The Feasibility of Implementing Maximum Flow and Minimum-Cost Flow in Almost-Linear Time with an IPM

Exam presentation

Adrian Borup, Albert Rise Nielsen

Research Project
IT University of Copenhagen

December 18, 2024

Preamble

- Chen, Kyng, Liu, Peng, Gutenberg, and Sachdeva presented a breakthrough in 2022
- Solves minimum-cost flow and maximum flow in $m^{1+o(1)}$ time
- Two key components:
  1. Interior-point method (IPM) solving $m^{1+o(1)}$ subproblems.
  2. A dynamic graph data structure to solve subproblems efficiently.
- No known implementation.
- Is the algorithm feasible in practice?

Table of Contents

Table of Contents

Background: Minimum-cost flow

"the cheapest possible way of sending flow, within some capacities, through a flow network"

Background: Minimum-cost flow

- Directed graph
- Lower and upper capacities
- Cost to use: $c \cdot f$
- Vertex demands
- Minimise cost while respecting capacities and demands

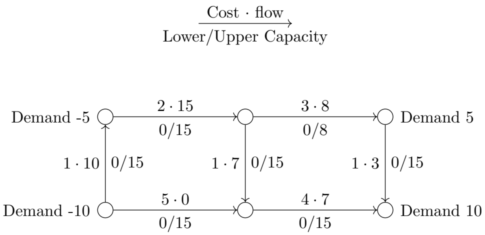


Table of Contents

Min-cost flow as a linear program

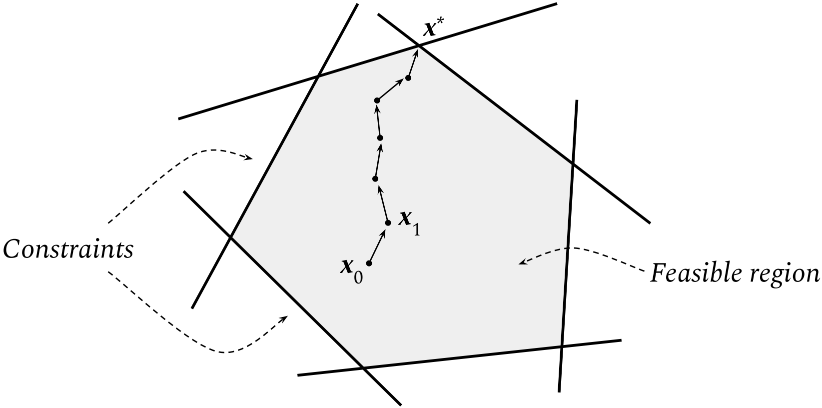
Define:

$\boldsymbol{f} \in \mathbb{R}^E$	the flow in each edge
$\boldsymbol{c} \in \mathbb{R}^E$	the cost of each edge
$\boldsymbol{u}_e^-, \boldsymbol{u}_e^+ \in \mathbb{R}^E$	the lower/upper capacity of each edge
$\boldsymbol{d} \in \mathbb{R}^V$	the demand of each vertex
$\mathbf{B} \in \mathbb{R}^{E \times V}$	edge-vertex incidence matrix

---

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^\top \boldsymbol{f} && \text{(minimum cost)} \\ \text{subject to} \quad & \mathbf{B}^\top \boldsymbol{f} = \boldsymbol{d} && \text{(vertex demand constraints)} \\ \text{and} \quad & \boldsymbol{u}_e^- \le \boldsymbol{f}_e \le \boldsymbol{u}_e^+ && \text{(edge capacity constraints)} \end{aligned}$$

Optimising a linear program: the interior-point method

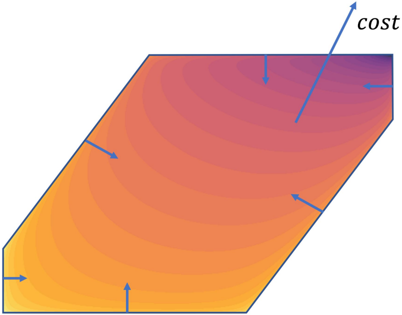


A potential-reduction IPM for min-cost flow

$$\Phi(\boldsymbol{f}) = \underbrace{m \log(\boldsymbol{c}^\top \boldsymbol{f} - F^*)}_{\text{Objective}} \underbrace{- \sum_{e \in E} \left(\log(\boldsymbol{u}_e^+ - \boldsymbol{f}_e) + \log(\boldsymbol{f}_e - \boldsymbol{u}_e^-)\right)}_{\text{Barrier}}$$

A potential-reduction IPM for min-cost flow

$$\Phi(\boldsymbol{f}) = \underbrace{m \log(\boldsymbol{c}^\top \boldsymbol{f} - F^*)}_{\text{Objective}} - \underbrace{\sum_{e \in E} \left(\log(\boldsymbol{u}_e^+ - \boldsymbol{f}_e) + \log(\boldsymbol{f}_e - \boldsymbol{u}_e^-)\right)}_{\text{Barrier}}$$



- Goal: minimise $\Phi(\boldsymbol{f})$
- Objective → minimises cost
- Barrier → avoids boundary
- It's a balance

[LC22; Kar84]

What do we need?

1. An initial point: some feasible flow
2. A step function: find some flow that decreases the potential

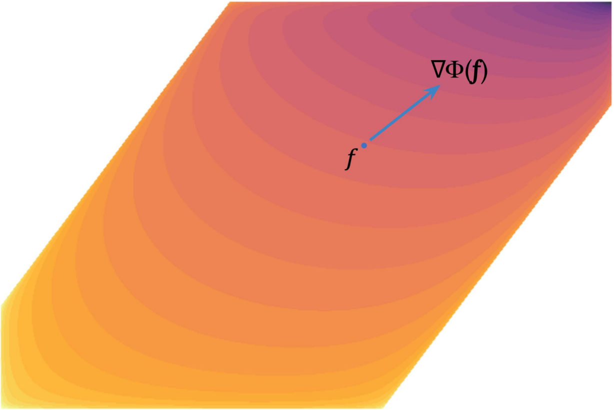
Dropping the potential

Question: How do we find which direction decreases $\Phi(\boldsymbol{f})$?

Dropping the potential

Question: How do we find which direction decreases $\Phi(\boldsymbol{f})$?

Answer: $\nabla\Phi(\boldsymbol{f})$, the gradient of $\Phi(\boldsymbol{f})$



Dropping the potential

Question: How do we find a *valid flow* that decreases $\Phi(\boldsymbol{f})$?

Dropping the potential

Question: How do we find a *valid flow* that decreases $\Phi(\boldsymbol{f})$?

Idea:

- We have the constraint $\mathbf{B}^\top \boldsymbol{f} = \boldsymbol{d}$
- Find a circulation Δ such that $\mathbf{B}^\top \Delta = \mathbf{0}$
  Augmenting Δ gives $\mathbf{B}^\top (\boldsymbol{f} + \Delta) = \boldsymbol{d}$, still respecting demands
- Goal: find a Δ that steps in the direction of the gradient.

Minimum-ratio cycles (in general)

Define:

C	a cycle consisting of a set of edges
$c(e)$	the cost of an edge
$t(e)$	the time of an edge

Minimum-ratio cycles (in general)

Define:

C	a cycle consisting of a set of edges
$c(e)$	the cost of an edge
$t(e)$	the time of an edge

Find the cycle that minimises:

$$\sum_{e \in C} c(e) / \sum_{e \in C} t(e)$$

The min-ratio cycle subproblem in the IPM

$$\begin{array}{rl} \mathbf{g} \in \mathbb{R}^E & \text{edge gradients, } \mathbf{g} = \nabla\Phi(\boldsymbol{f}) \\ \boldsymbol{\ell} \in \mathbb{R}^E_{>0} & \text{edge lengths, } \boldsymbol{\ell}_e = 1/\left(\min\left(\boldsymbol{u}^+_e - \boldsymbol{f}_e, \boldsymbol{f}_e - \boldsymbol{u}^-_e\right)\right) \\ \mathbf{L} \in \mathbb{R}^{E\times E} & \mathbf{L} = \mathrm{diag}(\boldsymbol{\ell}) \end{array}$$

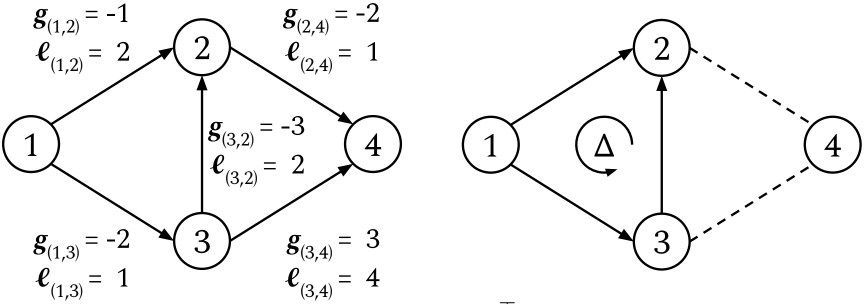
$$\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$$

In simpler terms:

Find a circulation Δ that minimises $\dfrac{\sum_e \mathbf{g}_e \cdot \Delta_e}{\sum_e |\boldsymbol{\ell}_e \cdot \Delta_e|}$

Examples

$$\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$$

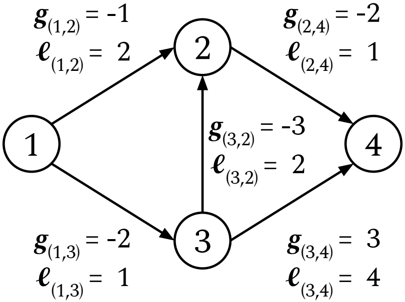


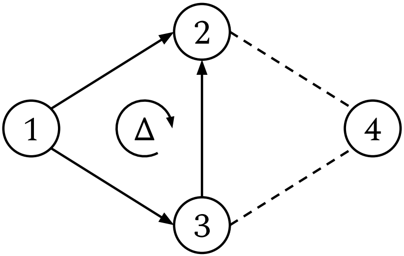
$$\mathbf{g}^\top \Delta = 1 + (-2) + (-3) = -4$$

$$\|\mathbf{L}\Delta\|_1 = 2 + 1 + 2 = 5$$

Examples

$$\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$$





$$\mathbf{g}^\top \Delta = -1+2+3 = 4$$
$$\|\mathbf{L}\Delta\|_1 = \quad 2+1+2 = 5$$

Why min-ratio cycles?

What are the benefits of using min-ratio cycles?

1. Gradients ensure good step directions for IPM
2. Outputs simple cycles
3. [CKLPGS22] can compute *approximate* min-ratio cycles very efficiently

Combining everything

Given some initial flow $\boldsymbol{f}^{(0)}$, perform iterations:

1. Compute $\mathbf{g}(\boldsymbol{f}^{(t)})$ and $\ell(\boldsymbol{f}^{(t)})$

Combining everything

Given some initial flow $\boldsymbol{f}^{(0)}$, perform iterations:

1. Compute $\mathbf{g}(\boldsymbol{f}^{(t)})$ and $\ell(\boldsymbol{f}^{(t)})$
2. Find min-ratio cycle Δ with $\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$

Combining everything

Given some initial flow $\boldsymbol{f}^{(0)}$, perform iterations:

1. Compute $\mathbf{g}(\boldsymbol{f}^{(t)})$ and $\ell(\boldsymbol{f}^{(t)})$
2. Find min-ratio cycle Δ with $\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$
3. Compute scaling factor $\eta = -\kappa^2/(50 \cdot \mathbf{g}^\top \Delta)$

Combining everything

Given some initial flow $\boldsymbol{f}^{(0)}$, perform iterations:

1. Compute $\mathbf{g}(\boldsymbol{f}^{(t)})$ and $\ell(\boldsymbol{f}^{(t)})$
2. Find min-ratio cycle Δ with $\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$
3. Compute scaling factor $\eta = -\kappa^2/(50 \cdot \mathbf{g}^\top \Delta)$
4. Update the flow, $\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{f}^{(t)} + \eta\Delta$

Combining everything

Given some initial flow $\boldsymbol{f}^{(0)}$, perform iterations:

1. Compute $\mathbf{g}(\boldsymbol{f}^{(t)})$ and $\ell(\boldsymbol{f}^{(t)})$
2. Find min-ratio cycle Δ with $\min_{\mathbf{B}^\top\Delta=0} \frac{\mathbf{g}^\top\Delta}{\|\mathbf{L}\Delta\|_1}$
3. Compute scaling factor $\eta = -\kappa^2/(50 \cdot \mathbf{g}^\top\Delta)$
4. Update the flow, $\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{f}^{(t)} + \eta\Delta$

Run $m^{1+o(1)}$ iterations, then round the result.

Running time

Problem: exact $\mathbf{g}$, ℓ, and min-ratio cycles are expensive to compute.

Solution: use *approximations*.

Running time

Problem: exact $\mathbf{g}$, ℓ, and min-ratio cycles are expensive to compute.

Solution: use *approximations*.

Claims by [CKLPGS22]:

1. IPM can terminate in $m^{1+o(1)}$ iterations
2. An $m^{o(1)}$-approximate solution for min-ratio cycle suffices
3. Can compute $m^{o(1)}$-approximate solution in amortized $m^{o(1)}$ time
4. At most $m^{1+o(1)}$ changes to $\mathbf{g}$ and ℓ

Running time

Problem: exact $\mathbf{g}$, ℓ, and min-ratio cycles are expensive to compute.

Solution: use *approximations*.

Claims by [CKLPGS22]:

1. IPM can terminate in $m^{1+o(1)}$ iterations
2. An $m^{o(1)}$-approximate solution for min-ratio cycle suffices
3. Can compute $m^{o(1)}$-approximate solution in amortized $m^{o(1)}$ time
4. At most $m^{1+o(1)}$ changes to $\mathbf{g}$ and ℓ

$m^{1+o(1)}$ iterations, each running in amortized $m^{o(1)}$ time.
Total $m^{1+o(1)}$ time.

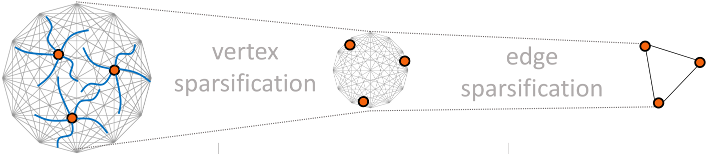
Extras

- How do we find an initial feasible flow? $O(m)$ time algorithm described in report.
- IPM assumes that optimal cost is known. When unknown, binary search can be applied. Described in report.

Table of Contents

Dynamic min-ratio cycle

The fundamental idea of the data structure is to reduce the number of vertices and edges as much as possible while supporting rebuilds, slowly-changing $\mathbf{g}$, ℓ, and graph updates.



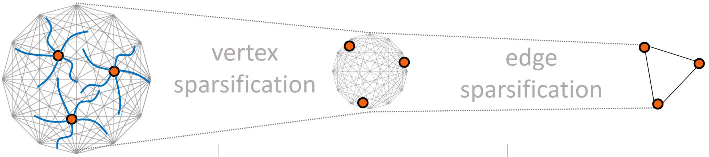
$G = G_0$	$C(G_0, F)$	$S\big(C(G_0, F)\big) = G_1$
$\approx m$ edges	$\approx m$ edges	$\approx m/K$ edges
$\approx m$ vertices	$\approx m/K$ vertices	$\approx m/K$ vertices
	"core graph"	"sparsified core graph"

Dynamic min-ratio cycle

The goal is to approximately solve $\min_{\mathbf{B}^\top \Delta = 0} \frac{\mathbf{g}^\top \Delta}{\|\mathbf{L}\Delta\|_1}$ with the help of a specialized data structure

Algorithm:

1. Sample a random "low-stretch spanning tree" T
2. Return the best "tree cycle" in T ($\text{cycle}_T(e)$)
3. Repeat $O(\log n)$ times



$G = G_0$	$C(G_0, F)$	$S(C(G_0, F)) = G_1$
$\approx m$ edges	$\approx m$ edges	$\approx m/K$ edges
$\approx m$ vertices	$\approx m/K$ vertices	$\approx m/K$ vertices
	"core graph"	"sparsified core graph"

Table of Contents

Implementation

We implement a partial, static algorithm:

- Focus on IPM
- Ignore dynamic data structure

Still allows us to evaluate claims about running time via the number of iterations.

Implementation

We implement a partial, static algorithm:

- Focus on IPM
- Ignore dynamic data structure

Still allows us to evaluate claims about running time via the number of iterations.

Differences from [CKLPGS22]:

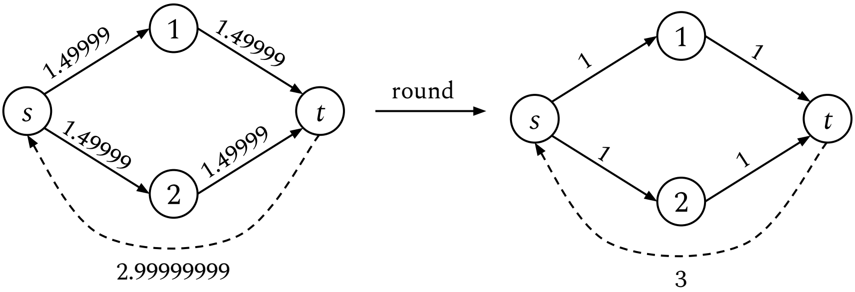
- We use an *inefficient*, *exact* oracle for minimum-ratio cycles.
- We take larger steps than suggested
- We devise our own binary search
- . . . see more in report

Correctness

Test suite:

- Focus on maximum flow. Tests from various sources.
- Almost all test cases yield correct maximum flow value.
- Some tests fail flow conservation checks.
- Some crash due to overflow.

Problems: fractional flows



Other problems

- Numeric overflows
- Floating point numbers and precision
- Off-by-one error in binary-search

Experimental results

Grows almost linearly in m, but the constants are huge.

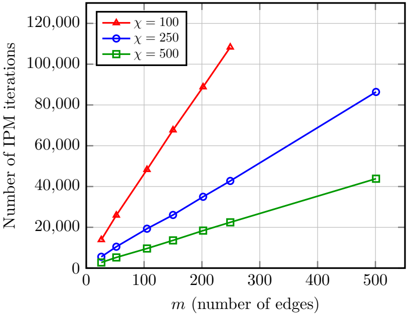


Figure: All graphs are directed acyclic graphs

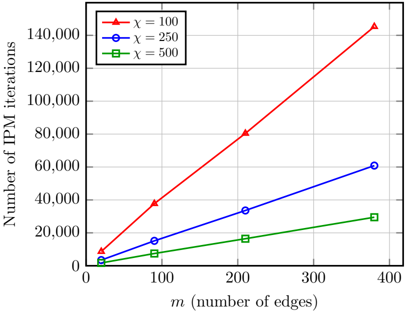


Figure: All graphs are fully connected graphs

Experimental results

[CKLPGS22] claims running time of $m^{1+o(1)} \log U$ for max flow.

The impact of U on the running time when binary-searching:

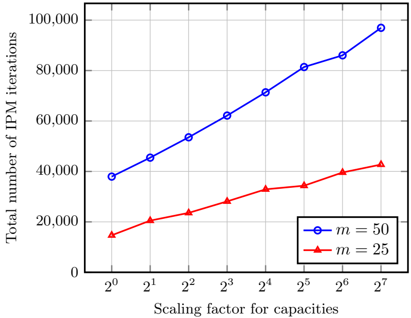


Figure: The total number of IPM iterations for each scaling factor, shown with a logarithmic x-scale.

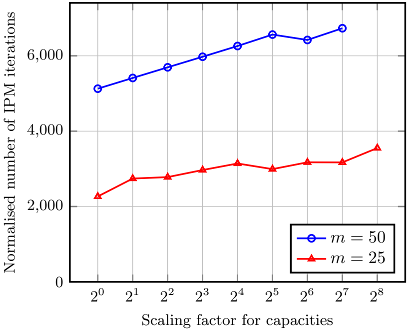
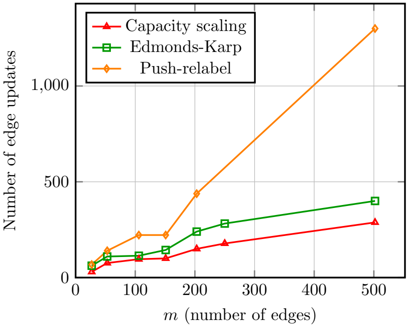


Figure: The number of IPM iterations, normalised by dividing by $\log_2(F_{\max})$ where $F_{\max}$ is the upper limit for the binary search.

Experimental results

Edge updates compared to other flow algorithms.



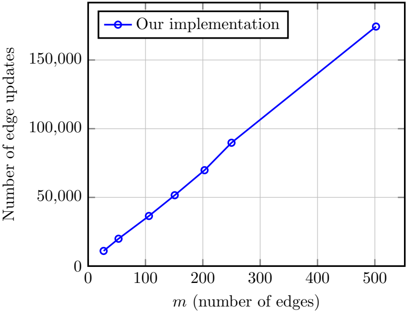


Table of Contents

Conclusion and scope for masters thesis

- We give a high-level overview of the theory behind [CKLPGS22]
- We provide a static algorithm for the IPM
- We evaluate the algorithm on its usefulness in practice

Conclusion and scope for masters thesis

- We give a high-level overview of the theory behind [CKLPGS22]
- We provide a static algorithm for the IPM
- We evaluate the algorithm on its usefulness in practice
- We suggest that [CKLPGS22] is not very practical.
  - Continuous optimisation is problematic in practice
  - High number of iterations
  - Precision issues
  - Complex
- Scope for master's thesis
  - We could attempt to implement the data structure.
  - Work by Bernstein, Blikstad, Saranurak, and Tu ([BBST24]) seems promising and implementable.

Questions

It's time for questions.

References

[BBST24] Aaron Bernstein, Joakim Blikstad, Thatchaphol Saranurak, and Ta-Wei Tu. *Maximum Flow by Augmenting Paths in $n^{2+o(1)}$ Time*. arXiv:2406.03648. June 2024. DOI: 10.48550/arXiv.2406.03648. URL: http://arxiv.org/abs/2406.03648 (visited on 11/13/2024).

[CKLPGS22] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. *Maximum Flow and Minimum-Cost Flow in Almost-Linear Time*. arXiv:2203.00671. Apr. 2022. DOI: 10.48550/arXiv.2203.00671. URL: http://arxiv.org/abs/2203.00671 (visited on 11/13/2024).

[Kar84] Narendra Karmarkar. "A New Polynomial-Time Algorithm for Linear Programming-II". In: *Combinatorica* 4 (Dec. 1984), pp. 302–311. DOI: 10.1007/BF02579150.

[LC22] Yang P. Liu and Li Chen. *Maximum Flow and Minimum-Cost Flow in Almost Linear Time*. Stanford, Mar. 2022. URL: https://yangpliu.github.io/pdf/Flows_Stanford.pdf.