

Maximum Flows by Incremental Breadth-First Search

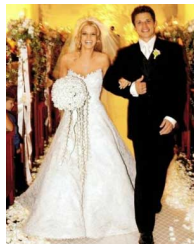
Andrew V. Goldberg¹ Sagi Hed Haim Kaplan Robert
E. Tarjan Renato F. Werneck

Microsoft Research – Silicon Valley
<http://research.microsoft.com/~goldberg/>

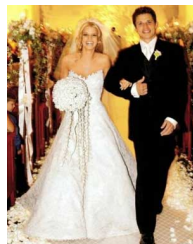
Outline

- 1 Introduction
- 2 Background
- 3 Boykov-Kolmogorov (BK) Algorithm
- 4 Incremental Breadth-First Search (IBFS)
- 5 Experimental Results
- 6 Concluding Remarks

Theory vs. Practice



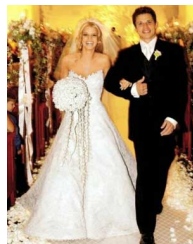
Theory vs. Practice



Scientific Method

- Make an observation
- Form a hypothesis or theory
- Make a prediction
- Verify by experiment

Theory vs. Practice



Scientific Method

- Make an observation
- Form a hypothesis or theory
- Make a prediction
- Verify by experiment

Design-analysis-experimentation loop.

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$
- **Cut:** Partitioning $V = (S, T) : s \in S, t \in T$

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$
- **Cut:** Partitioning $V = (S, T) : s \in S, t \in T$
- **Cut capacity:** $u(S, T) = \sum_{v \in S, w \in T, (v,w) \in E} u(v, w)$

Flows and Cuts

Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$
- **Cut:** Partitioning $V = (S, T) : s \in S, t \in T$
- **Cut capacity:** $u(S, T) = \sum_{v \in S, w \in T, (v,w) \in E} u(v, w)$
- **Max-flow problem:** find f maximizing $|f|$

Flows and Cuts

Definitions

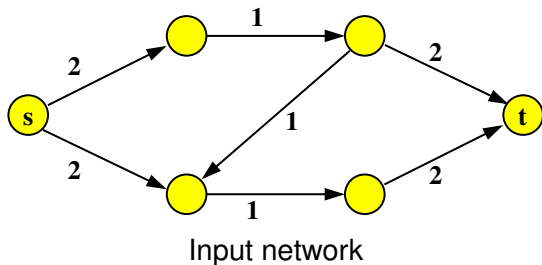
- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$
- **Cut:** Partitioning $V = (S, T) : s \in S, t \in T$
- **Cut capacity:** $u(S, T) = \sum_{v \in S, w \in T, (v,w) \in E} u(v, w)$
- **Max-flow problem:** find f maximizing $|f|$
- **Min-cut problem:** find (S, T) minimizing $u(S, T)$

Flows and Cuts

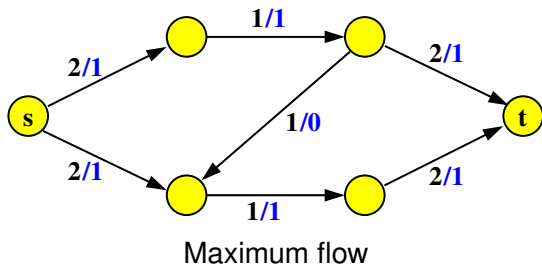
Definitions

- **Input:** Graph $G = (V, E)$, $s, t \in V$, $u : E \rightarrow \mathcal{R}^+$
- **Flow:** $f : E \rightarrow \mathcal{R}^+$ satisfying **capacity constraints**
 $\forall a \quad f(a) \leq u(a)$ and **conservation constraints**
 $\forall v \neq s, t \quad \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w)$
- **Flow value:** $|f| = \sum_{(s,v) \in E} f(s, v) \quad (= \sum_{(v,t) \in E} f(v, t))$
- **Cut:** Partitioning $V = (S, T) : s \in S, t \in T$
- **Cut capacity:** $u(S, T) = \sum_{v \in S, w \in T, (v,w) \in E} u(v, w)$
- **Max-flow problem:** find f maximizing $|f|$
- **Min-cut problem:** find (S, T) minimizing $u(S, T)$
- **Duality theorem:** (maximum $|f|$) = (minimum $u(S, T)$)

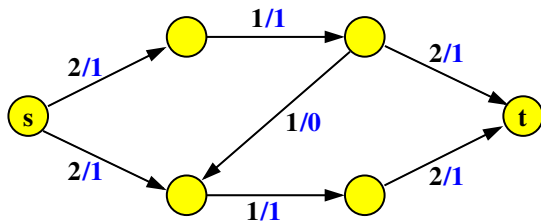
Example



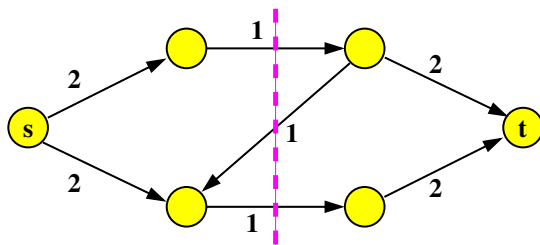
Example



Example



Maximum flow



Minimum cut

Outline

- 1 Introduction
- 2 Background**
- 3 Boykov-Kolmogorov (BK) Algorithm
- 4 Incremental Breadth-First Search (IBFS)
- 5 Experimental Results
- 6 Concluding Remarks

Residual Graph

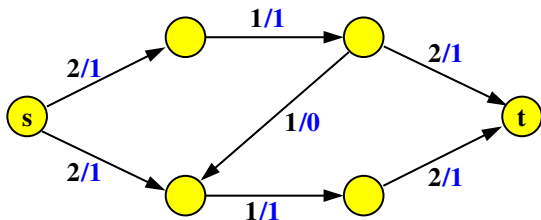
Definition: residual capacity

- For $(v, w) \in E$, $u_f(v, w) = u(v, w) - f(v, w)$
- For $(w, v) \in E$, $u_f(v, w) = f(w, v)$
- **Residual graph** G_f is induced by $(v, w) \in E \cup E^R : u_f(v, w) > 0$

Residual Graph

Definition: residual capacity

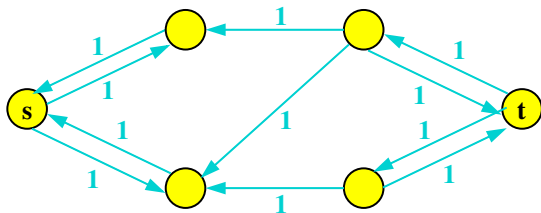
- For $(v, w) \in E$, $u_f(v, w) = u(v, w) - f(v, w)$
- For $(w, v) \in E$, $u_f(v, w) = f(w, v)$
- **Residual graph** G_f is induced by $(v, w) \in E \cup E^R : u_f(v, w) > 0$



Residual Graph

Definition: residual capacity

- For $(v, w) \in E$, $u_f(v, w) = u(v, w) - f(v, w)$
- For $(w, v) \in E$, $u_f(v, w) = f(w, v)$
- Residual graph G_f is induced by $(v, w) \in E \cup E^R : u_f(v, w) > 0$

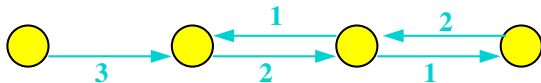


Augmenting Path Algorithm

- **Augmenting path** is an s - t path in G_f
- **Augmentation** operation increases flow on an augmenting path
- Flow remains valid, at least one arc is **saturated**

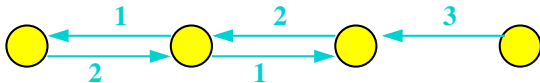
Augmenting Path Algorithm

- **Augmenting path** is an $s-t$ path in G_f
- **Augmentation** operation increases flow on an augmenting path
- Flow remains valid, at least one arc is **saturated**



Augmenting Path Algorithm

- **Augmenting path** is an s - t path in G_f
- **Augmentation** operation increases flow on an augmenting path
- Flow remains valid, at least one arc is **saturated**



Augmenting Path Algorithm

- **Augmenting path** is an s - t path in G_f
- **Augmentation** operation increases flow on an augmenting path
- Flow remains valid, at least one arc is **saturated**



Theorem [Ford & Fulkerson 56]

Flow f is maximum iff G_f has no augmenting path

Augmenting Path Algorithm

- **Augmenting path** is an s - t path in G_f
- **Augmentation** operation increases flow on an augmenting path
- Flow remains valid, at least one arc is **saturated**



Theorem [Ford & Fulkerson 56]

Flow f is maximum iff G_f has no augmenting path

Augmenting path algorithm

While there is an augmenting path, find one and augment on it

Polynomial-Time Algorithms

- Algorithms studied for over 50 years
- Efficient algorithms (theory and practice)
- Progress still being made

Polynomial-Time Algorithms

- Algorithms studied for over 50 years
- Efficient algorithms (theory and practice)
- Progress still being made

Efficient algorithms

- **Shortest path augmentation** [Dinic 70, Edmonds & Karp 72] (BFS)
- Blocking flow method [Dinic 70, Karzanov 74]
- Push-relabel method [Goldberg & Tarjan 86]
- Binary blocking flow method [Goldberg & Rao 97]
 $O(\min(n^{2/3}m^{1/2})m \log(n^2/m) \log U)$ time

Polynomial-Time Algorithms

- Algorithms studied for over 50 years
- Efficient algorithms (theory and practice)
- Progress still being made

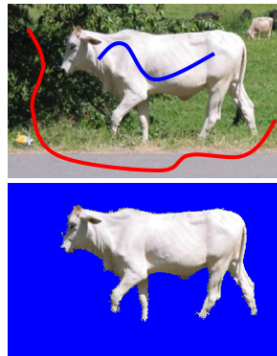
Efficient algorithms

- **Shortest path augmentation** [Dinic 70, Edmonds & Karp 72] (BFS)
- Blocking flow method [Dinic 70, Karzanov 74]
- Push-relabel method [Goldberg & Tarjan 86]
- Binary blocking flow method [Goldberg & Rao 97]
 $O(\min(n^{2/3}m^{1/2})m \log(n^2/m) \log U)$ time

Worst-case bounds usually too pessimistic in real life

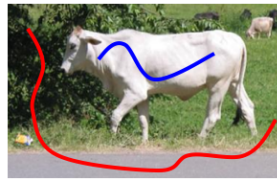
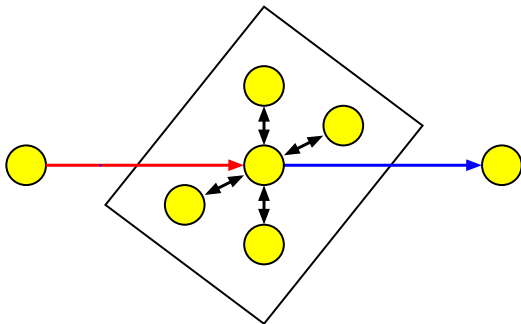
Computer Vision Applications

- Image segmentation
- Multiview reconstruction
- Stereo vision
- Surface fitting
- **Special structure**



Computer Vision Applications

- Image segmentation
- Multiview reconstruction
- Stereo vision
- Surface fitting
- **Special structure**



Outline

- 1 Introduction
- 2 Background
- 3 Boykov-Kolmogorov (BK) Algorithm**
- 4 Incremental Breadth-First Search (IBFS)
- 5 Experimental Results
- 6 Concluding Remarks

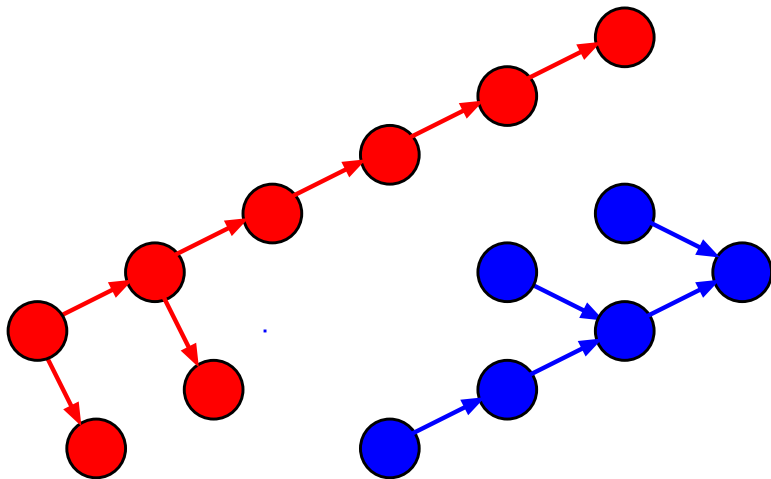
Boykov-Kolmogorov Algorithm

- Excellent performance on many vision applications
- Widely used in vision applications
- No polynomial-time bound, not robust

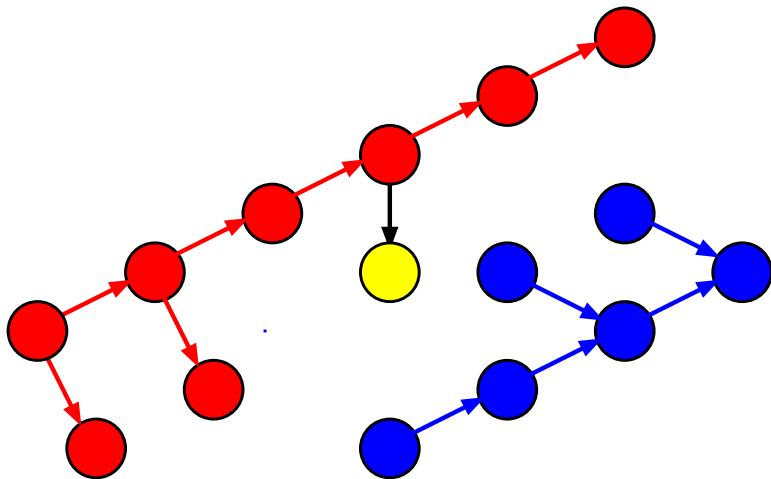
BK algorithm outline

- Augmenting path algorithm
- Maintains residual trees: out-tree S from s and in-tree T to t
- Every vertex is in S , in T , or free
- Initially $S = \{s\}$, $T = \{t\}$, other vertices free
- Three phases: **growth**, **augmentation**, and **adoption**
- Termination: S and T cannot grow, no augmentation possible

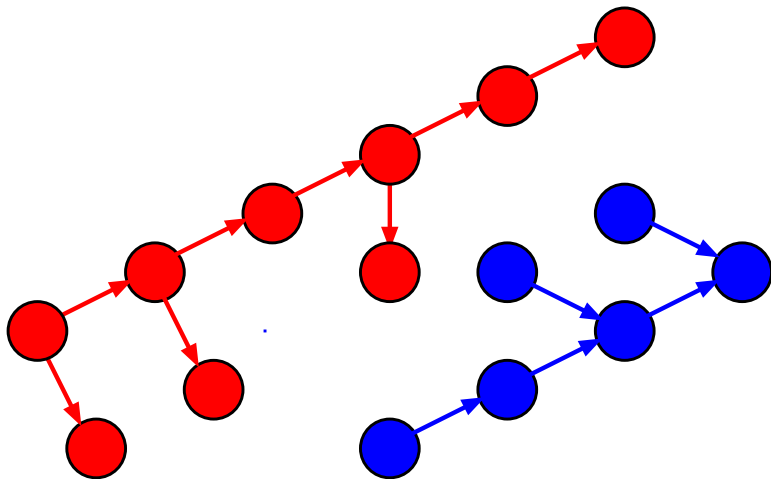
BK Growth Phase



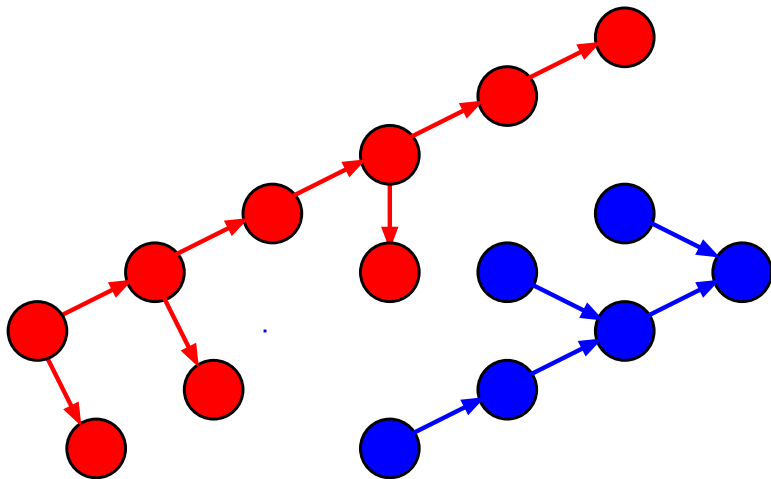
BK Growth Phase



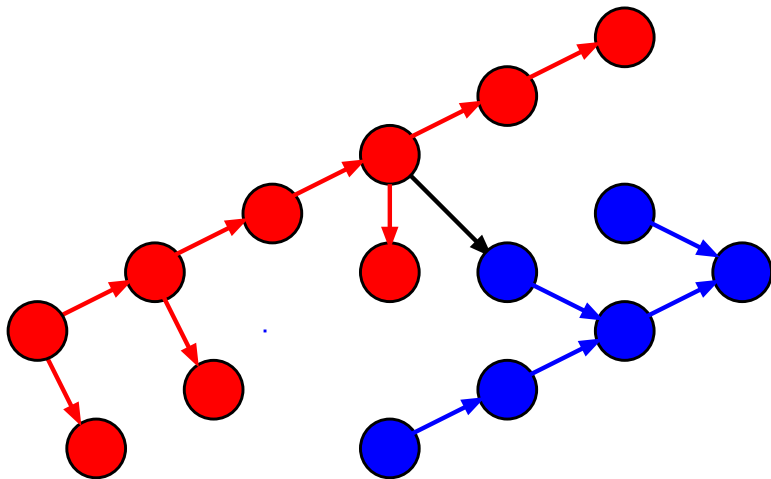
BK Growth Phase



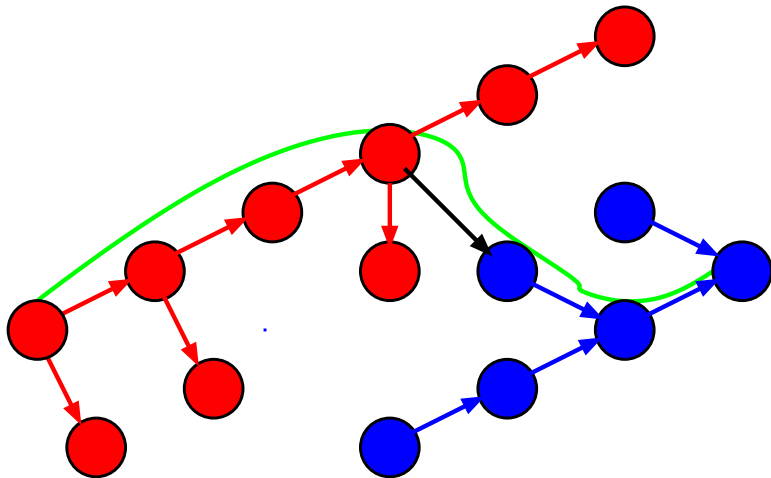
BK Augmentation Phase



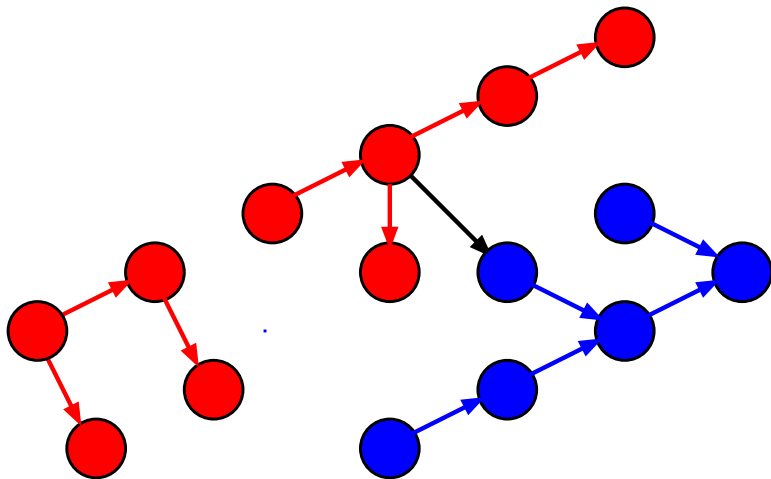
BK Augmentation Phase



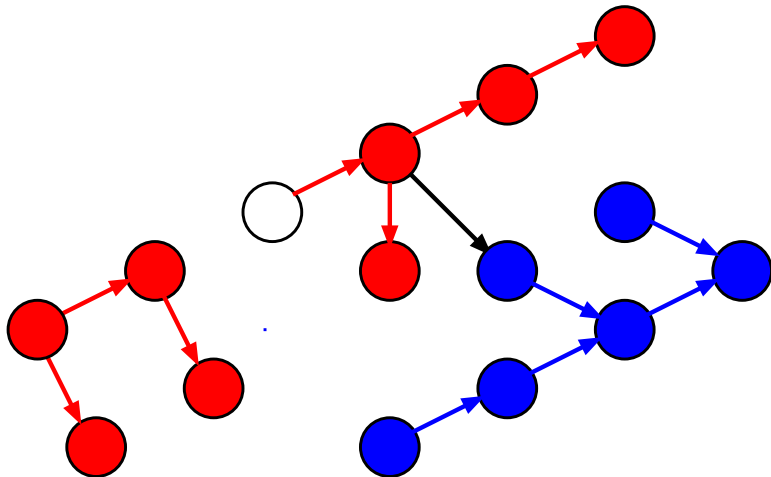
BK Augmentation Phase



BK Augmentation Phase

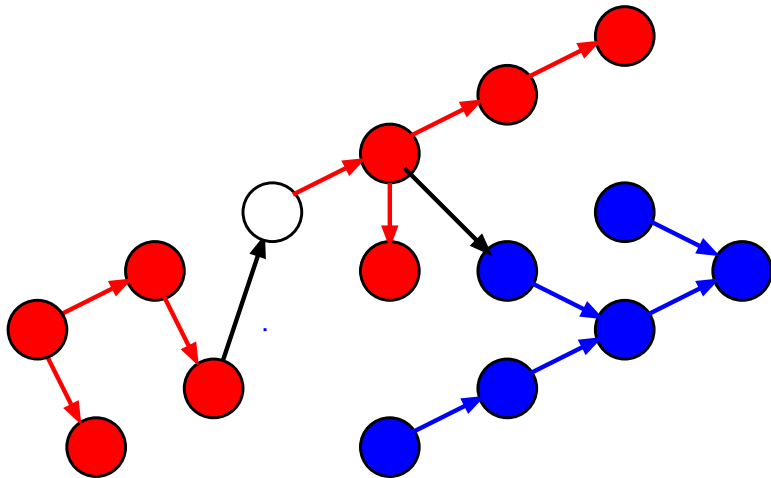


BK Adoption Phase



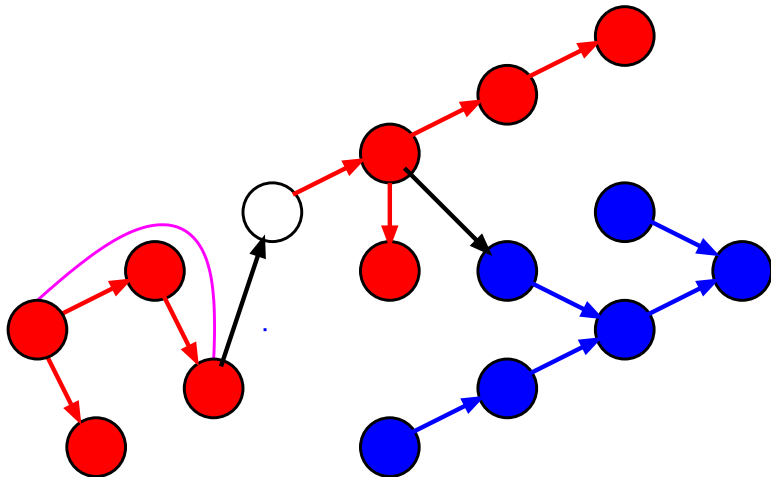
Node with no parent (orphan)

BK Adoption Phase



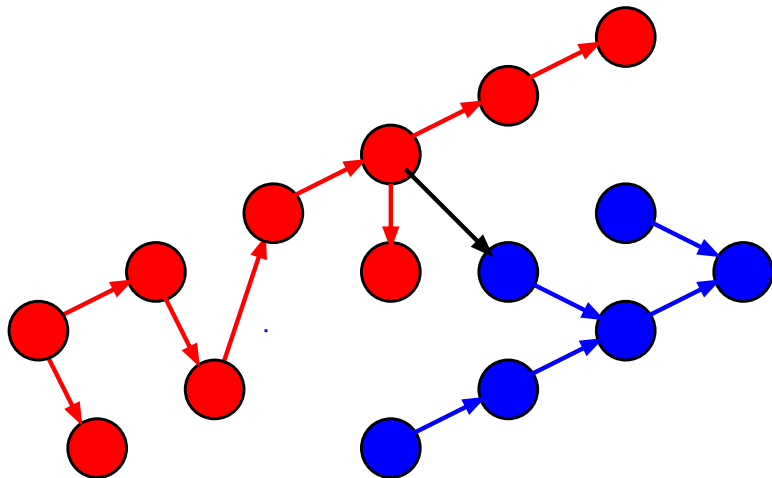
Potential parent

BK Adoption Phase



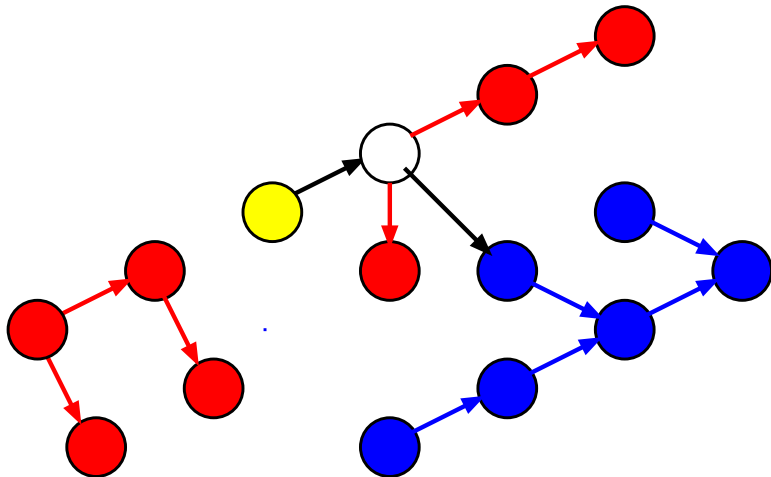
Parent test

BK Adoption Phase



Successful reconnection

BK Adoption Phase



Failure to reconnect

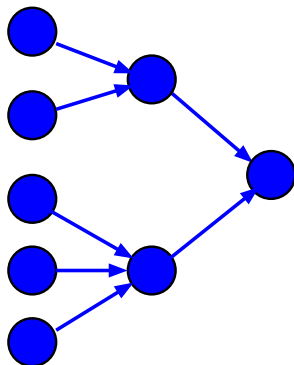
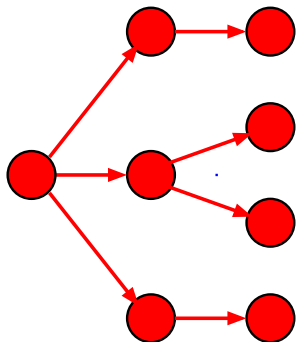
Outline

- 1 Introduction
- 2 Background
- 3 Boykov-Kolmogorov (BK) Algorithm
- 4 Incremental Breadth-First Search (IBFS)**
- 5 Experimental Results
- 6 Concluding Remarks

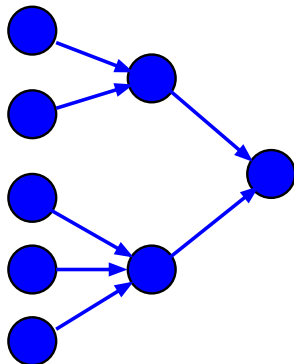
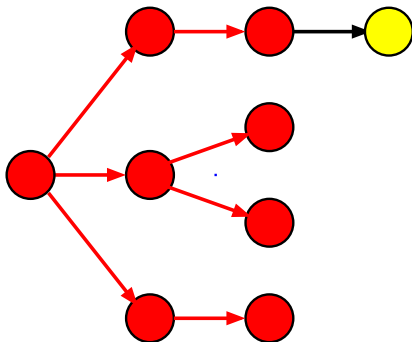
Incremental Breadth-First Search (IBFS)

- Combines BK and shortest path augmentation ideas
- Distances from s and to t are monotone
- Maintain S and T as **BFS trees**
- S (T) contains all vertices at distance up to D_S from s (D_T to t)
- For $v \in S$ ($v \in T$), $d(v)$ is the distance from s (to t)
- Invariant: augmenting path length is at least $L = D_S + D_T + 1$
- A pass grows S (or T) and augments on length L paths
- D_S (or D_T) and L increase after a pass
- Growth, augment, and (modified) adoption phases
- Adoption phase uses **relabel** operation for efficiency
- Initially $S = \{s\}$, $T = \{t\}$, $D_S = 0$, $D_T = 0$, $L = 1$

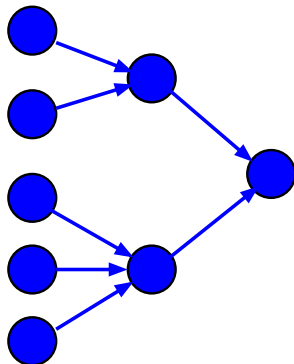
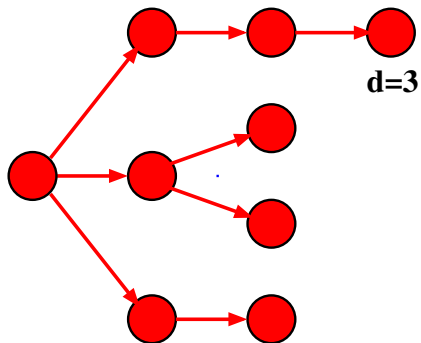
IBFS Growth Phase



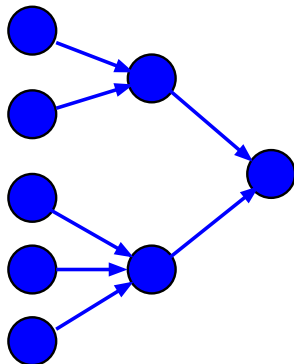
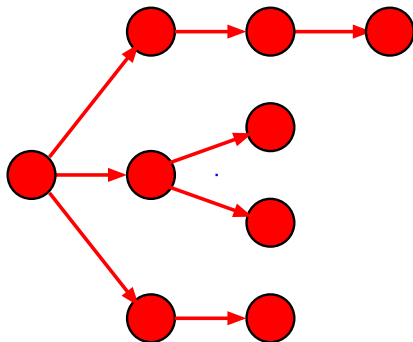
IBFS Growth Phase



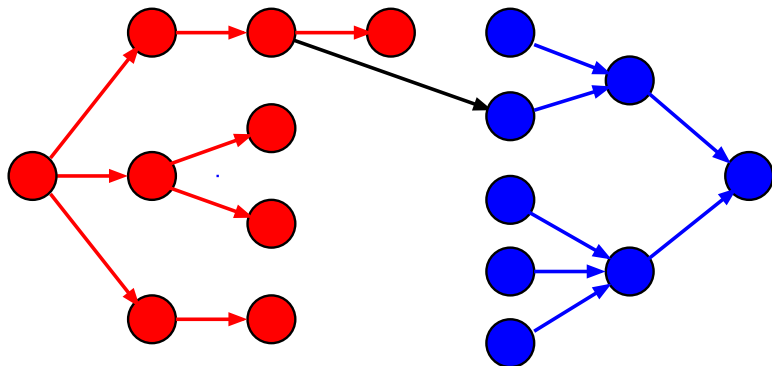
IBFS Growth Phase



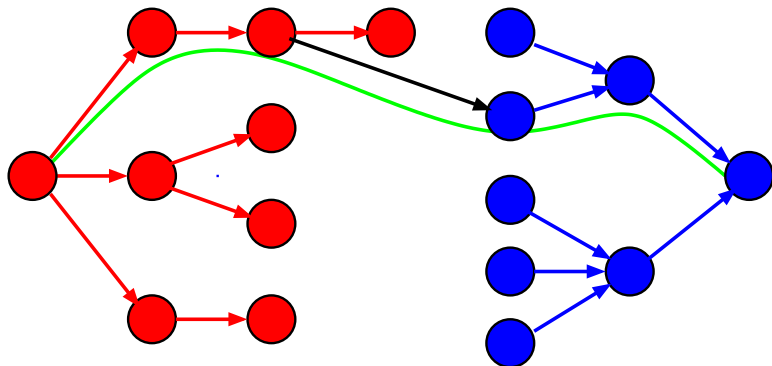
IBFS Augmentation Phase



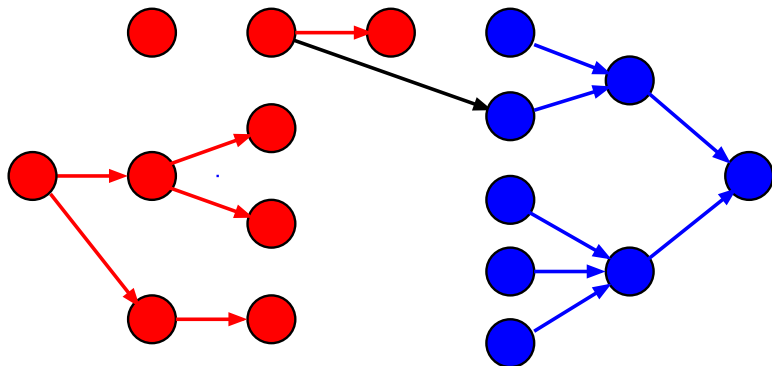
IBFS Augmentation Phase



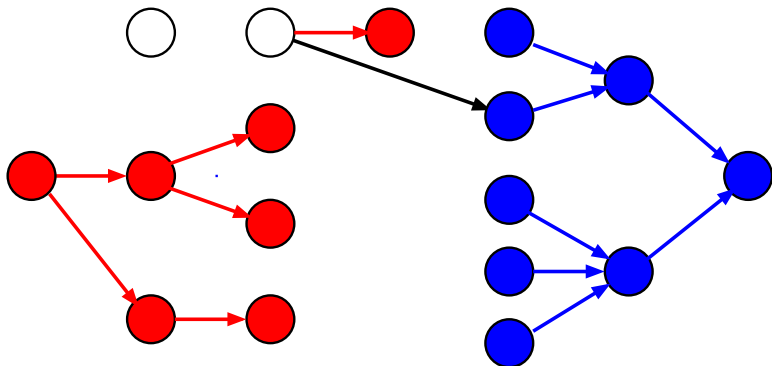
IBFS Augmentation Phase



IBFS Augmentation Phase

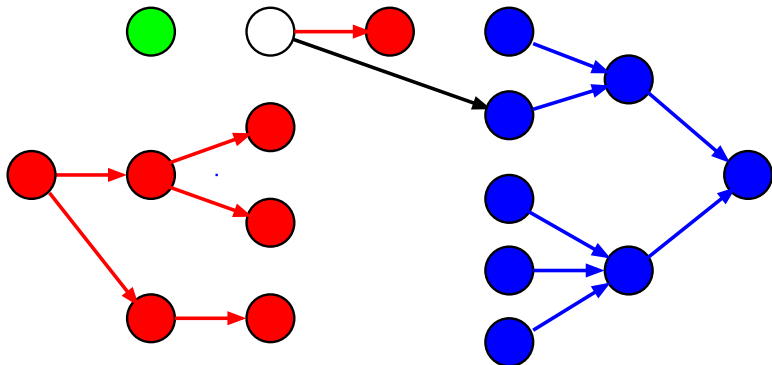


IBFS Adoption Phase



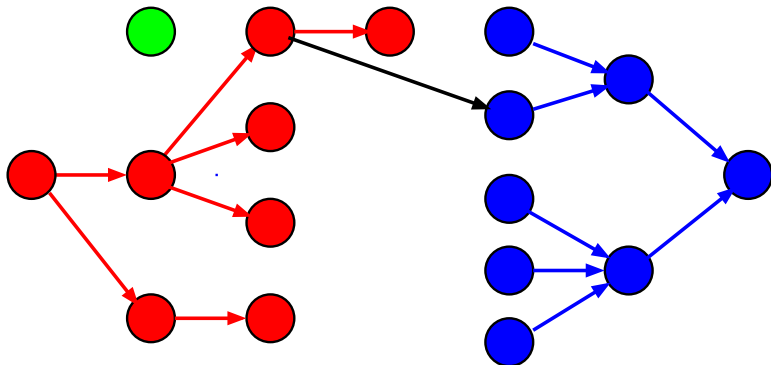
Pick an orphan with the smallest d ; no parent test needed

IBFS Adoption Phase



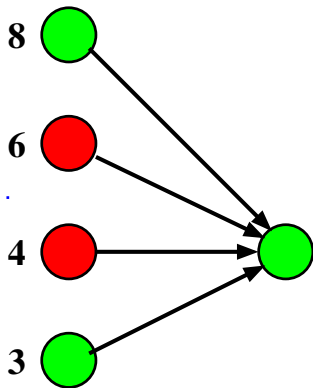
Failed to connect at the same level; **no valid parent**

IBFS Adoption Phase



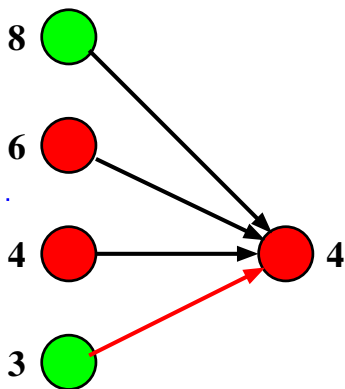
Successful reconnection at the same level

Orphan Relabeling



S-nodes, *S*-orphans (no parent)

Orphan Relabeling



Relabeled vertex becomes S -vertex with $d(v) \leq D_S + 1$, or free

Analysis

- Similar to push-relabel algorithm
- Distances are monotone
- After complete scan of v 's arc list by relabel, $d(v)$ increases
- $d(v) < n$
- Work charged to distance increases
- $O(nm)$ work for relabel, $O(n^2m)$ total
- Can be improved but makes the algorithm more complicated
- We implement the simple version

Outline

- 1 Introduction
- 2 Background
- 3 Boykov-Kolmogorov (BK) Algorithm
- 4 Incremental Breadth-First Search (IBFS)
- 5 Experimental Results**
- 6 Concluding Remarks

Experimental Results

UBK: BK with low-level optimizations, $\approx 20\%$ faster.

Experimental Results

UBK: BK with low-level optimizations, $\approx 20\%$ faster.

NAME	n[M]	ibfs[s]	ubk[s]	speedup
segmentation				
bone_subx100	3.9	3.30	5.32	1.6
liver100	4.2	6.62	14.21	2.2
bone100	7.8	7.01	5.56	0.8
multiview				
gargoyle-sml	1.1	0.89	8.56	9.6
gargoyle-med	8.8	22.58	139.06	6.2
camel-sml	1.2	0.84	1.31	1.6
camel-med	9.7	21.00	32.33	1.5
stereo				
BVZ-sawtooth	—	0.70	0.84	1.2
BVZ-venus	—	1.06	1.19	1.1
KZ2-sawtooth	—	1.68	2.49	1.5
KZ2-venus	—	2.98	4.14	1.4

Experimental Results (cont)

NAME	n[M]	ibfs[s]	ubk[s]	speedup
surf. fitting				
bunny-med	6.3	1.04	1.28	1.2
scribbles segm.				
diggedshweng	0.3	0.42	1.26	3.0
hessi1a	0.5	5.81	6.43	1.1
monalisa	0.8	2.92	4.33	1.5
house	1.0	2.54	3.16	1.2
DIMACS (hard)				
rmf-wide-14	0.017	0.17	0.57	3.4
rmf-wide-16	0.065	2.06	13.22	6.4
rmf-wide-18	0.259	25.37	641.83	25.3

Discussion

- Most problems easy, 10's or 100's scans per arc.
- Total augmenting path length is much smaller for IBFS
- The number of growth steps is lower for IBFS
- BK performs fewer adaption steps
- BK needs to perform orphan tests
- IBFS is more robust
- IBFS appears faster on vision instances
- More testing is needed
- Delayed variant of IBFS needs to be evaluated

Concluding Remarks

- Theory and practice go well together
- IBFS interesting on its own, practical
- More test problems needed
- Further improvements may be possible



Thank You



Code available: <http://www.cs.tau.ac.il/~sagihed/ibfs/>
Publications: <http://research.microsoft.com/~goldberg/>