The Binary Blocking Flow Algorithm

Andrew V. Goldberg
Microsoft Research – Silicon Valley
www.research.microsoft.com/∼goldberg/

Why this Max-Flow Talk?

The result: $O(\min(n^{2/3}, m^{1/2}) m \log(n^2/m) \log(U))$ maximum flow algorithm [Goldberg & Rao 97].

- My first joint work with Bob was on max-flows.
- The result is strong, appropriate for the event.
- Closely related to Bob's work.
  - Motivated by $O(\min(n^{2/3}, m^{1/2}) m)$ unit capacity flow algorithm [Even & Tarjan 75].
  - Uses dynamic trees [Sleator & Tarjan 83].
  - Uses $O(m \log(n^2/m))$ blocking flow algorithm [Goldberg & Tarjan 88].
  - Bob has the best strongly polynomial algorithm [King, Rao & Tarjan 94].
  - Bob teaches the algorithm in his advanced algorithms class.
  - Improved and beautified a part of it [Haeupler & Tarjan 07].

Problem Definition

- Input: Digraph $G = (V, A)$, $s, t \in V$, $u : A \rightarrow [1, \ldots, U]$.
- $n = |V|$ and $m = |A|$.
- Similarity assumption [Gabow 85]: $\log U = O(\log n)$
  For modern machines $\log U$, $\log n \leq 64$.
- The $\tilde{O}\,()$ bound ignores constants, $\log n$, $\log U$.
- Flow $f : A \rightarrow [0, \ldots U]$ obeys capacity constraints and conservation constraints.
- Flow value $|f|$ is the total flow into t.
- Cut is a partitioning $V = S \cup T : s \in S, t \in T$.
- Cut capacity $u(S, T) = \sum_{v \in S, w \in T} u(v, w)$.

Maximum flow problem: Find a maximum flow.
Minimum cut problem (dual): Find a minimum cut.

Time Bounds

year	discoverer(s)	bound	note
1951	Dantzig	$O(n^2mU)$	$\tilde{O}\left(n^2mU\right)$
1955	Ford & Fulkerson	$O(m^2U)$	$\tilde{O}\left(m^2U\right)$
1970	Dinitz	$O(n^2m)$	$\tilde{O}\left(n^2m\right)$
1972	Edmonds & Karp	$O(m^2 \log U)$	$\tilde{O}\left(m^2\right)$
1973	Dinitz	$O(nm \log U)$	$\tilde{O}\left(nm\right)$
1974	Karzanov	$O(n^3)$	
1977	Cherkassky	$O(n^2m^{1/2})$	
1980	Galil & Naamad	$O(nm \log^2 n)$	
1983	Sleator & Tarjan	$O(nm \log n)$	
1986	Goldberg & Tarjan	$O(nm \log(n^2/m))$	
1987	Ahuja & Orlin	$O(nm + n^2 \log U)$	
1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/m))$	
1989	Cheriyan & Hagerup	$E(nm + n^2 \log^2 n)$	
1990	Cheriyan et al.	$O(n^3/\log n)$	
1990	Alon	$O(nm + n^{8/3} \log n)$	
1992	King et al.	$O(nm + n^{2+\epsilon})$	
1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$	
1994	King et al.	$O(nm \log_{m/(n \log n)} n)$	
1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3}m \log(n^2/m) \log U)$	$\tilde{O}\left(m^{3/2}\right)$ $\tilde{O}\left(n^{2/3}m\right)$

blocking flow and push-relabel algorithms.

Background

- Residual capacity $u_f(a) = u(a) - f(a)$ $a \in A$ and $f(a^R)$ o.w.
- Residual graph $G_f = (V, A_f)$ is induced by arcs with positive residual capacity.
- Let $\ell \geq 0$ be a length function on A_f.
- Reduced cost $c_d(v,w) = \ell(v,w) - d(v) + d(w)$.
- Shortest paths w.r.t. ℓ and c_d are the same.
- If $d(t) = 0$ and $c_d \geq 0$, then $d(v) \leq \text{dist}(v,t)$.
- $d(v) = \text{dist}(v,t)$ iff $\exists$ a v-t path of zero reduced cost arcs.
- If $d(v) \geq d(w)$, increasing $f(v,w)$ creates no negative arcs.
- Given f and d, the admissible graph $G_d = (V, A_d)$ is induced by zero reduced cost residual arcs.
- If $(v,w) \in A_d$, then $d(v) \geq d(w)$.
- An s-t flow augmentation in G_d does not decrease $\text{dist}(s,t)$.

Augmenting Path Algorithm

An Augmenting path is an s-t path in G_f.

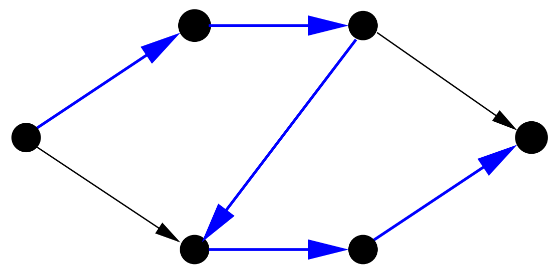
f is optimal iff there is no augmenting path.
Flow augmentation: Given an augmenting path Γ, increase f on all arcs on Γ by the minimum residual capacity of arcs on Γ.
Saturates at least one arc on Γ.

Augmenting path algorithm: While there is an augmenting path, find one and augment.
Runs in $O(m^2 U)$ time.

Unit lengths: $\forall a \in A_f$ let $\ell(a) = 1$.
Augmenting along a shortest path yields a polynomial-time algorithm.

Blocking Flows

f in G is blocking if every s-t path in G is saturated.



- The admissible graph G_d contains all arcs of G_f on s-t shortest paths.
- For unit lengths, G_d is acyclic.
- $O(m \log(n^2/m))$ algorithm to find a blocking flow in an acyclic graph [Goldberg & Tarjan 88].

Blocking flow method:[Dinitz 70]
Repeatedly augment f by a blocking flow in G_f.

Lemma: Each iteration increases the s to t distance in G_f.

$O(nm \log(n^2/m))$ maximum flow algorithm.

Binary Length Function

How does one beat the nm barrier?
[Edmonds & Karp 1972]: general lengths (but no results).
Algorithm intuition [Goldberg & Rao 1997]:

- Capacity-based lengths:
  $\ell(a) = 1$ if $0 < u_f(a) < 2\Delta$, $\ell(a) = 0$ otherwise.
- Maintain residual flow bound F, update when improves by at least a factor of 2.
- Set $\Delta = F/\sqrt{m}$.
- Find a flow of value Δ or a blocking flow; augment.
- After $O(\sqrt{m})$ Δ-augmentations F decreases.
- After $4\sqrt{m}$ blocking flow augmentations, $d(s) \geq 2\sqrt{m}$.
- One of the cuts $(\{d(v) > i\}, \{d(v) \leq i\})$ has no 0-length arcs and at most $\sqrt{m}/4$ length one arcs.
- After $O(\sqrt{m})$ blocking flows F decreases.

Why stop blocking flow computation at Δ value?

Zero Length Arcs

Pros:

- Seem necessary for the result to work.
- Large arcs do not go from hight to low vertex layers.
- Small cut when $d(s) << n$.

Cons:

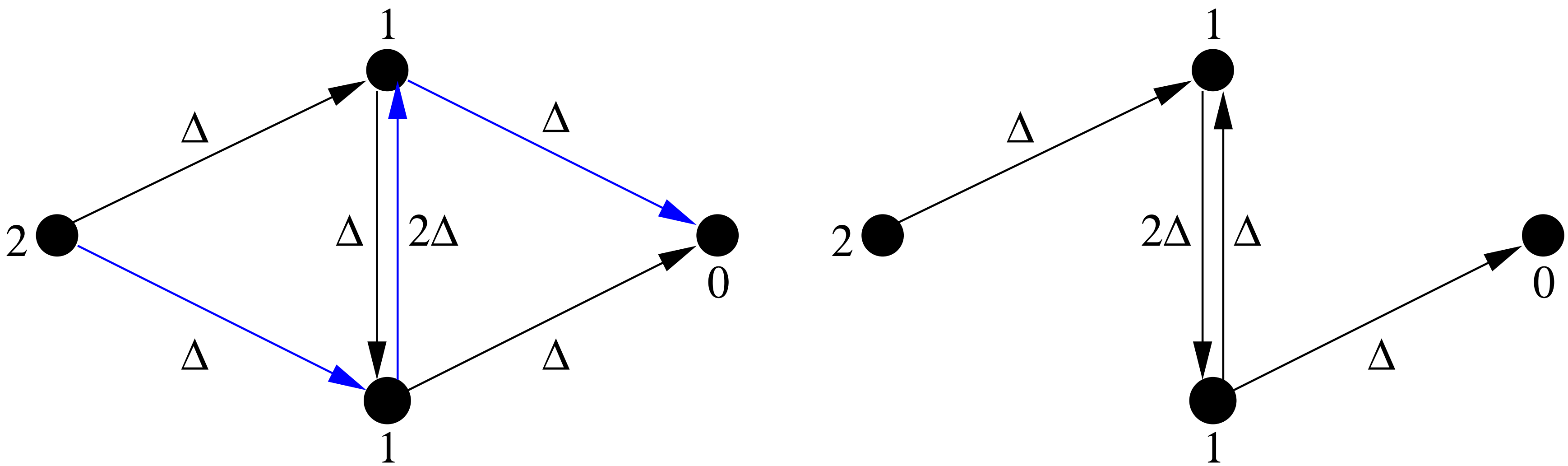
- G_d need not be acyclic.
- Increasing flow in G_d may create new admissible arcs: $d(v) = d(w)$, increasing $f(v, w)$ increases $u_f(w, v)$ from Δ to 2Δ.
- The new arcs are created only if an arc length is reduced to zero.

These problems can be resolved.

Problem: Admissible Cycles

- G_d can have only cycles of zero-length arcs between vertices with the same d.
- These arcs have capacities of at least 2Δ.
- Contract SCCs of G_d to obtain acyclic G'_d.
- Δ flow can be routed in such a strongly connected graph in linear time [Erlebach & Hagerup 02, Haeupler & Tarjan 07].
- Stop a blocking flow computation if the current flow has value Δ.
- After finding a flow in G'_d, extend it to a flow in G_d.
- A blocking flow in G'_d is a blocking flow in G_d.

Problem: Arc Length Decrease



An arc length can decrease from one to zero and s-t distance may not increase.

Special Arcs

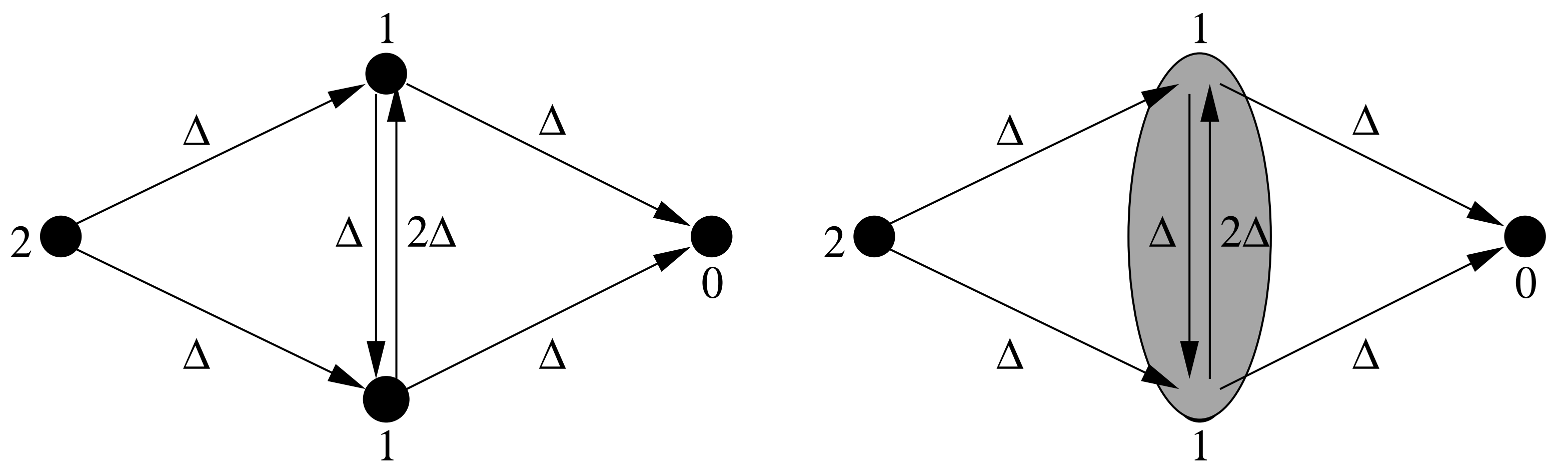
When length decrease on (v, w) can happen and hurt?

1. $\Delta \leq u_f(v, w) < 2\Delta$
2. $d(v) = d(w)$
   - $d(v) > d(w)$: $f(v, w)^R$ not increases, $\ell(v, w)$ not decreases.
   - $d(v) < d(w)$: decreasing $\ell(v, w)$ does not hurt.
3. (optional) $u_f(v, w)^R \geq 2\Delta$

Special arc: Satisfies (1), (2) and optionally (3).

Can reduce special arc length to zero: d does not change, residual capacity large.

Main Loop



- Assign arc lengths, compute distances to t.
- Reduce special arc length to zero.
- Contract SCCs in G_d to obtain G'_d.
- Find a Δ-flow or a blocking flow in G'_d.
- Extend to a flow in G_d, augment.

Main Theorem

Theorem: While F stays the same, d is monotone. In the blocking flow case, $d(s)$ increases.

Proof:

- No negative reduced cost arcs created, d monotone.
- No zero arcs created (special arcs excluded).
- No admissible arcs created w.r.t. new lengths.
- Blocked cut remains blocked after length update.
- $d(s)$ increases by a blocking flow augmentation.

Analysis

$O(\sqrt{m}\log(mU))$ iteration bound is obvious. To do better:

- While $\Delta \geq U$ no zero-length arcs, $d(s)$ monotone.
- After $O(\sqrt{m})$ iterations $F \leq \sqrt{m}U$.
- $O(\sqrt{m})$ iterations reduces F by a factor of two.
- In $O(\sqrt{m}\log U)$ iterations $F \leq \sqrt{m}$.
- Integral flow, an iteration decreases F.
- $O(\sqrt{m}\log U)$ iterations total.
- An iteration is dominated by a blocking flow.
- A slight variation gives an $O(n^{2/3}\log U)$ iteration bound.

Additional Topics

- The new algorithm not as robust as push-relabel in practice...
- ...but outperforms Dinitz' algorithm [Hagerup et al 98].
- Problems extending the bound to the push-relabel method.
- Extends to the augment-relabel method.
- Open problem: extending the bound to min-cost flows.

Push–Relabel Method

Push–relabel algorithms [Goldberg & Tarjan 86] are more practical than blocking flow algorithms.

- Preflow f [Karzanov 1974]: $v \neq s$ may have flow excess $e_f(v)$, but not deficit.
- Distance labeling gives lower bounds on distance to t in G_f. Formally $d : V \rightarrow \mathcal{N}$, $d(t) = 0$, $\forall(v, w) \in G_f$, $d(v) \leq d(w) + 1$.
- Initially $d(v) = 1$ for $v \neq s, t$, $d(s) = n$, arcs out of s are saturated.
- Apply push and relabel operations until none applies.
- Algorithm terminates with a min-cut. Converting preflow into flow is fast.

Push–Relabel (cont.)

- Algorithm updates f and d using push and relabel operations.
- push(v, w): $e_f(v) > 0$, (v, w) admissible.
  Increase $f(v, w)$ by at most $\min(u_f(v, w), e_f(v))$.
- relabel(v): $d(v) < n$, no arc (v, w) is admissible.
  Increase $d(v)$ by 1 or the maximum possible value.
- Current arc data structure: Current arc of v starts at the first arc of v initially and after each relabeling; advances only if the current arc is not admissible.
- Selection rules: Pick the next vertex to process, e.g., FIFO on vertices with excess, highest-labeled vertex with excess.

Can extend the algorithm to the binary lengths, but the improved analysis fails: $\triangle$ flow can move around a cycle, neither flow value nor $d(s)$ increases.

Augment–Relabel Algorithm

Intuitively, push-relabel with DFS operation ordering.

```
FindPath(v)
{
  if (v == t) return(true);
  while (there is an admissible arc (v,w)) {
    if (FindPath(w) {
      v->current = (v,w); return(true);
    }
  }
  relabel(v); return(false);
}
```

The algorithm repeatedly calls `FindPath(s)` and augments along the current arc path from s to t until $d(s) \geq n$.
Can use binary lengths to get the improved bounds.
Does not work well in practice.

Open Problem

Min-cost flow algorithms:

- For unit lengths, max-flow + cost-scaling = min-cost flow with $\log(nC)$ slowdown, where C is the maximum arc length.
- For unit capacities, [Gabow & Tarjan 87] give an $O(\min(n^{2/3}m^{1/2})m\log(nC))$ algorithm.
- For min-cost flows with integral data, is there an $O(\min(n^{2/3}m^{1/2})m\log(nC)\log U)$ algorithm?
- ...or a more modest $\tilde{O}\left(n^{1-\epsilon}m\right)$ algorithm for $\epsilon > 0$?

Thank You!