A Novel Dual Ascent Algorithm for Solving the Min-Cost Flow Problem

Ruben Becker, Maximilan Fickert and Andreas Karrenbauer

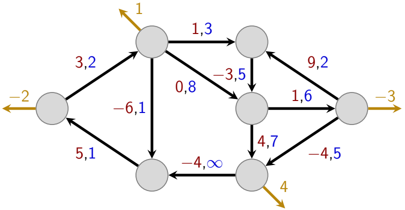
Max Planck Institute for Informatics

January 11, 2016

The Min-Cost Flow Problem

Given **directed** graph $G = (V, A)$ with

- arc **costs** $c \in \mathbb{Z}^m$, arc **capacities** $u \in (\mathbb{N} \cup \{\infty\})^m$
- node **demands** $b \in \mathbb{Z}^n$ with $\mathbb{1}^T b = 0$



A flow $x \in \mathbb{R}^m$ is **feasible**, if

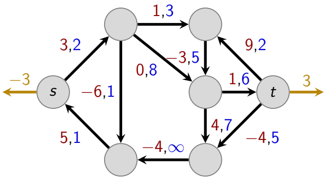
- for all $v \in V$:
  $x(\delta^{\text{in}}(v)) - x(\delta^{\text{out}}(v)) = b_v$,
- $0 \leq x \leq u$

The Min-Cost Flow Problem

Output feasible flow $x^* \in \mathbb{R}^m$ with $c^T x^* \leq c^T x$ for all feasible x

or infeasible/unbounded.

LP-Formulation and Dual Problem



Node-Arc Incidence Matrix $A \in \mathbb{R}^{n \times m}$

$$A_{va} = \begin{cases} -1 & \text{if } a = (v, \_) \\ 1 & \text{if } a = (\_, v) \\ 0 & \text{otherwise.} \end{cases}$$

- **Primal Dual LP pair:**

  $$\min\{c^T x : Ax = b \text{ and } 0 \le x \le u\}$$
  $$= \max\{b^T y - u^T z : A^T y - z \le c \text{ and } z \ge 0\}$$

- **Dual Constraints:** $y_w - y_v - z_a \le c_a$, $z_a \ge 0$ for all $a = (v, w) \in A$

- **Dual Ascent:**
  - Start with $y = 0$ and $x = 0$
  - Dual Step: improve **optimality of** y
  - Primal Step: improve **feasibility of** x

Previous Work

Classical Combinatorial

- $O(m \log U \cdot \mathrm{SP}_+(n, m, C))$
  Edmonds and Karp, 1972
- $O(m^2 \log n \cdot \mathrm{MF}(n, m))$
  Tardos, 1985
- $O(nm \log\log U \cdot \log(nC))$
  Ahuja, Goldberg, Orlin and Tarjan, 1992
- etc.

Interior Point Methods

- $O(n^2 \sqrt{m} \log(n \max\{U, C\}))$
  Vaidya, 1989
- $O(nm^2 L)$ combinatorial
  Wallacher and Zimmermann, 1992
- expected $\tilde{O}(m^{3/2} \log U)$
  Daitch and Spielman, 2008
- expected $\tilde{O}(m\sqrt{n} \log^2 U)$,
  Lee and Sidford, 2014

- In **theory the interior point methods** dominate over the combinatorial approaches.
- In **practice the combinatorial algorithms** perform quite well.

Preliminaries and Invariants

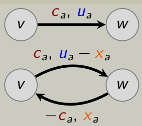
Let $y \in \mathbb{R}^n$ and $x \in \mathbb{R}^m_{\geq 0}$, define

- **reduced costs:** $c_a^y := c_a + y_v - y_w$
- **residual demands:** $b_v^x = b_v - \sum_{a \in \delta_v^{\text{in}}} x_a + \sum_{a \in \delta_v^{\text{out}}} x_a$

Complementary Slackness Conditions

$$
\begin{aligned}
c_a^y > 0 &\implies x_a = 0 \\
c_a^y < 0 &\implies x_a = u_a \qquad \text{for all } a \in A. \\
0 < x_a < u_a &\implies c_a^y = 0
\end{aligned}
$$

Residual Network G^x

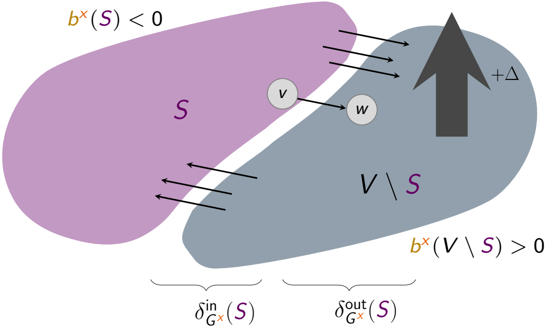


with flow x_a

Invariants of the Algorithm

- x fulfills the **capacity constraints**,
- (y, z) is **dual feasible** and
- **complementary slackness** holds.

Shift Potentials Along a Cut



$$\begin{aligned} \text{maximize } & b^T y - u^T z \\ & y_w - y_v - z_a \leq c_a \\ & z_a \geq 0 \end{aligned}$$

$$c^y_a := c_a + y_v - y_w$$

$$z^y_a := \max\{0, -c^y_a\}$$

$$\Delta := \min_{a=(v,w)\in\delta^{\text{out}}_{G^x}(S)} \{c^y_a\}$$

- this preserves **complementary slackness**
- yields $c^{y'}_a = 0$ for at least one $a \in \delta^{\text{out}}_{G^x}(S)$
- $b^T y' - u^T z^{y'} \geq b^T y - u^T z^y$
- If $b^x(S) \geq 0$, a symmetric approach works.

The Algorithm

Dual Step – Improve Optimality of y

- Construct **nested cuts** $\{s\} = S_1 \subset S_2 \subset \ldots \subset S_n = V$.
- The picked arcs form a **spanning tree** T of G^x with $c_a^{y'} = 0$ for all $a \in T$.

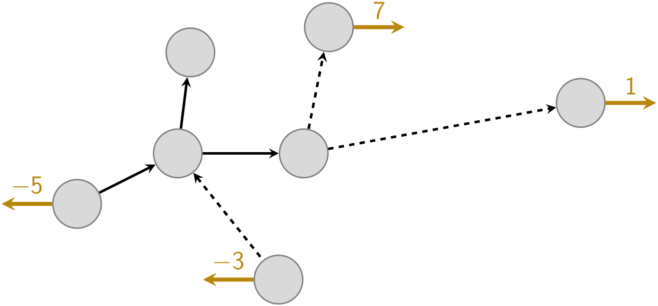
Primal Step – Improve Feasibility of x

- Send maximum flow on T from the sources to the sinks.
- Reduces $\|b^x\|_1$ by at least 2.

Theorem

The algorithm terminates and returns a correct result in $O(\|b\|_1(m + n\log n))$ **time**.

Why Would This Be a Good Idea?

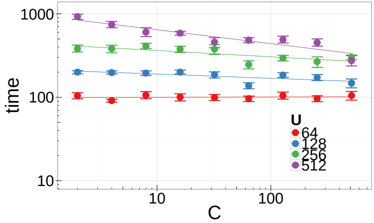


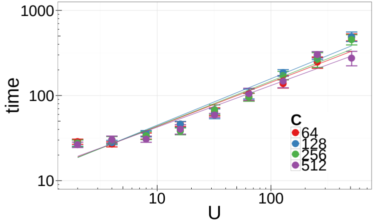
- Classical algorithms like **Successive Shortest Path** send flow along **one path** in one iteration.
- **Our Algorithm** can send flow along **one spanning tree** in one iteration.
- When forming the spanning tree, it takes sign $b^x(S)$ into account.

Experimental Runtime Analysis

Generate 2D Grid Graphs of fixed size

- draw $c \in [-C, C]^m$ and $u \in [0, U]^m$ uniformly at random
- generate **demands** by saturating **negative cost** arcs
- there is a linear dependence between U and $\|b\|_1$





- seems to be **no significant influence of** C on the run-time
- dependence of the run-time proportional to $\|b\|_1^{1/2}$

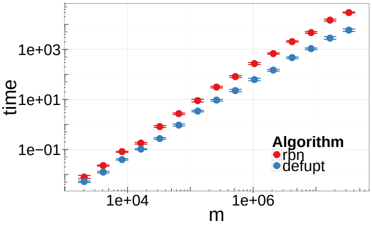
Variants of the Algorithm

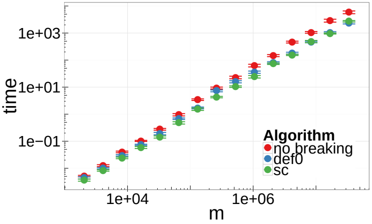
Primal Update

- rbn: max-flow on T from sources to sinks
- defupt: greedily send residual demand up the tree T

Interrupting Dual Step

- def0: interrupt dual step if residual demand of S is zero
- sc: interrupt dual step if sign of residual demand of S changes





Also evaluated: choice of starting node, different priority queues, etc.

Comparison with Other Algorithms

	our	ssp	ns	cos
netgen_sr	1.34 ± 0.02	1.42 ± 0.01	1.25 ± 0.02	$\mathbf{1.17} \pm 0.02$
netgen_8	1.37 ± 0.02	1.80 ± 0.02	1.69 ± 0.02	$\mathbf{1.22} \pm 0.01$
goto_8	1.66 ± 0.04	2.03 ± 0.02	2.17 ± 0.03	$\mathbf{1.53} \pm 0.02$
road_paths	$\mathbf{1.33} \pm 0.03$	1.40 ± 0.04	1.74 ± 0.03	1.50 ± 0.02
road_flow	$\mathbf{1.42} \pm 0.03$	1.43 ± 0.04	1.76 ± 0.04	1.46 ± 0.03

Table : Exponents of the run-time dependence on m.

- The algorithm is better than ns on **all but the dense graph class**.
- It **beats ssp on all instance classes**.
- It **beats all implementations** on the **road instances**.

Thanks for listening!