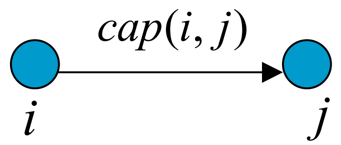
Introduction to Min-Cut/Max-Flow Algorithms

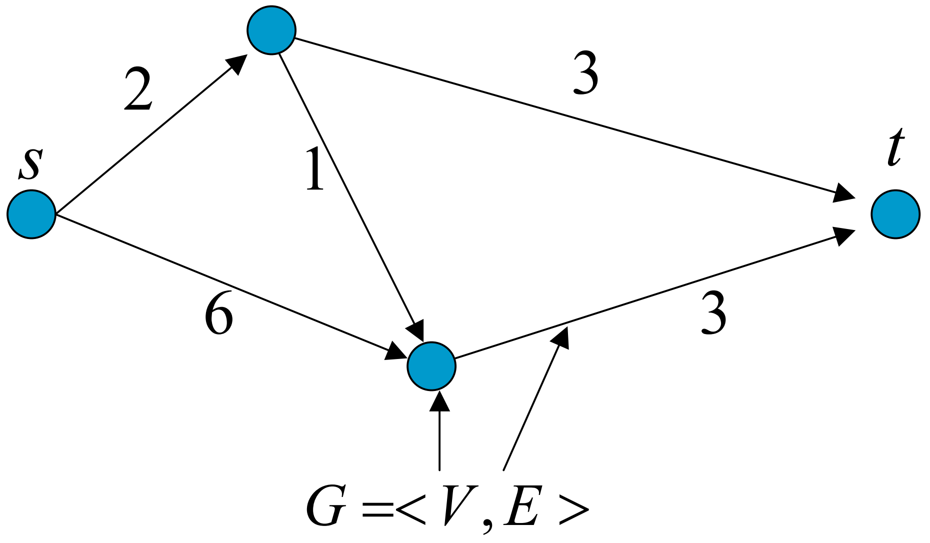
Outline

- Introduction to graph,flow and cut
- The equivalence of max-flow/min-cut
- Maximum flow algorithms

s-t Graph

- A source node and a sink node
- Directed edge (Arc) <i,j> from node i to node j
- Each arc <i,j> has a nonnegative capacity cap(i,j)
- $cap(i,j)=0$ for non-exist arcs





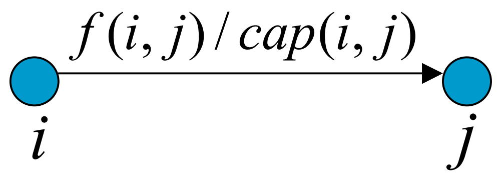
Flow in s-t Graph

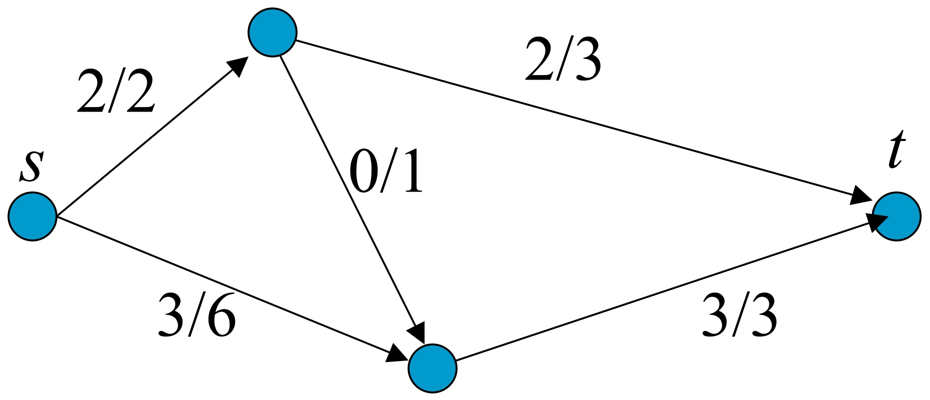
- Flow is a real value function f that assign a real value $f(i,j)$ to each arc <i,j> under :
  - Capacity constraint : $f(i,j) \le cap(i,j)$
  - Mass balance constraint:

$$\sum_{<i,j>\in E} f(i,j) - \sum_{<k,i>\in E} f(k,i) = \begin{cases} 0 & i \in V - \{s,t\} \\ |f| & i = s \\ -|f| & i = t \end{cases}$$

$|f|$ is the value of flow f

Flow in s-t Graph



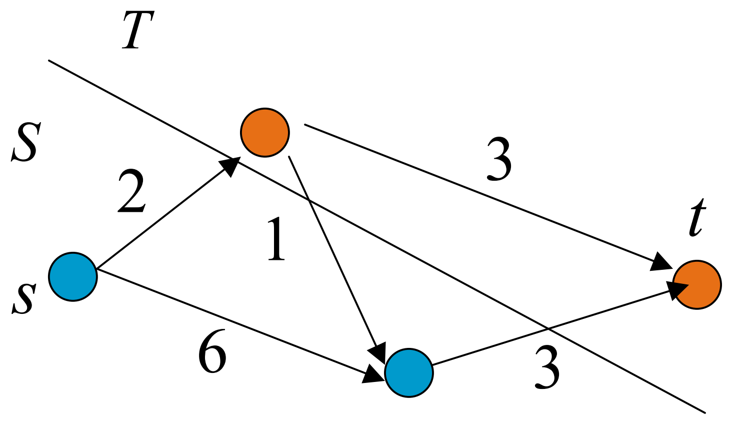


An example of flow

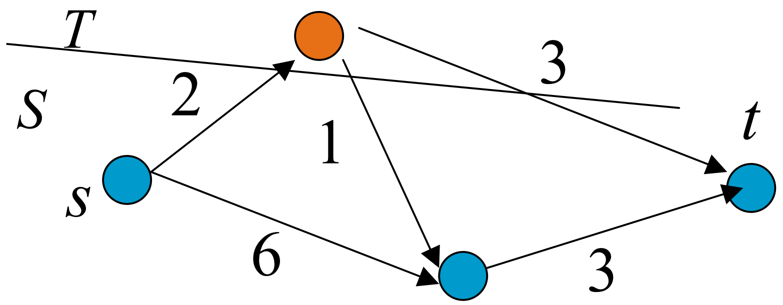
- maximum flow is the flow has maximum value among all possible flow functions

s-t Cut

- A cut is a partition of node set V which has two subsets S and T
- A cut is a s-t cut iif $s \in S, t \in T$



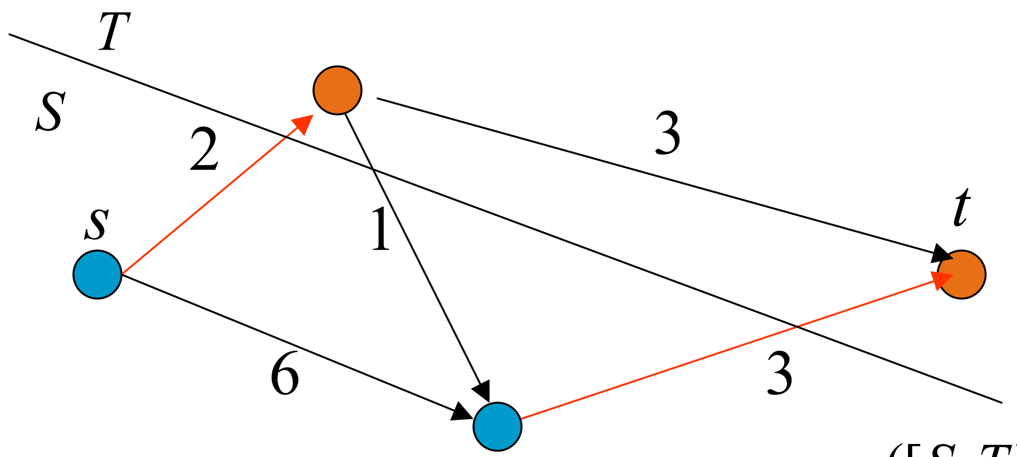
s-t cut



not a s-t cut

Capacity of s-t Cut (cost)

- $$cap([S,T]) = \sum_{<i,j>\in E, i\in S, j\in T} cap(i,j)$$

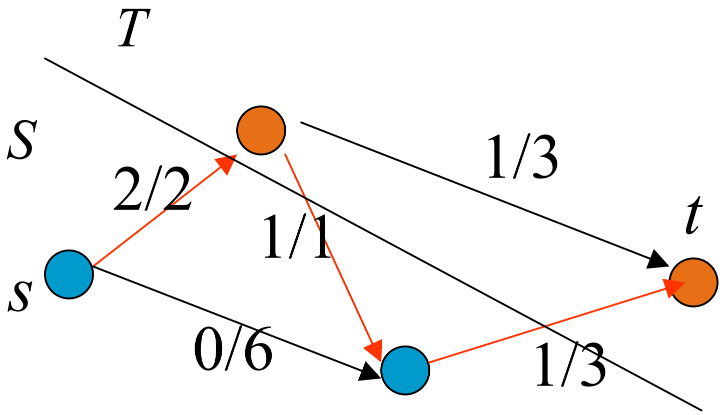


$$cap([S,T]) = 2 + 3 = 5$$

- Minimum cut is the s-t cut whose capacity is minimum among all possible s-t cuts

Flow Across s-t Cut

- $$f([S,T]) = \sum_{<i,j>\in E, v_i \in S, v_j \in T} f(i,j) - \sum_{<j,i>\in E, v_i \in S, v_j \in T} f(j,i)$$



$$f([S,T]) = 2 + 1 - 1 = 2$$

Properties I

- For any s-t cut and flow, the capacity of s-t cut is the upper-bound of the flow across the s-t cut

$$
\begin{aligned}
& cap([S,T]) \\
&= \sum_{i\in S,<i,j>\in E,j\in T} cap(i,j) \\
&\geq \sum_{i\in S,<i,j>\in E,j\in T} f(i,j) \\
&\geq \sum_{i\in S,<i,j>\in E,j\in T} f(i,j) - \sum_{i\in S,<j,i>\in E,j\in T} f(j,i) \\
&= f([S,T])
\end{aligned}
$$

Properties II

- For any s-t cut, the value of flow across the cut equals the value of the flow

$$
\begin{aligned}
|f| &= \sum_{i\in S}\Big[\sum_{<i,j>\in E} f(i,j) - \sum_{<j,i>\in E} f(j,i)\Big] \\
&= \sum_{i\in S}\Big[\sum_{<i,j>\in E,\, j\in S} f(i,j) + \sum_{<i,j>\in E,\, j\in T} f(i,j) - \sum_{<j,i>\in E,\, j\in S} f(j,i) - \sum_{<j,i>\in E,\, j\in T} f(j,i)\Big] \\
&= \sum_{i\in S,<i,j>\in E,\, j\in S} f(i,j) + \sum_{i\in S,<i,j>\in E,\, j\in T} f(i,j) - \sum_{i\in S,<j,i>\in E,\, j\in S} f(j,i) - \sum_{i\in S,<j,i>\in E,\, j\in T} f(j,i) \\
&= \sum_{i\in S,<i,j>\in E,\, j\in T} f(i,j) - \sum_{i\in S,<j,i>\in E,\, j\in T} f(j,i) \\
&= f([S,T])
\end{aligned}
$$

Properties III

- For any s-t cut and any flow, the capacity of s-t cut is the upper-bound of value of the flow

$$\because f([S,T]) \leq cap([S,T])$$

$$\because f([S,T]) = |f|$$

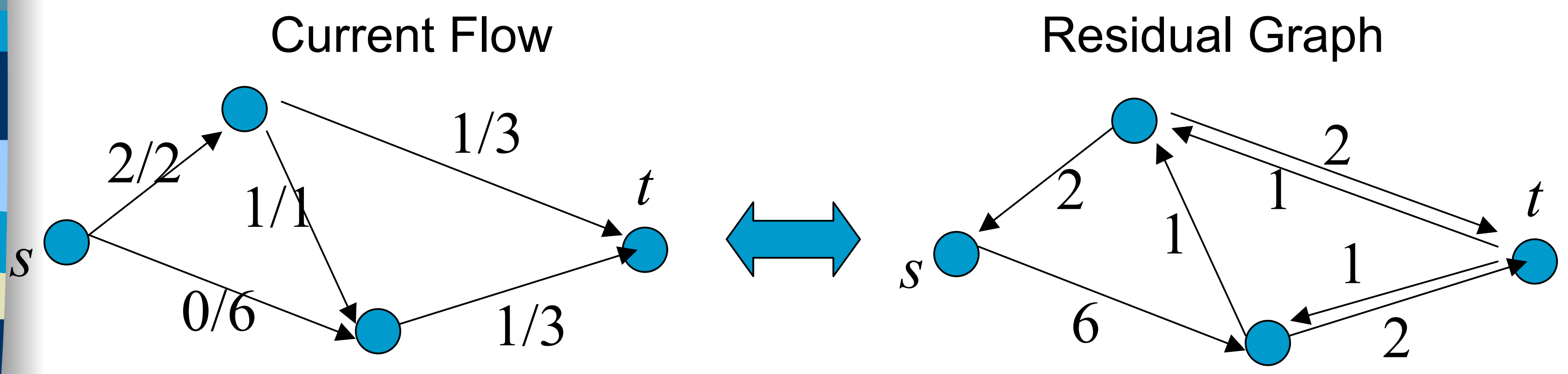
$$\therefore |f| \leq cap([S,T])$$

Outline

- Generic frame work of Graph cut
- Introduction to graph,flow and cut
- The equivalence of max-flow/min-cut
- Maximum flow algorithms

Residual Graph G_r

- The non-negative value $r(i, j)$ means maximum additional flow on the arc <i,j>
- Add flow in inversed arc means reduce flow in current arc
- From flow to residual graph
  - Deduct flow from capacity
  - Add inverse arc of current flow



Equivalence of Maximum Flow and Minimum Cut

- Theorem: If f is a flow function of a s-t graph,then the follows statements are equivalent:
  - A. f is a maximum flow
  - B. there is a s-t cut that it’s capacity equals to the value of f
  - C. The residual graph contains no directed path from source to sink

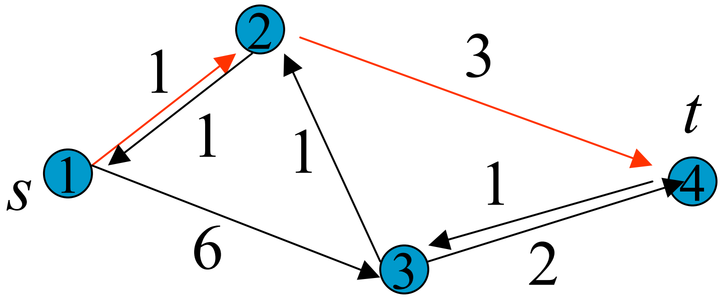
Equivalence of Maximum Flow and Minimum Cut

- If B: there is a s-t cut that it's capacity equals to the value of f
- Then A: f is a maximum flow
  - The capacity of any s-t cut is upper bound of the value of any flow. So if there is a s-t cut that its capacity equals to the value of flow f, then the value of f is also the upper bound of any flow. That means f is a maximum flow.

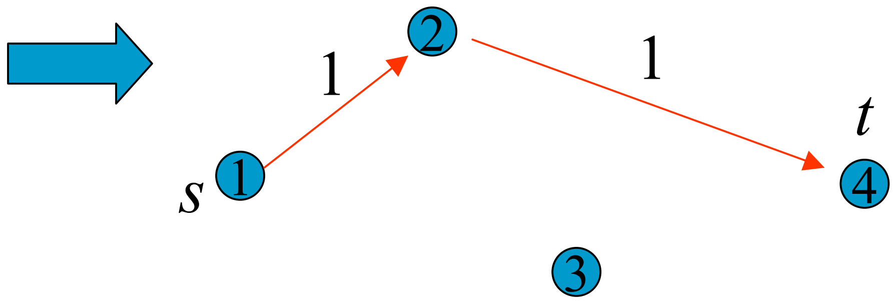
Equivalence of Maximum Flow and Minimum Cut

- If A: f is a maximum flow
- Then C: The residual graph contains no directed path from source to sink
  - If has an path from source to sink, we augment the flow of this path by an amount equal to the minimum residual capacity of the edges along this path. So the original flow f is not a maximum flow. So if f is a maximum flow, then no augmenting path can exist.

Residual Graph

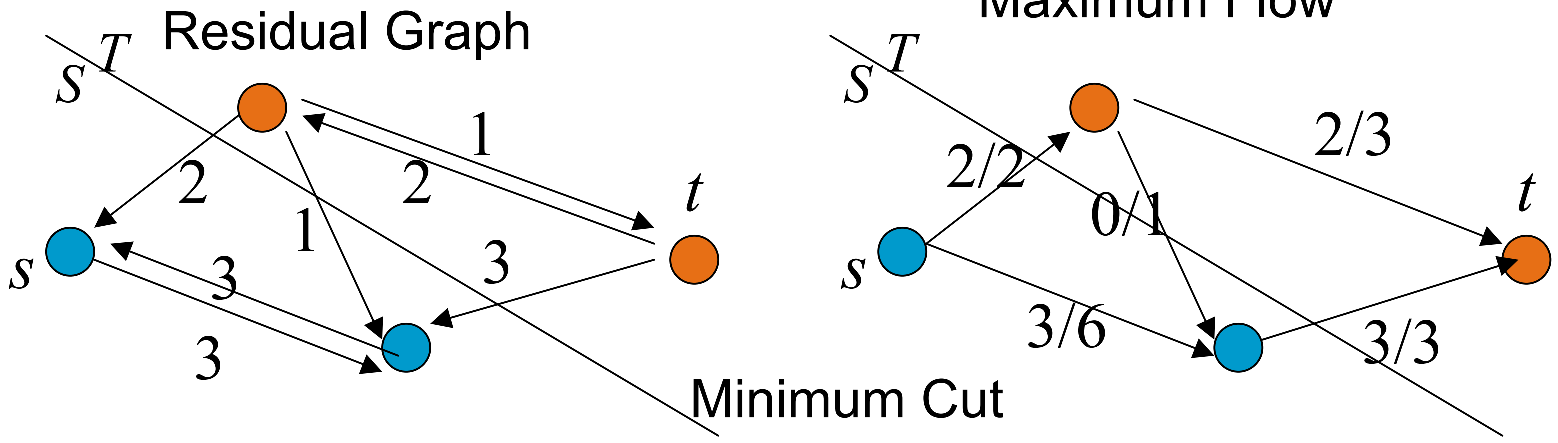


Additional flow



Equivalence of Maximum Flow and Minimum Cut

- If C: The residual graph contains no directed path from source to sink
- Then B: there is a s-t cut that it's capacity equals to the value of f
  - From the source, the residual graph can be separate into reachable S and non-reachable T, it must be a s-t cut. And there is no arc from S to T in the residual graph. So f(i,j)=cap(i,j) for any arc from S to T and f(j,i)=0 for arc from T to S. In this case, cap([S,T])=f([S,T]), as we know f([S,T])=|f|, so cap([S,T]) = |f|



Outline

- Generic framework of Graph cut
- Introduction of graph,flow and cut
- The equivalence of max-flow/min-cut
- Maximum flow algorithms

Maximum Flow Algorithms

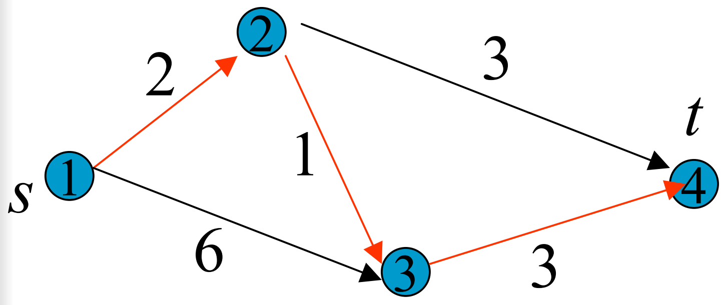
- Augmenting Path (Ford-Fulkerson)
  - ford-fulkerson $O(nmU)$
  - Capacity scaling $O(nm \log U)$
  - Successive shortest path $O(n^2 m)$
  - Layered Network $O(n^2 m)$
  - .....
- Push/Relabel
  - Push/relabel $O(n^2 m)$
  - FIFO preflow-push $O(n^3)$
  - Highest-label preflow-push $O(n^2 \sqrt{m})$
  - .....

Augmenting Path Algorithm (Ford-Fulkerson)

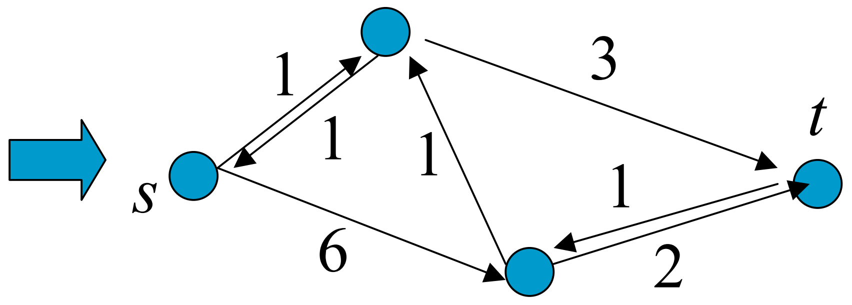
- Computer the Residual Graph G_r
- Repeat
  - Find a directed path in G_r from the source to the sink by depth-first search
  - Augment the flow of this path by an amount equal to the minimum residual capacity of the edges along this path
  - Update the residual graph
- Until no augmenting path exists

An Example of Augmenting Path Algorithm - I

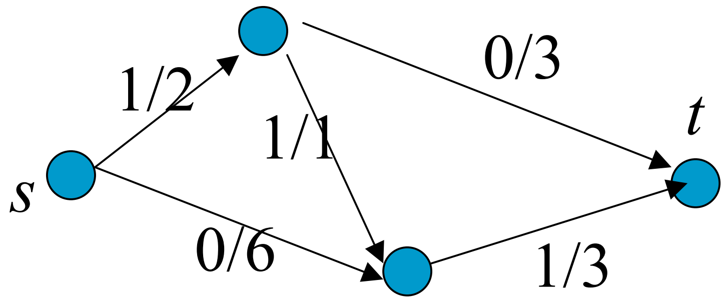
Augmenting Path



Residual Graph

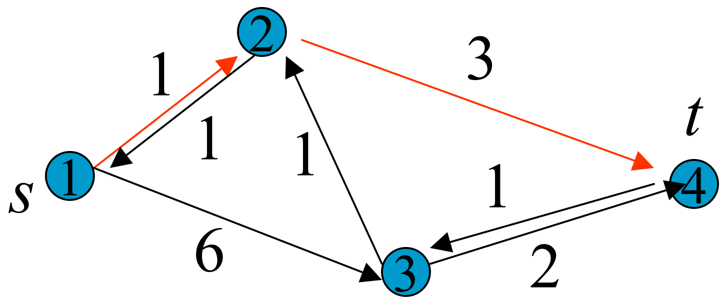


Current Flow

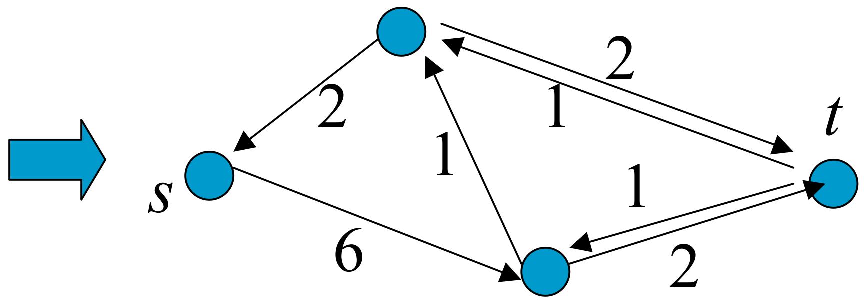


An Example of Augmenting Path Algorithm - II

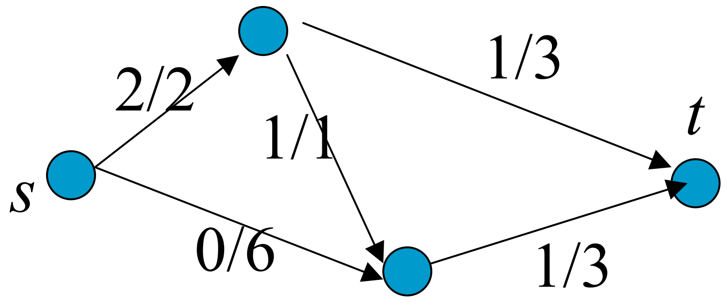
Augmenting Path



Residual Graph

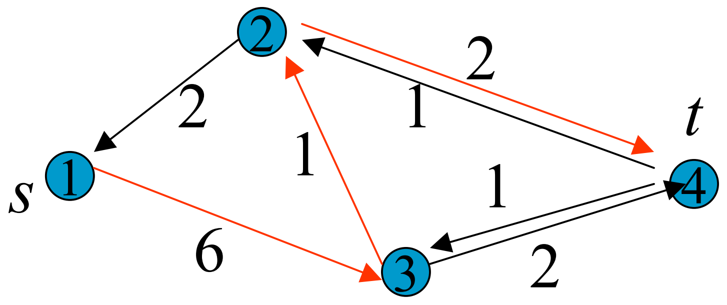


Current Flow

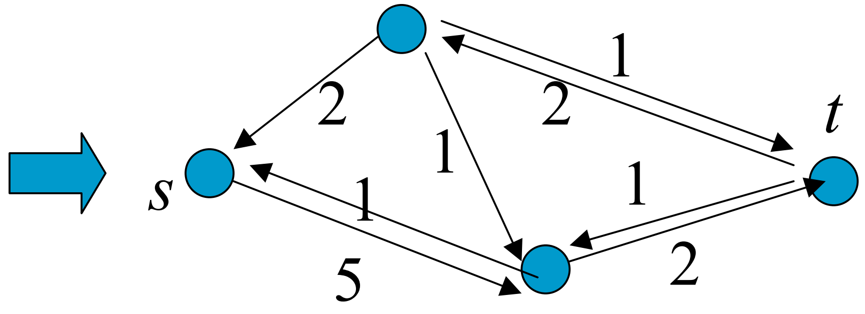


An Example of Augmenting Path Algorithm - III

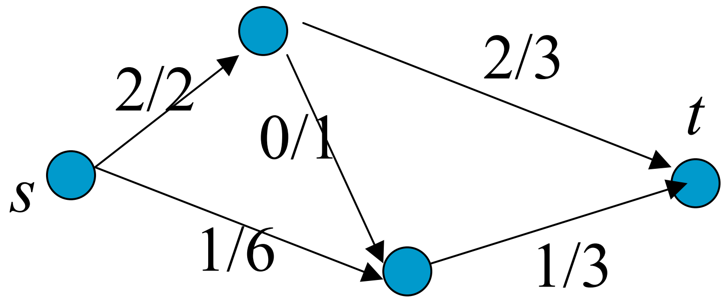
Augmenting Path



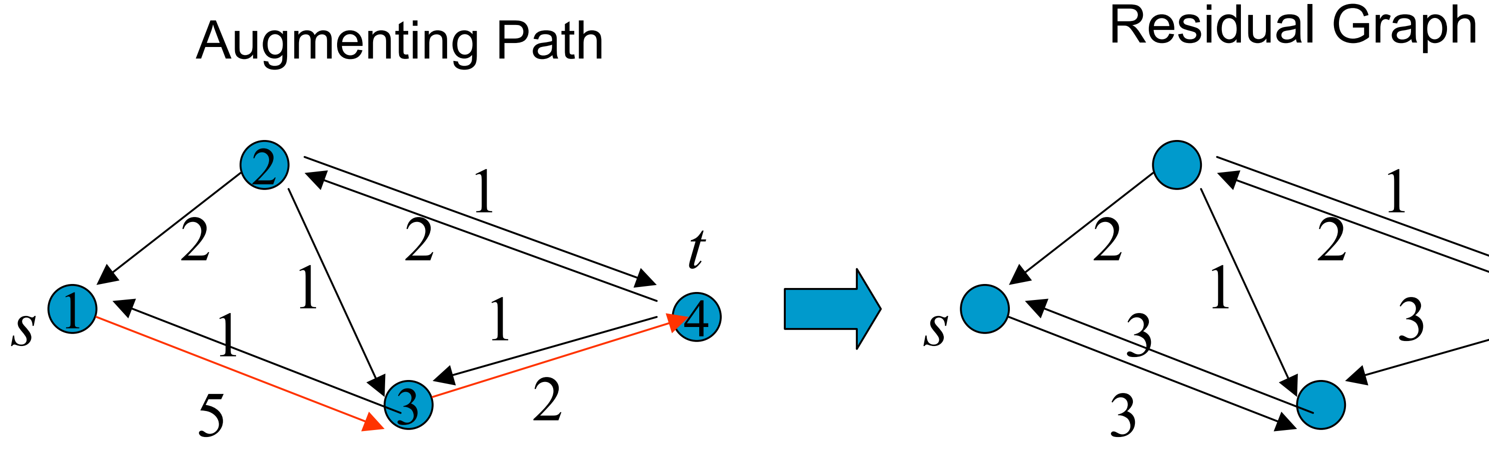
Residual Graph

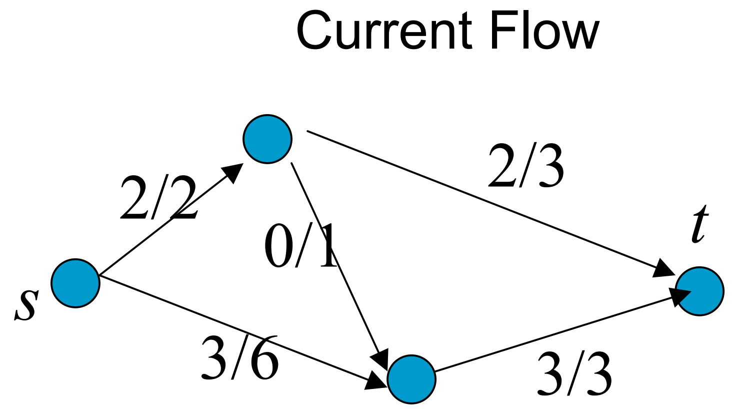


Current Flow



An Example of Augmenting Path Algorithm - IV





Complexity of Basic Augmenting Path Algorithm

$O(nU)$ Augmentations

$O(nmU)$ Runtimes

n is the number of nodes
m is the number of arcs
U is the maximum capacity
(all capacity is integer)

Generic Improved Augmenting Path Algorithm (Ford-Fulkerson)

- Computer the Residual Graph G_r
- Repeat
  - Find a directed path in G_r from the source to the sink by better methods
  - Augment the flow of this path by an amount equal to the minimum residual capacity of the edges along this path
  - Update the residual graph
- Until no augmenting path exists

Improved Augmenting Path Algorithm

- Find a directed path in G_r from the source to the sink by better method
  - larger augmenting flow
    - Maximum capacity algorithm
    - Capacity scaling algorithm
  - shorter path
    - Shortest augmenting path algorithm
    - Layered network algorithm
  - store searching history
    - Dynamical tree algorithm

Maximum Capacity Algorithm

- Find a path with the maximum residual capacity by O(m+logn)
- O(mlogU) augmentations
- runs in O(mlogU(m+logn))
- Inefficient in practice

Capacity Scaling Algorithm

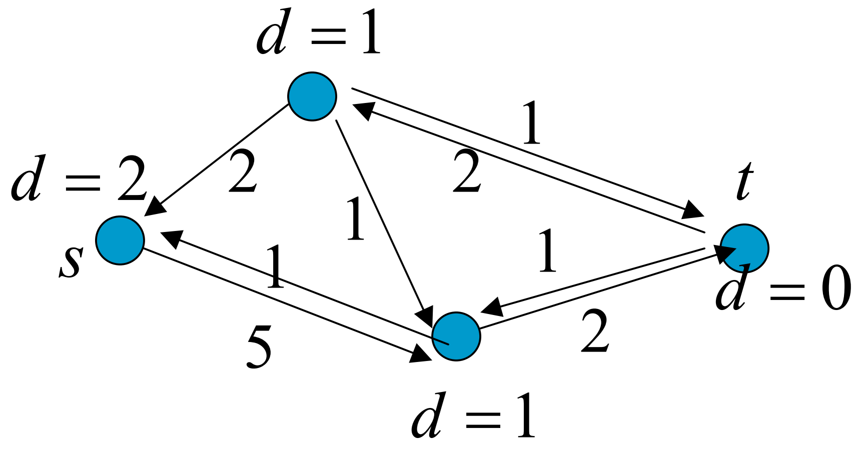
- Set a threshold starts from $2^{\lfloor \log U \rfloor}$ and halves at every scaling phase
- Before the scaling phase
  - get sub-residual graph contains larger capacity than threshold
  - do path augment iteration
- O(mlogU) augmentation and runs in O(m^2logU)

Improved Augmenting Path Algorithm

- Find a directed path in G_r from the source to the sink by better method
  - larger augmenting flow
    - Maximum capacity algorithm
    - Capacity scaling algorithm
  - shorter path
    - Shortest augmenting path algorithm
    - Layered network algorithm
  - store searching history
    - Dynamical tree algorithm

Distance Labels

- The lower bound of distance to sink (exactly label is the true value)
- Distance function d assigns each node a nonnegative integers d(i) , it is valid if $d(t)=0$ and $d(i) \le d(j)+1$ if <i,j> is an arc in residual network
- Arc <i,j> is admissible if $d(i)=d(j)+1$
- An admissible path is a shortest augmenting path from source to the sink



Shortest Augmenting Path Algorithm

- Augment flows along shortest directly paths from source to sink
- Use distance labels to identify shortest paths
- $O(n^2m)$
- Easy to implement and very efficient in practice

Layered Network Algorithm

- Construct layered network by the distance of the node to the sink
- Find path from source to sink in the layered network
- $O(n^2m)$

Improved Augmenting Path Algorithm

- Find a directed path in G_r from the source to the sink by better method
  - larger augmenting flow
    - Maximum capacity algorithm
    - Capacity scaling algorithm
  - shorter path
    - Shortest augmenting path algorithm
    - Layered network algorithm
  - store searching history
    - Dynamical tree algorithm

Dynamic Tree Algorithm

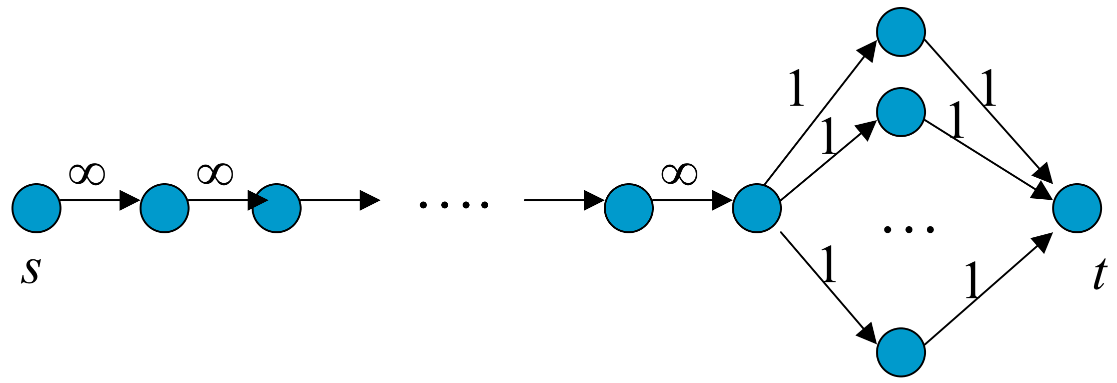
- Use the dynamic tree data structure to implement the shortest augmenting path
- Improve to $O(nm \log n)$
- The dynamic tree data structure is quite sophisticated, has substantial overhead, and its practical usefulness has not yet been established (from *Network Flows: Theory, Algorithms, and Applications*)

Maximum Flow Algorithms

- Augmenting Path (Ford-Fulkerson)
  - Generic ford-fulkerson $O(nmU)$
  - Capacity scaling $O(nm \log U)$
  - Successive shortest path $O(n^2 m)$
  - Layered Network $O(n^2 m)$
  - .....
- Push/Relabel
  - Generic preflow-push $O(n^2 m)$
  - FIFO preflow-push $O(n^3)$
  - Highest-label preflow-push $O(n^2 \sqrt{m})$
  - .....

Drawback of Path Augmentation

- Pathological case for all path augmentation algorithm
  - Every time augment a path has common part



Basic Idea of Push/Relabel

- Augment flow by arc instead of by path. Push to the flow from source to sink as much as possible
  - Pre-flow and exceed
  - Adjust the pre-flow to flow by reducing the exceed, push the excess of node to neighbor nodes
  - Use distance label to guide where to push

Pre-flow and Exceed

- Pre-flow is a function f
  - Capacity constraint : $f(i,j) \le cap(i,j)$
  - Relaxation of Mass balance constraint:

$$\sum_{<i,j>\in E} f(i,j) - \sum_{<k,i>\in E} f(k,i) \ge 0 \quad i \in V - \{s,t\}$$

- Excess of each node

$$e(i) = \sum_{<i,j>\in E} f(i,j) - \sum_{<k,i>\in E} f(k,i)$$

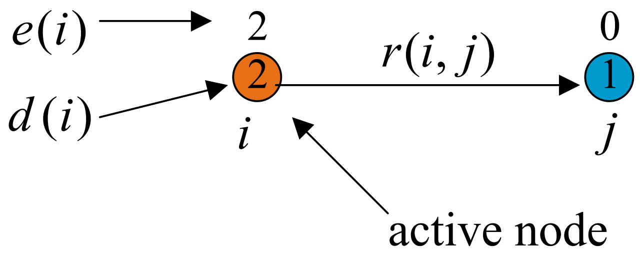
- Active node is the node has positive excess

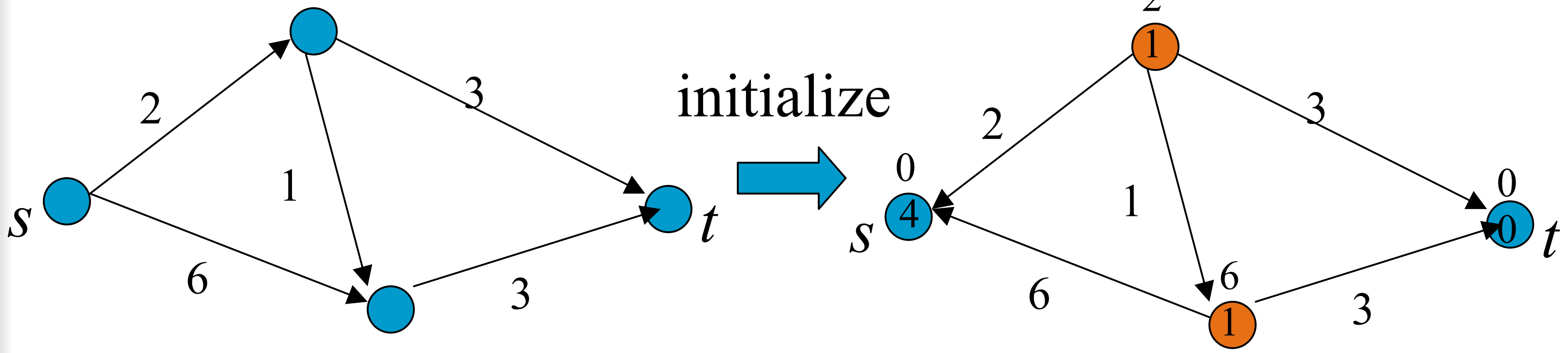
Push/Relabel Algorithm

- Initialize
  - Computer the distance label
  - Push flow from source node
  - Set d(s) = n
- While graph has active node
  - Select an active node
  - Push/Relabel
    - If has admissible arc from this node, push flow by $\min(e(i), r(i, j))$
    - Else replace the label by

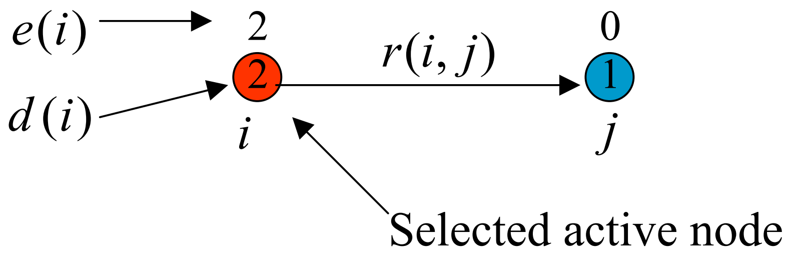
$$\min(d(j)+1) :< i, j > \in E(r), r(i, j) > 0$$

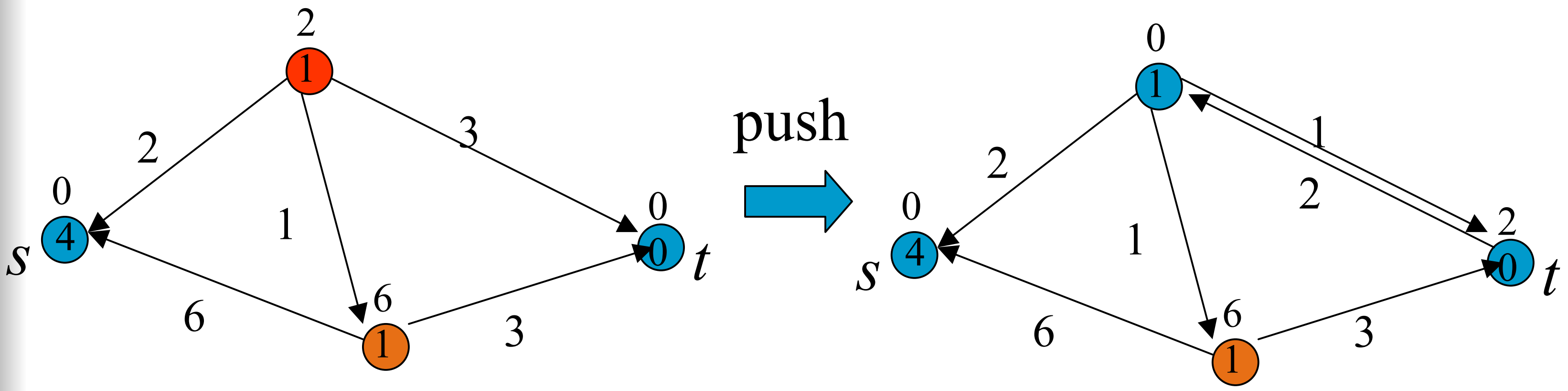
An Example of Push/Relabel Algorithm - I



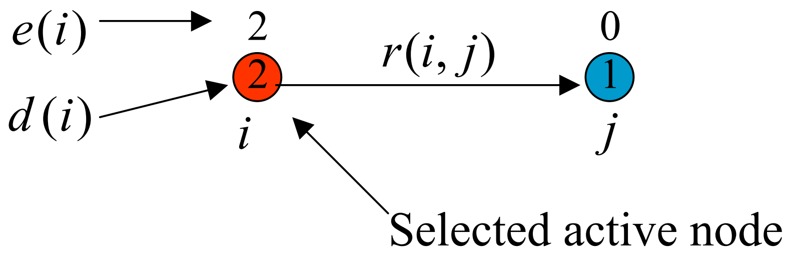


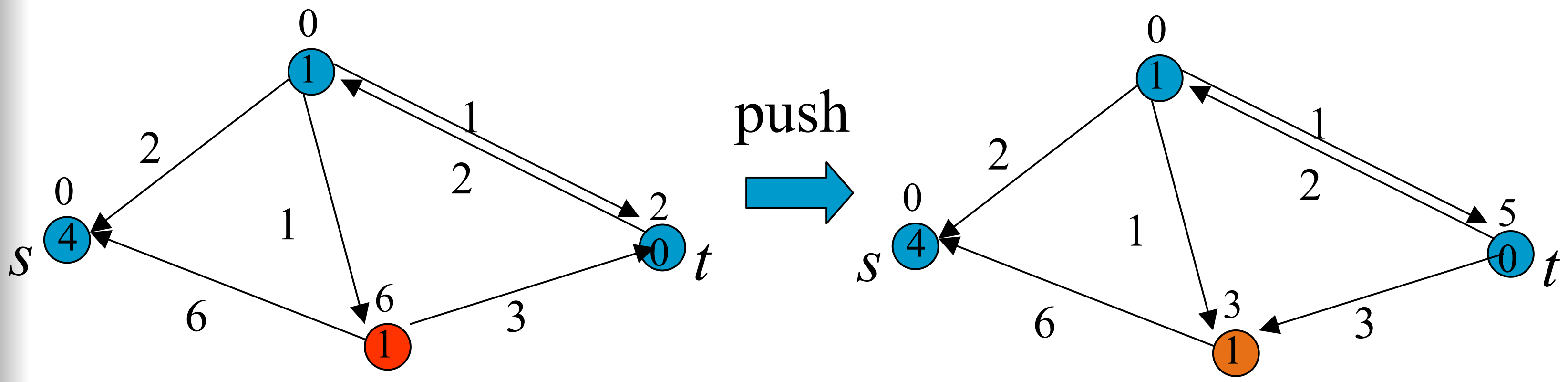
An Example of Push/Relabel Algorithm - II



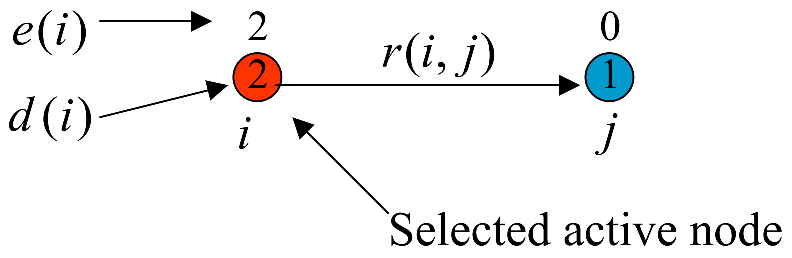


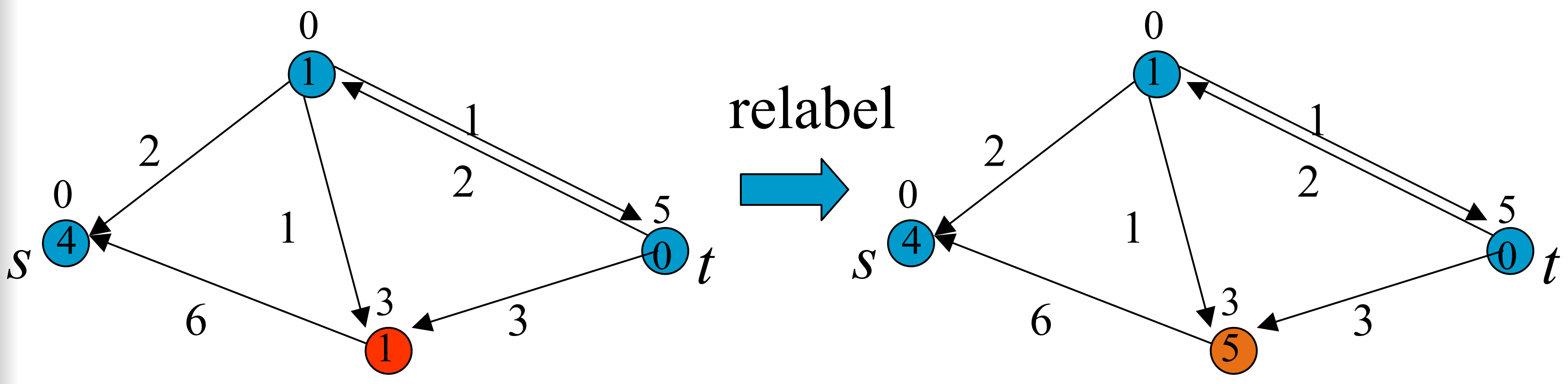
An Example of Push/Relabel Algorithm - III



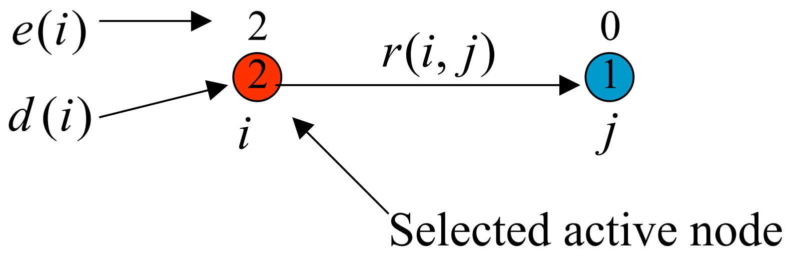


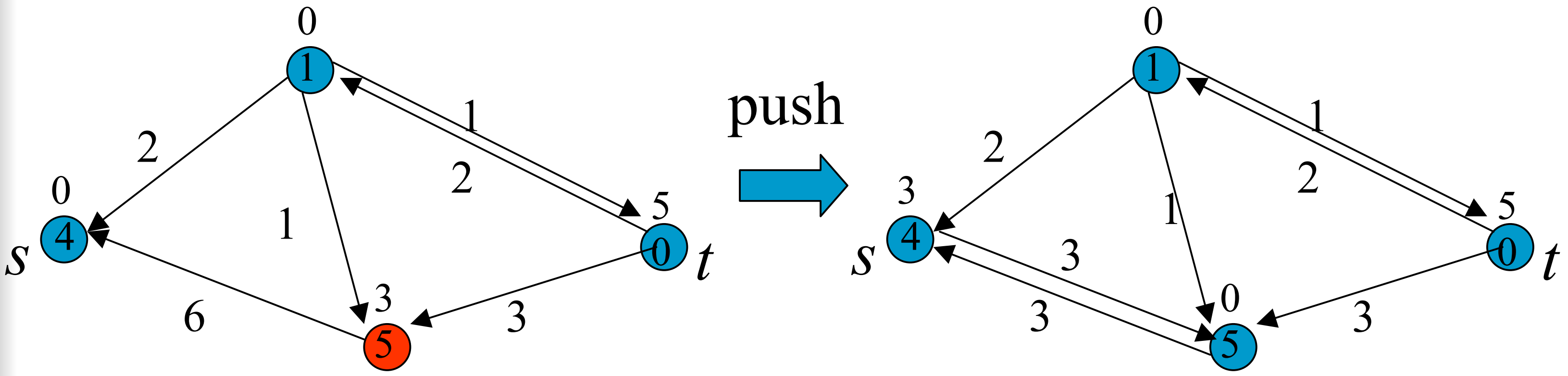
An Example of Push/Relabel Algorithm - IV





An Example of Push/Relabel Algorithm - V





Push/Relabel Algorithm

- Initialize
  - Computer the distance label
  - Push flow from source node
  - Set d(s) = n
- While graph has active node
  - Select an active node
  - Push/Relabel
    - If has admissible arc from this node, push flow
    - Else replace the label by

Generic Improved Push/Relabel Algorithm

- Initialize
  - Computer the distance label
  - Push flow from source node
  - Set d(s) = n
- While graph has active node
  - Select an active node by better method
  - Push/Relabel
    - while has admissible arc from this node,
      - push flow
    - Else replace the label by

FIFO Push/Relabel Algorithm

- Choose active node by FIFO use a active nodes list
- Examine node: push flow until no exceed or change label
- $O(n^3)$
- Very efficient in practice

Highest-label Push/Relabel Algorithm

- Choose active node by highest distance label
- Examine node: push flow until no exceed or change label
- $O(n^2\sqrt{m})$
- Possibly the most efficient maximum flow algorithm in practice

Excess Scaling Push/Relabel Algorithm

- Select active node with sufficiently large excesses, and among these nodes, select a node with smallest distance label
- $O(nm + n^2 \log U)$
- Efficient without using complex data structure

Questions

- Thank you!

Suggestions on Algorithm Selection

- *Y.Boykov and V.Kolmogorov,* An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision, in PAMI Sep 2004
  - Dynamic Tree Algorithm

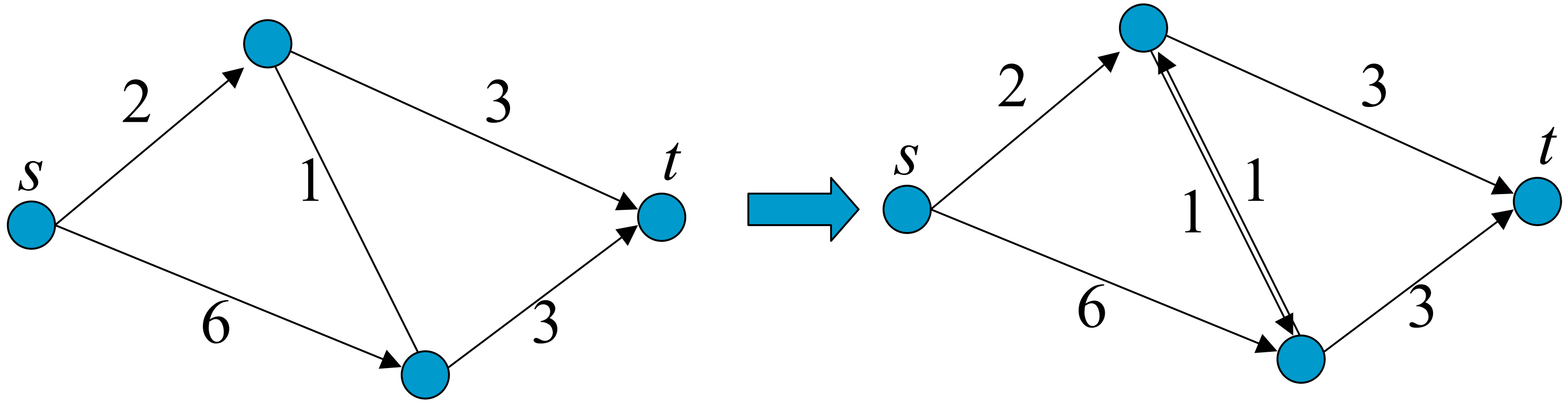
    Best performance for common vision problem.

  - FIFO Push/Relabel Algorithm
  - Highest Level Algorithm
  - Shortest Augmenting Path Algorithm

    Efficient and easy to implement

Undirected edge

- Undirected edge equals to a pair of directed arc



Flow

- Flow is a real value function f that assign a real value $f(i,j)$ to each arc under :
  - Capacity constraint : $f(i,j) \le cap(i,j)$
  - Mass balance constraint:

$$\sum_{<i,j>\in E} f(i,j) - \sum_{<k,i>\in E} f(k,i) = \begin{cases} 0 & i \in V - \{s,t\} \\ |f| & i = s \\ -|f| & i = t \end{cases}$$

$|f|$ is the value of flow f

  - For notation convenience, also has skew symmetry

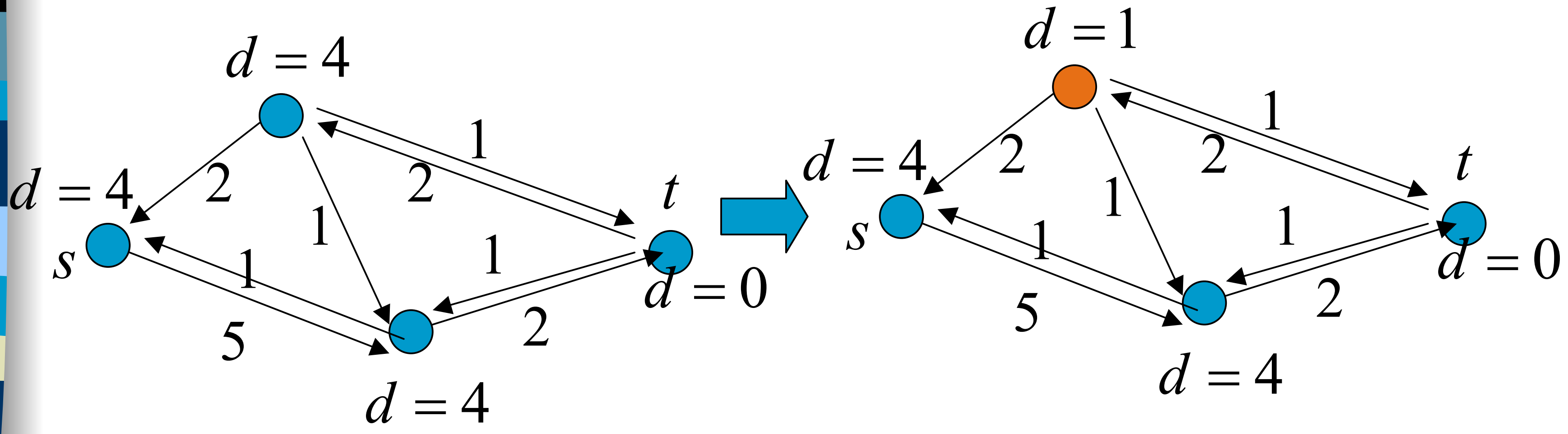
$$f(i,j) = -f(j,i)$$

Properties of Distance Labels

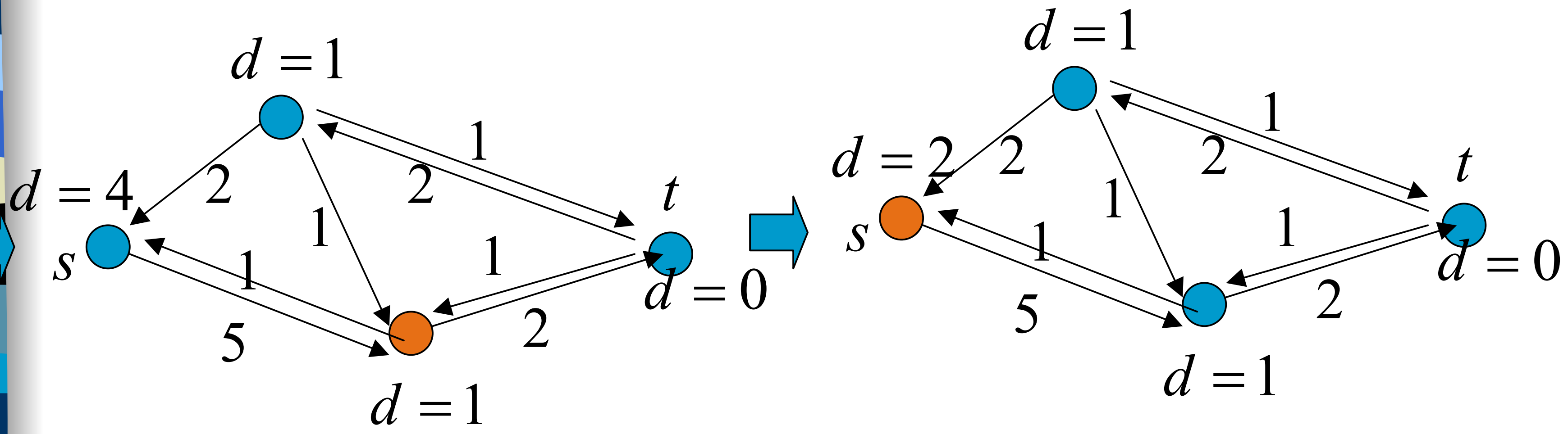
- The distance label d(i) of a valid distance labels is a lower bound on the shortest length from node i to sink in the residual network. And we call the labels are exact if d(i) equals the shortest length from i to sink.
- If $d(s) = n$, the residual network contains no directed path from the source to sink
- Arc <i,j> is admissible if $d(i) = d(j) + 1$
- An admissible path is a shortest augmenting path from source to the sink

Construct Distance Labels

- Initialize all label by n, except the sink node by 0
- Backward breadth-first search from the sink node
- $\min(d(j)+1) :< i, j > \in E(r), r(i,j) > 0$



Construct Distance Labels



A Case Study: Timing Budgeting with network flow

$$Max \sum_{e_{ij}\in E} b_{ij} \qquad (1)$$

$$\sum (d_{ij}+b_{ij}) \le T \quad \forall SI \rightarrow SO paths \qquad (2)$$

$$b_{ij}, d_{ij}, T \in Z_{+} \quad \forall e_{ij} \in E \qquad (3)$$

$$\Longrightarrow$$

$$Max \sum_{e_{ij}\in E} b_{ij} \qquad (5)$$

$$r_i = r_j + d_{ij} + b_{ij} \quad \forall e_{ij} \in E \qquad (6)$$

$$r_{SI} - r_{SO} \le T \qquad (7)$$

$$r_i, b_{ij} \in Z_{+} \quad \forall v_i \in V \text{ and } e_{ij} \in E \qquad (8)$$

Lemma 1. *In an optimal budget assignment, the delay of all paths from SI to SO is T.*

$$\sum_{e_{ij}\in E} b_{ij} = \sum_{e_{ij}\in E} r_i - r_j - d_{ij} = \sum_{v_i\in V} r_i[out(v_i) - in(v_i)] - \sum_{e_{ij}\in E} d_{ij}$$

$$Max \sum_{i\in V} \rho_i r_i \qquad (9)$$

$$r_j - r_i \le -d_{ij} \quad \forall e_{ij} \in E \qquad (10)$$

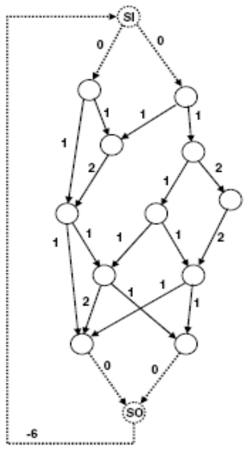
$$r_i \in Z_{+} \quad \forall v_i \in V \qquad (11)$$

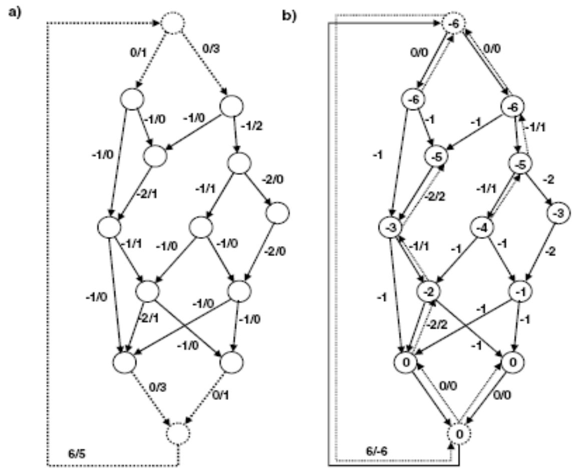
$$\Longrightarrow$$

$$Min \sum_{e_{ij}\in E} -d_{ij} y_{ij} \qquad (12)$$

$$\sum_{e_{ki}\in E} y_{ki} - \sum_{e_{ij}\in E} y_{ij} = \rho_i \quad \forall v_i \in V \qquad (13)$$

$$y_{ij} \in Z_{+} \quad \forall e_{ij} \in E \qquad (14)$$





		dTLC-S	dTLC-LP	LP_trans	LP	nw	nw_sp	input	nw_vdd	nw_global
alu4	10716	124	483	254	96	51	2	23	17	9
apex2	14860	379	1153	954	425	137	3	58	53	23
apex4	9131	178	461	431	196	53	2	25	17	9
bigkey	18622	416	1344	969	501	169	3	97	51	18
clma	91620	16800	55901	>8hours	>8hours	5652	102	1017	3588	945
des	15243	288	1055	857	350	154	3	73	62	16
diffeq	13664	144	554	476	281	56	2	30	21	3
dsip	11444	132	407	407	180	79	2	39	29	9
elliptic	30192	913	3137	1883	900	413	8	201	199	5
ex1010	33265	1423	5109	5244	2460	737	1	285	325	126
ex5p	8722	94	187	1358	1222	58	9	25	16	8
frisc	40662	1912	6135	4230	1887	898	15	368	453	62
misex3	10271	107	276	383	190	62	3	30	20	9
pdc	40001	2509	8210	8189	3695	1275	17	375	672	211
s298	16852	238	837	665	399	109	3	50	45	11
s38417	57503	2896	9153	6337	3758	1033	35	345	524	129
s38584	46014	1893	6864	4128	2388	634	21	229	355	29
seq	13426	203	509	391	162	92	2	30	41	19
spla	27908	1009	3340	2018	882	324	17	107	143	57
tseng	9603	72	164	126	60	31	1	20	9	1
Ave.		1586	5264	4760	2442	601	13	171	332	85
		2x	9x	8x	200x	1	1			