!