Solving the Maximum Flow Problem by Augmenting Flows on Subnetworks

指導教授：王逸琳 博士
李宇欣 博士
洪政煌 博士
黃耀廷 博士

學　　生：李俊賢
日　　期：2009/06/01

Outline

- Introduction
- Literature review
- New maximum flow algorithms
  - Proportional arc augmenting algorithm
  - Flow splitting augmenting algorithm
  - Modified LSDP algorithm
- Computational experiments and analysis
- Conclusions

Motivation

- Ideas of previous max-flow algorithms
  - Send flow via single path
  - Push flow from an active node to its neighbors
- Our idea: augment flow along admissible subnetworks

(subgraphs composed by all shortest paths)

Objective

- Propose three poly-time max-flow algorithms
  - Ship more flows via more spacious arcs
    ➔ **PAA** , $O(n^2m)$ time
  - Distribute flows fairly to each outgoing arc
    ➔ **FSA** , $O(nm^2)$ time
  - Nondegenrate pivoting, by Kirchhoff's circuit laws
    ➔ **MLSDP** , $O(nm^4)$ time
- Computational experiments

Literature Review(1/6)
Network simplex methods

- Fulkerson and Dantzig (1955)
  - Network simplex method : $O(n^2mU)$
- Goldfarb and Hao (1990)
  - Choose the arc closest to the source node : $O(n^2m)$
- Armstrong et al. (1998)
  - Dual network simplex method : $O(n^3)$

Literature Review(2/6)
Augmenting path methods

- Ford and Fulkerson (1956)
  - Augmenting path : $O(nmU)$
- Dinic (1970)
  - Layered network : $O(n^2m)$
  - **To be used in our proposed algorithms**
- Edmonds and Karp (1972)
  - Shortest path：$O(nm^2)$
- Ahuja and Orlin (1991)
  - Distance labels : $O(n^2m)$

Literature Review(3/6)
Preflow push methods

- Karzanov (1974)
  - Preflow push on layered network : $O(n^3)$
- Goldberg and Tarjan (1986)
  - Preflow push with distance labels : $O(nm\log(n^2/m))$
- Cerulli et al. (2008)
  - Combine with augmenting path : $O(n^2m^{1/2})$

Literature Review(4/6)
Scaling methods

- Ahuja and Orlin (1989)
  - Incorporated with preflow push : $O(nm+n^2\log U)$
- Ahuja and Orlin (1991)
  - Incorporated with distance labels : $O(nm\log U)$
- Sedeño-Noda and González-Martín (2000)
  - Two phase scaling method : $O(nm\log(U/n))$

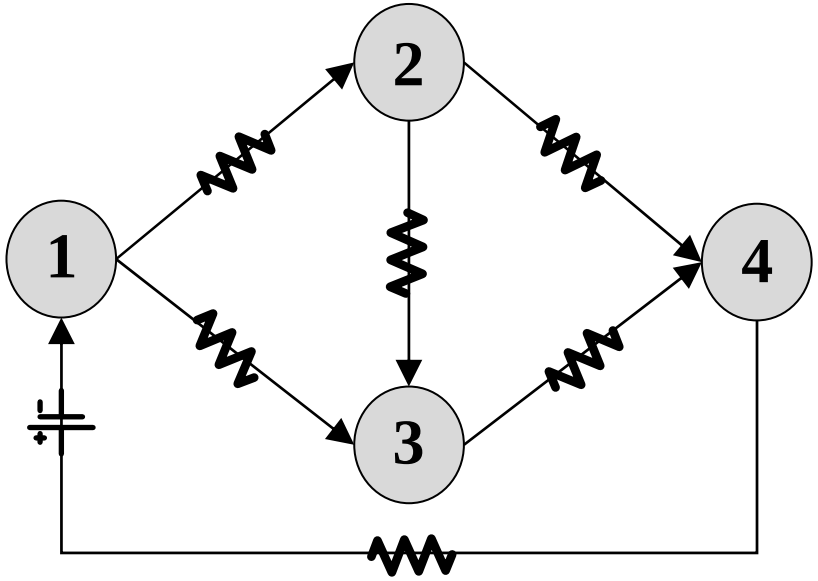
Literature Review(5/6)

Maximum adjacency ordering methods

- Fujishige (2003)
  - MA ordering augmenting path : $O(n(m+n\log n)\log nU)$
- Fujishige and Isotani (2003)
  - MA ordering with scaling phase : $O(nm\log U)$
- Matsuoka and Fujishige (2005)
  - MA ordering with preflows : $O(n(m+n\log n)\log nU)$

Literature Review(6/6)
Least-squares dual-primal method

- Chang (2006)
  - LSDP to solve maximum flow problem
    - NNLS subproblem is solved by Kirchhoff's laws (i.e. KCL, KVL)



NNLS subproblem

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_{12} \\ \delta_{13} \\ \delta_{23} \\ \delta_{24} \\ \delta_{34} \\ \delta_{41} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

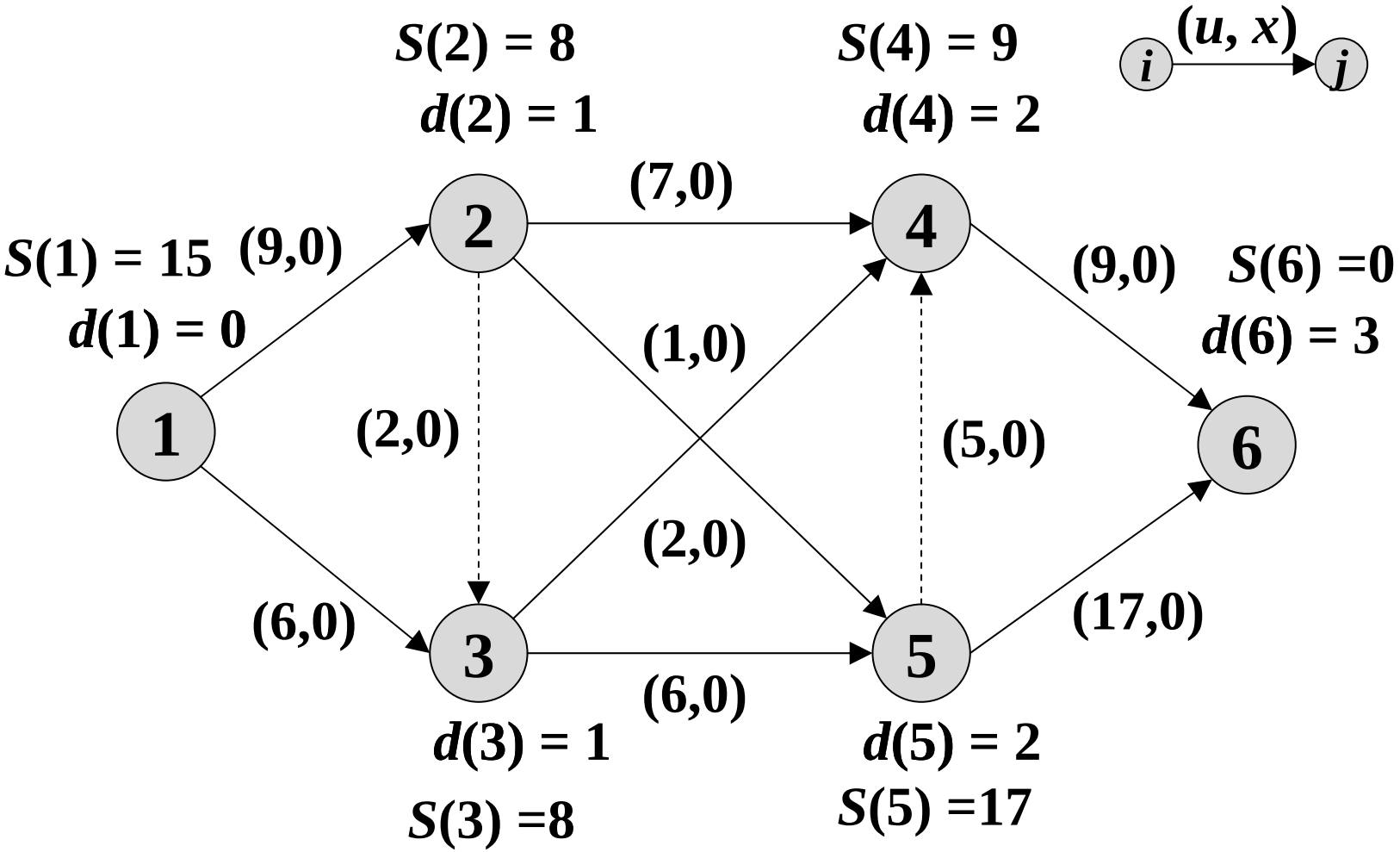
Definitions

- Residual network $G(x)$
  - Residual capacity r_{ij}
    - Unused capacity $u_{ij} - x_{ij}$ on arc (i, j)
    - Current flow x_{ij} on arc (j, i)
- Distance label $d(i)$
  - Minimum number of arcs from source to node i
- Admissible arc (i,j)
  - $d(j) = d(i) + 1$ and $r_{ij} > 0$

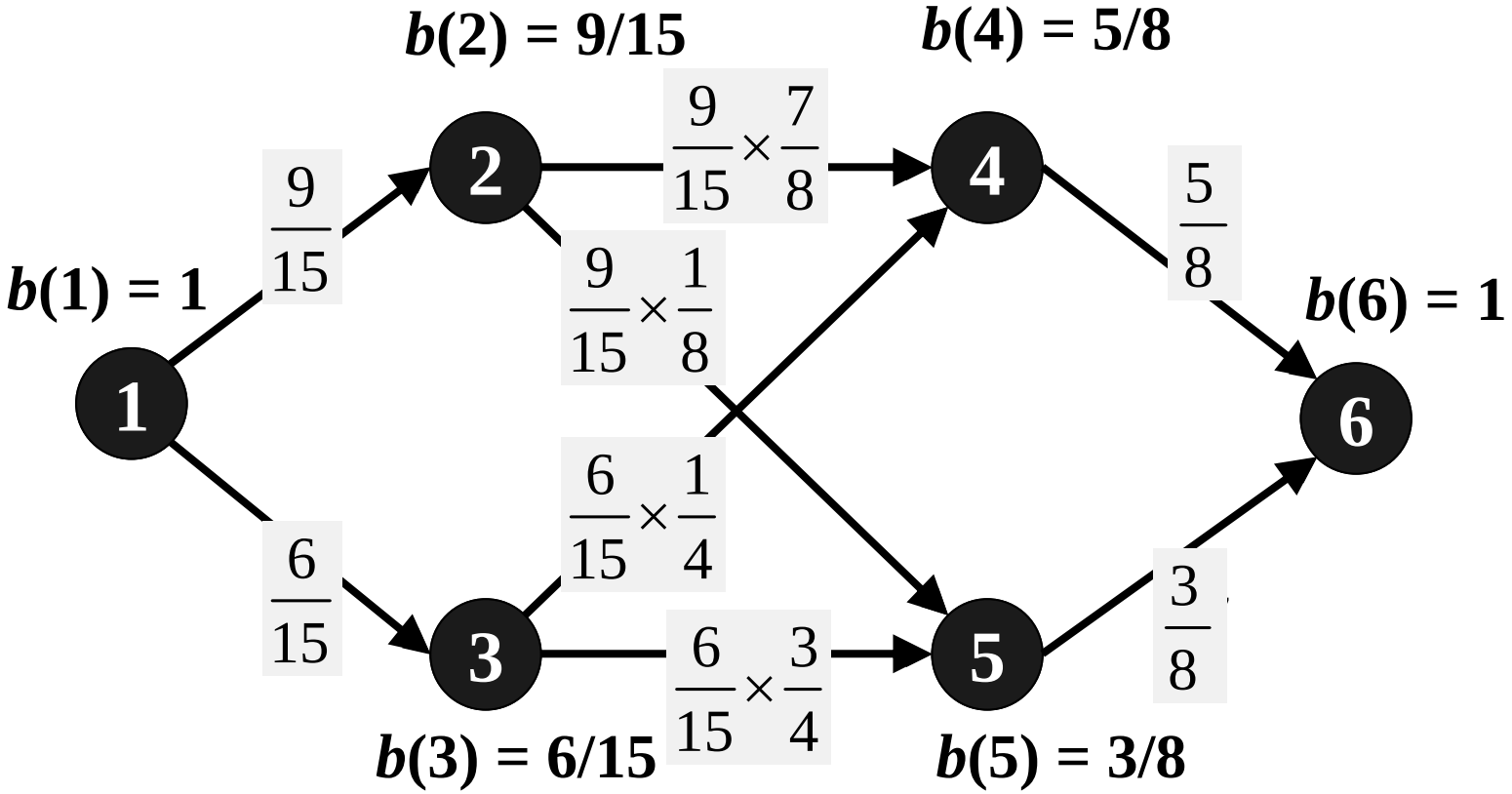
Proportional Arc Augmenting (PAA) Algorithm

- Use BFS and distance labels to construct a layered network
- Starting from node i = s based on a BFS order, calculate δ_{ij} for each admissible arc (i,j)
- Calculate maximum step length θ
- Augment flows by $\Delta x = \delta \times \theta$
- Saturate at least one node (and all of its outgoing arcs) after each augmenting iteration
- Theoretical Complexity：$O(n^2m)$

PAA Example(1/9)



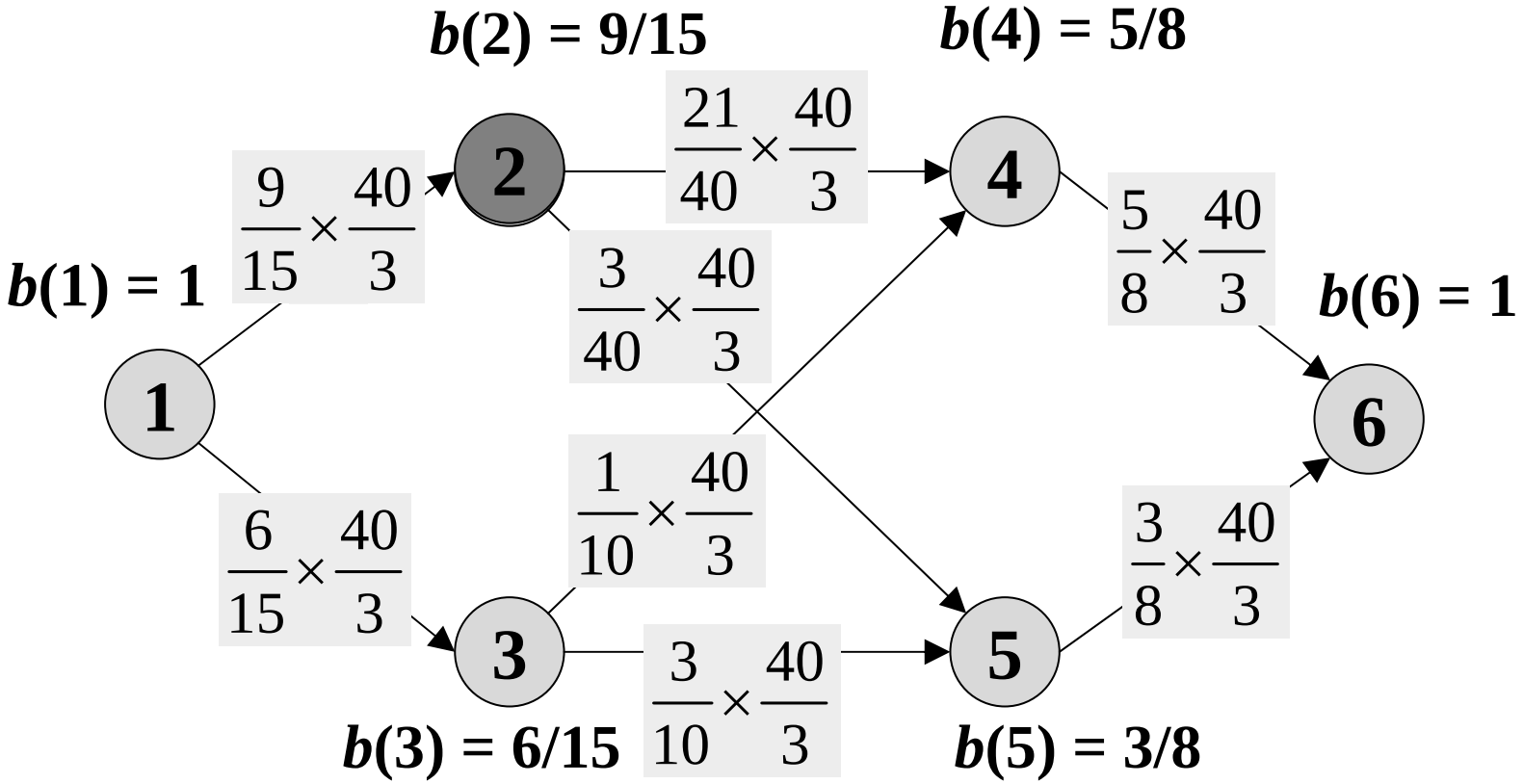
PAA Example(2/9)



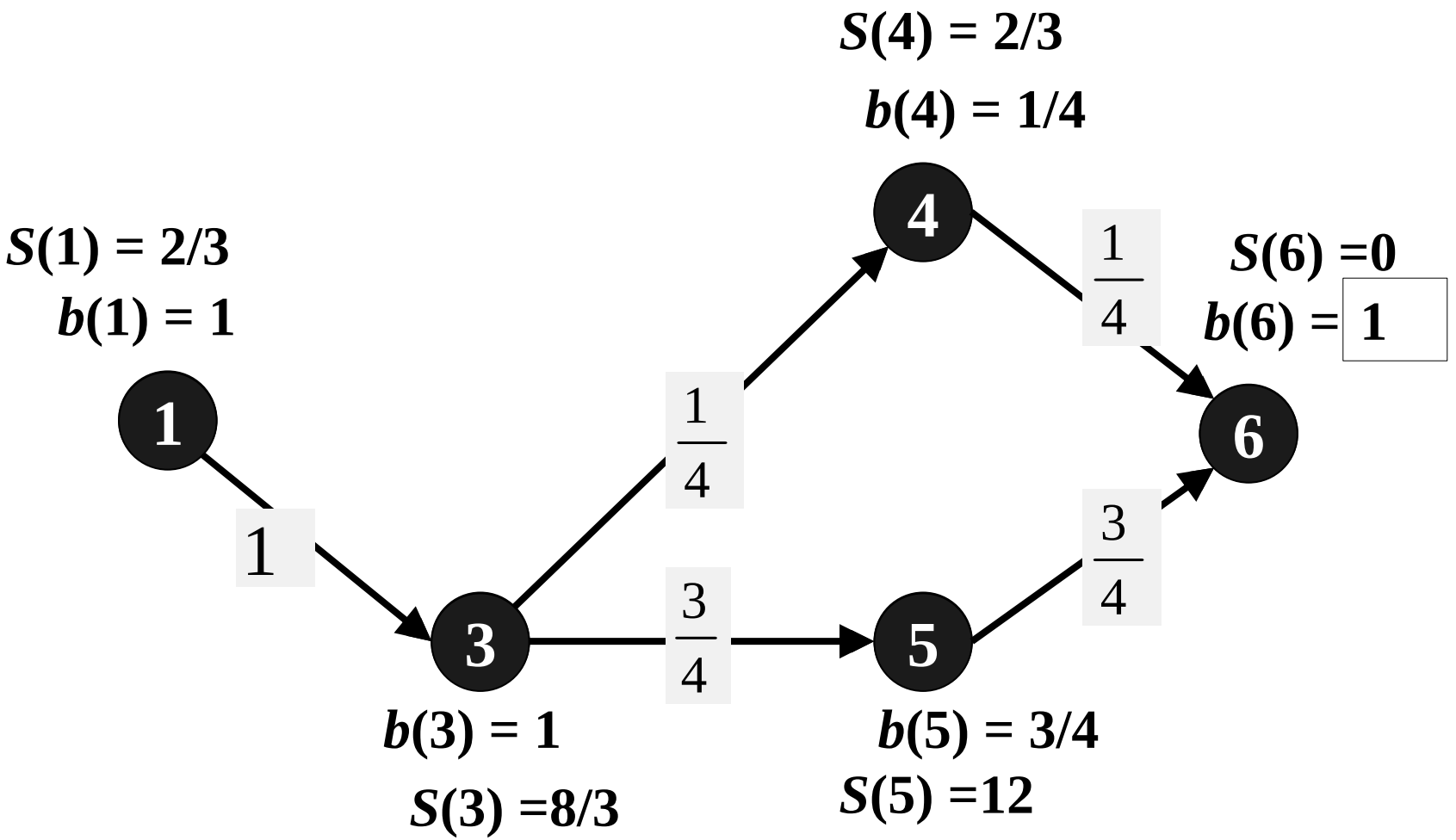
PAA Example(3/9)

$$\theta = \min\left\{\frac{15}{1}, \frac{8}{\frac{9}{15}}, \frac{8}{\frac{6}{15}}, \frac{9}{\frac{5}{8}}, \frac{17}{\frac{3}{8}}\right\} = \frac{40}{3}$$

$$S = S - b \times \theta = \left(\frac{5}{3}, 0, \frac{8}{3}, \frac{2}{3}, 12, 0\right)$$



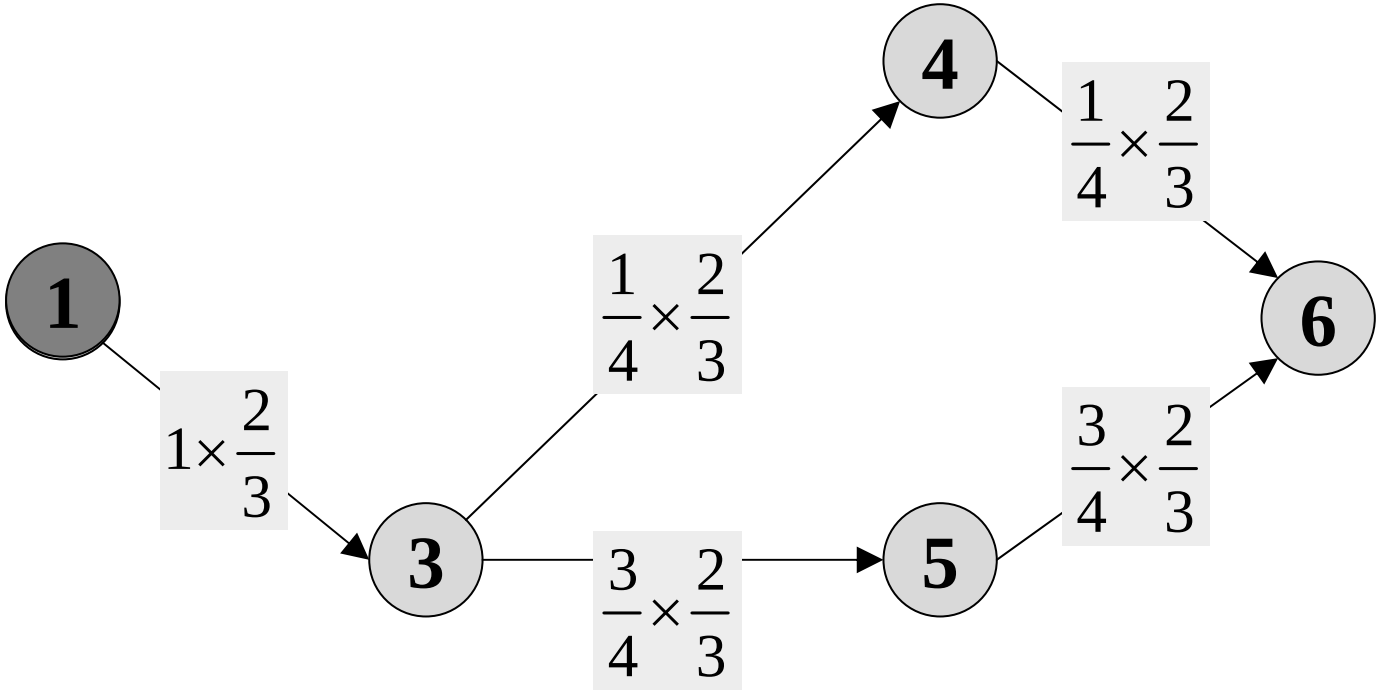
PAA Example(4/9)



PAA Example(5/9)

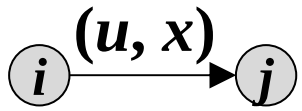
$$\theta = \min\left\{\frac{\frac{2}{3}}{1}, \frac{\frac{8}{3}}{1}, \frac{\frac{2}{3}}{\frac{1}{4}}, \frac{12}{\frac{3}{4}}\right\} = \frac{2}{3}$$

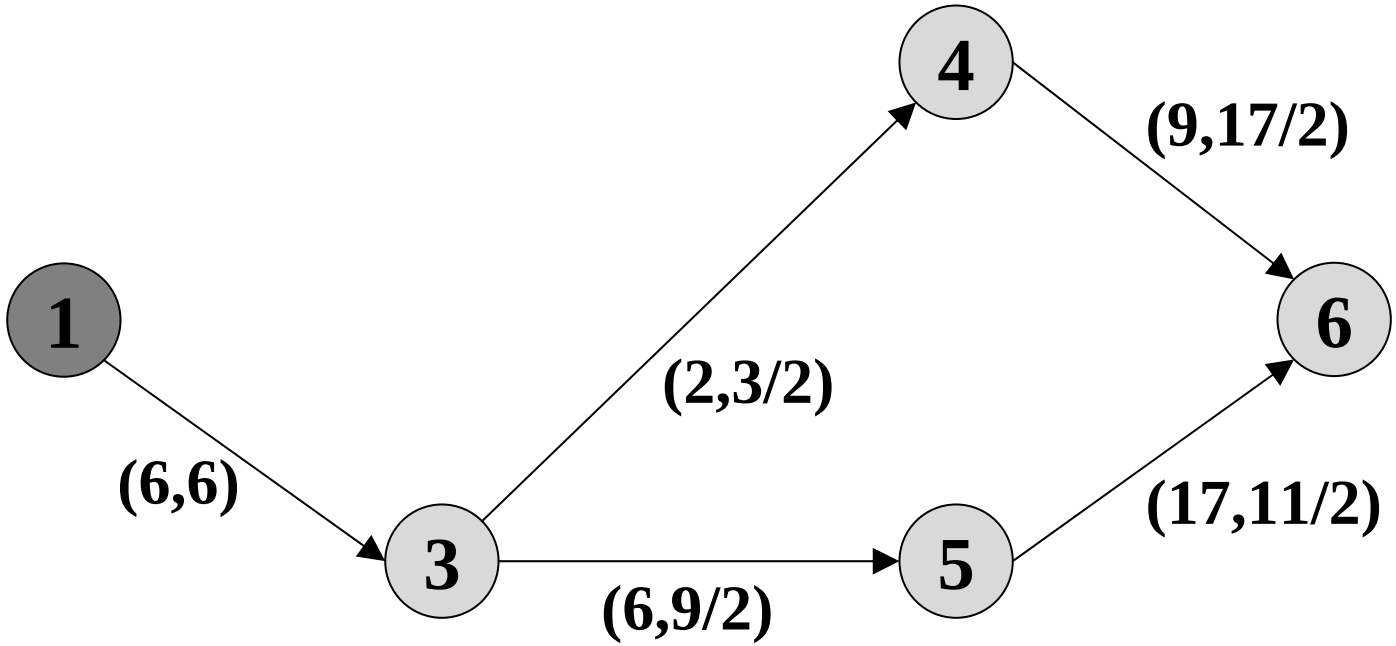
$$S = \left(0, 0, 2, \frac{1}{2}, \frac{23}{2}, 0\right)$$



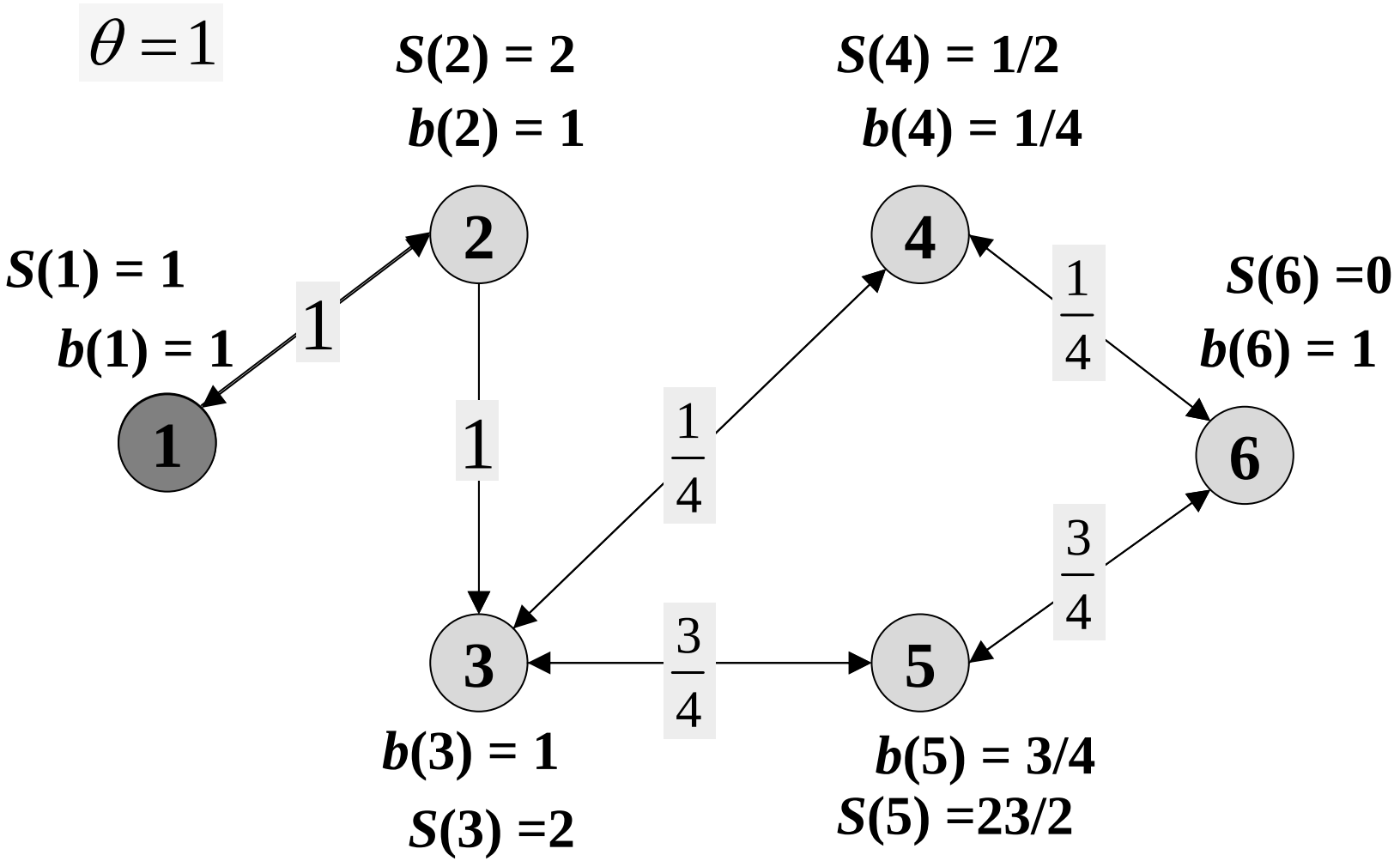
PAA Example(6/9)

NO more *s*-*t* path
in current layered network



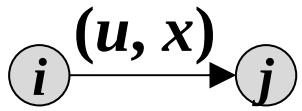


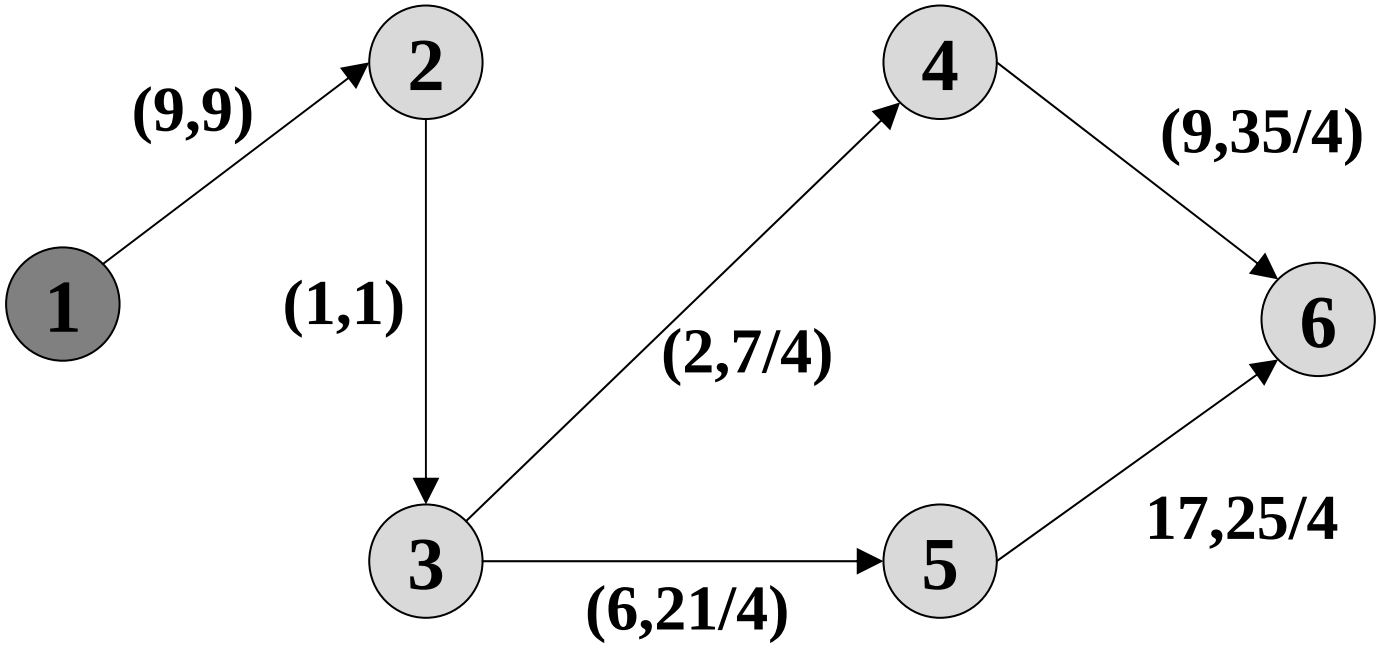
PAA Example(7/9)



PAA Example(8/9)

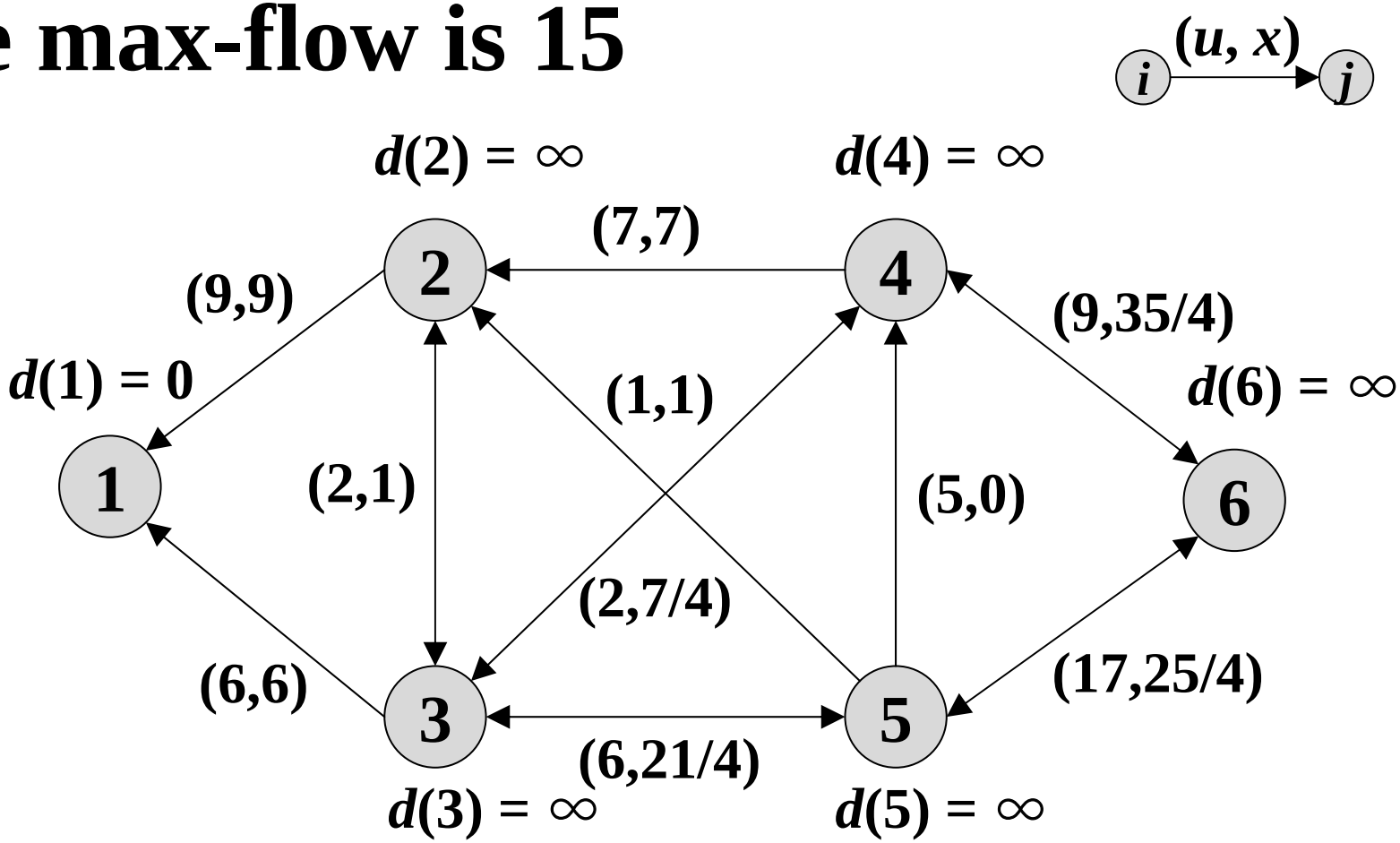
NO more *s*-*t* path
in current layered network





PAA Example(9/9)

The max-flow is 15



Complexity Analysis

- PAA algorithm constructs at most n-1 layered networks in the residual network
- Calculate the max-flow in a given layered network : $O(nm)$
  - Calculate flow augmenting vector δ : $O(m)$
  - Augment flow : $O(m)$
  - Update sum of residual capacity : $O(n)$
  - Remove at least one node after augmenting flow
- Theoretical complexity of PAA : $O(n^2m)$
  - Same as Dinic's algorithm

Convert Fractional Flow to Integral Flow

- Use the flow decomposition method to find integral paths from *s* to *t* to decompose fractional flow
- Can remove at least one arc after decomposing fractional flow
- Theoretical complexity : $O(m^2)$

Speed-up Techniques(1/4)
Skipping Exhausted Nodes

- A doubly linked list to store admissible nodes in BFS order
  - For each admissible node, record its previous and next node, according to the BFS order
  - Skip scanning those nodes not in layered networks
- Update the doubly linked list when exhausted nodes are removed
- Scan admissible nodes by the order of the doubly linked list

Speed-up Techniques(2/4)
Forward and Backward Augmentation

- **Forward augmentation** considers outgoing arcs, **Backward augmentation** considers incoming arcs
- Different augmenting orientations give different δ and θ
- Alternatively swapping augmenting orientations may help
  - In K^{th} layered network, perform forward augmentation
  - In $K+1^{th}$ layered network, perform backward augmentation
  - May select a larger θ to augment flow (same complexity)

Speed-up Techniques(3/4)
Scaling Phase

- **Step0:** $\Delta = 2^{\lceil \log U \rceil}$
- **Step1:** Construct a layered network that only contains those arcs with residual capacities at least $\triangle$
- **Step2:** If no layered network can be constructed by current $\triangle$, recursively set $\triangle' = \triangle/2$ until a layered network can be constructed
- **Step3:** Augment flow by PAA until current layered network is optimal. Then set $\triangle' = \triangle/2$

Speed-up Techniques(4/4)
Scaling Phase

- **Step4:** If $\triangle$=1, then set $\triangle$=0 and find layered networks to augment flow by PAA until the max-flow is attained
- Theoretical complexity of scaling version of PAA : $O(nm\log U+n^2m)$

Flow Splitting Augmenting (FSA) Algorithm

- Fairly distribute the incoming flow of each node to its outgoing arcs in a given layered network
- Remove at least one arc after one time of augmentation
- Total running time in a given layered network :$O(m^2)$
- Theoretical complexity of FSA : $O(nm^2)$
  - Same as Edmonds and Karp's algorithm

Computational Results(1/11)

Tested network families

- GENRMF generator
  - GENRMF-LONG (G-LONG)
  - GENRMF-WIDE (G-WIDE)
- Ak generator (AK)
- Washington generator with function 10 (WAS-10)
- Acyclic dense with equal-capacity arcs (ACU)

Computational Results(2/11)
Tested proposed algorithms

- PAA and its speed-up techniques
  - Skipping exhausted nodes (PAA2)
  - Forward and backward augmentation (PAAFB)
  - Scaling phase (PSA)
- Flow splitting augmenting algorithm (FSA)

Computational Results(3/11)

PAA and its speed-up techniques

Network		Running time (second)			
Family	**(*n*, *m*)**	**PAA**	**PAA2**	**PAAFB**	**PSA**
G-LONG	**(65536, 311040)**	535,89	324.74	294.05	170.113
	(130682, 625537)	2152.531	1421.287	1205.865	706.066
G-WIDE	**(65025, 314840)**	306.7	130.566	128.644	303.728
	(123210, 599289)	1283.38	549.044	557.084	1099.781
AK	**(65542, 98311)**	480.37	329.577	311.962	83.172
	(131078, 196615)	2117.58	1386.224	1331.585	405.33
WAS-10	**(49155, 65537)**	0.016	0.015	0.031	0.028
	(98307, 131073)	0.031	0.031	0.062	0.084
ACU	**(1024, 523776)**	0.125	0.063	0.078	0.047
	(2048, 2096128)	0.532	0.313	0.328	0.234

Computational Results(4/11)

PAA and its speed-up techniques

Network		# of augmentation		
Family	**(*n*, *m*)**	**PAA2**	**PAAFB**	**PSA**
G-LONG	**(65536, 311040)**	51243.6	51239.8	23928
	(130682, 625537)	105204.6	105200.2	46416.8
G-WIDE	**(65025, 314840)**	57566.4	57592.8	44323.8
	(123210, 599289)	110899.2	110953.6	90539.6
AK	**(65542, 98311)**	32770	32770	10928
	(131078, 196615)	65538	65538	21850
WAS-10	**(49155, 65537)**	1	1	1
	(98307, 131073)	1	1	1
ACU	**(1024, 523776)**	2	2	2
	(2048, 2096128)	2	2	2

Computational Results(5/11)

PAA and FSA

Network		Running time (second)		# of augmentation	
Family	**(*n*, *m*)**	**PAA2**	**FSA**	**PAA2**	**FSA**
G-LONG	**(65536, 311040)**	324.74	780.406	51243.6	107550.4
	(130682, 625537)	1421.287	3245.1	105204.6	215173
G-WIDE	**(65025, 314840)**	130.566	32.516	57566.4	128798.6
	(123210, 599289)	549.044	1423.503	110899.2	248974.4
AK	**(65542, 98311)**	329.577	285.919	32770	24578
	(131078, 196615)	1386.224	1196.606	65538	49154
WAS-10	**(49155, 65537)**	0.015	0.031	1	1
	(98307, 131073)	0.031	0.078	1	1
ACU	**(1024, 523776)**	0.063	0.078	2	2
	(2048, 2096128)	0.313	0.297	2	2

Performance Summary

fast ⟷ **slow**

- GENRMF-LONG

  PSA << PAAFB < PAA2 << PAA < FSA
- GENRMF-WIDE

  PAA2 < PAAFB << PSA < PAA < FSA
- AK

  PSA << FSA < PAAFB < PAA2 << PAA
- WAS-10

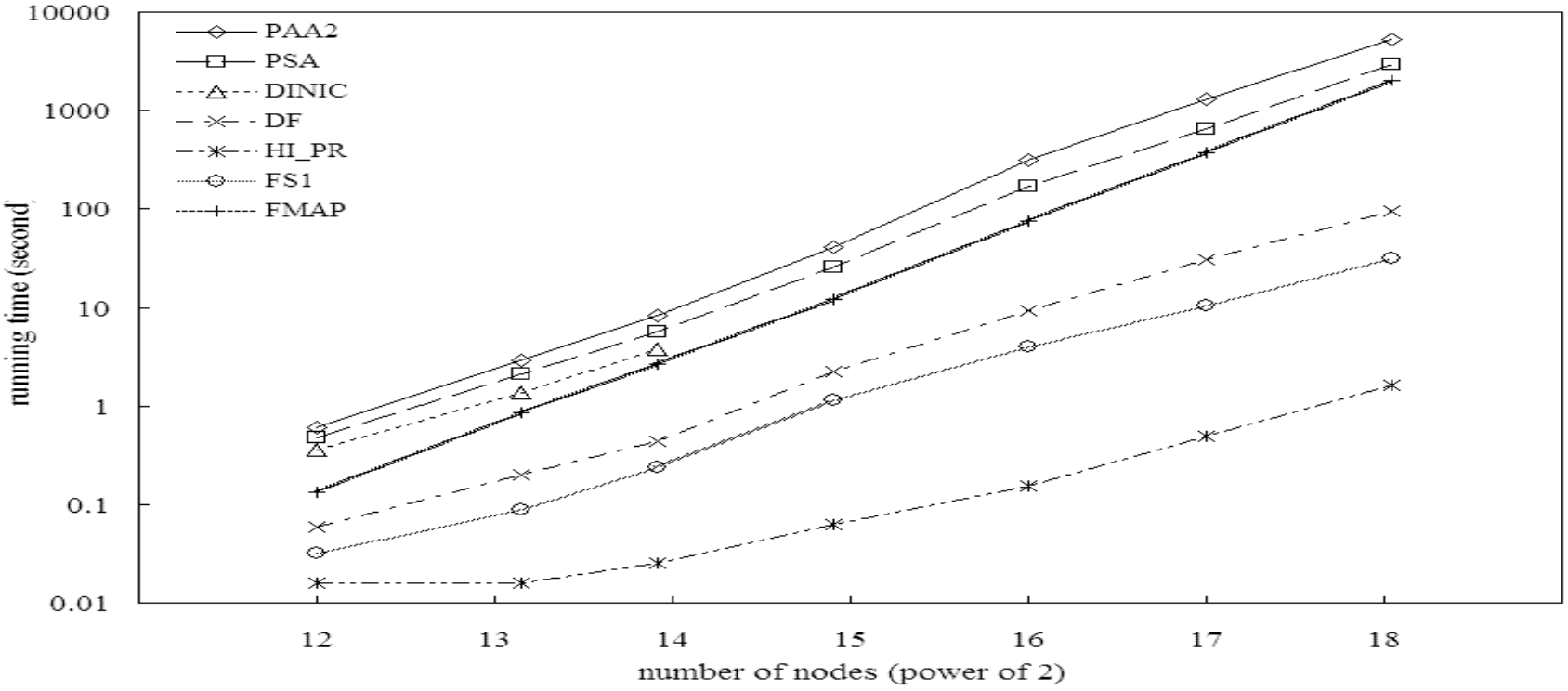
  PAA2 ≒ PAA < PAAFB < FSA < PSA
- ACU

  PSA < PAA2 ≒ FSA < PAAFB < PAA

Computational Results(6/11) Tested algorithms

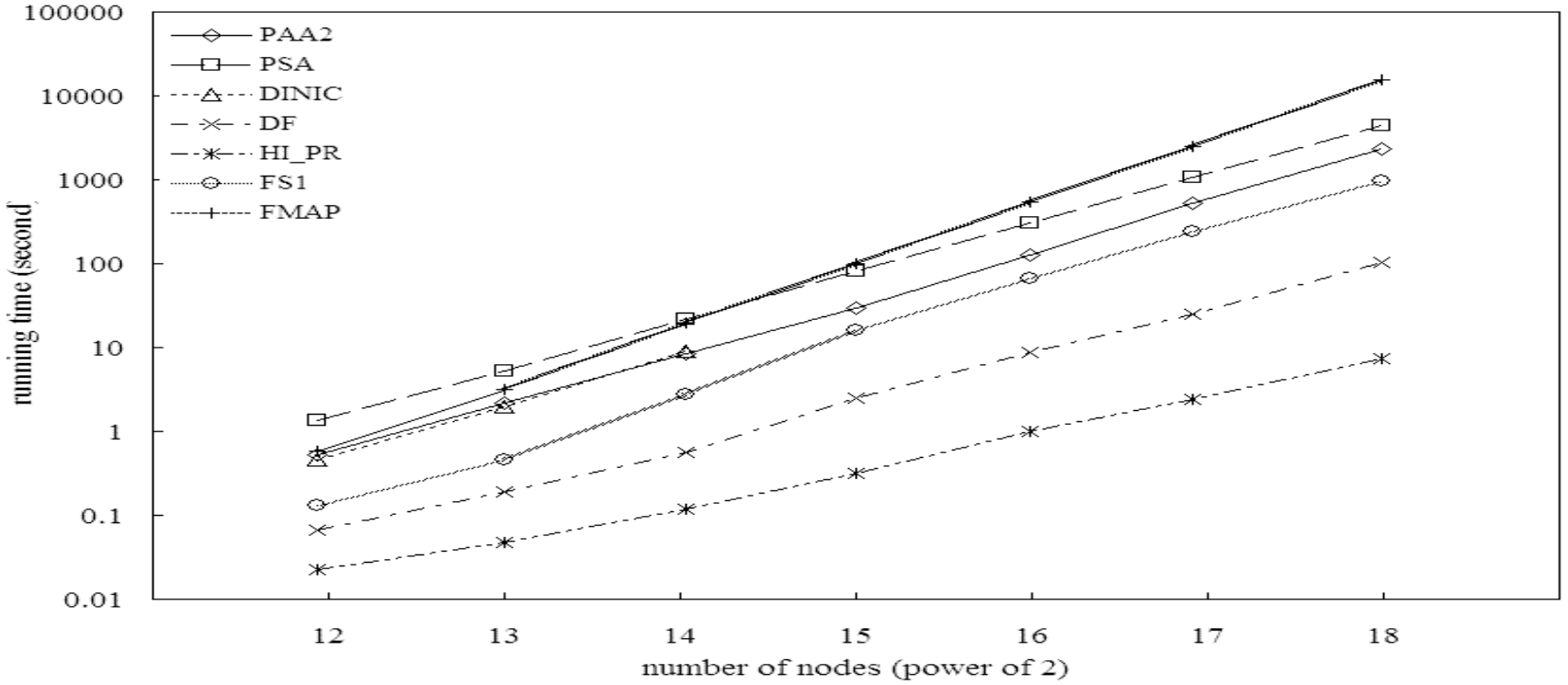
- PAA2 and PSA
- Dinic's algorithm
  - Implemented by Setubal (DINIC)
    - Maximum nodes：20000
  - Implemented by Cherkassky and Goldberg (DF)
- Highest-label preflow push algorithm (HI_PR)
- MA ordering
  - scaling phase (FS1)
  - Preflow (FMAP)

Computational Results(7/11)
G-LONG



G-LONG	Running time (second)						
(*n*, *m*)	PAA2	PSA	DINIC	DF	HI_PR	FS1	FMAP
(65536, 311040)	327.74	170.11	*	9.24	0.153	4.02	74.99
(130682, 625537)	1421.29	706.06	*	30.45	0.494	10.51	369.14
(270848, 1306607)	6066.44	3136.5	*	95.65	1.625	31.4	2011.4

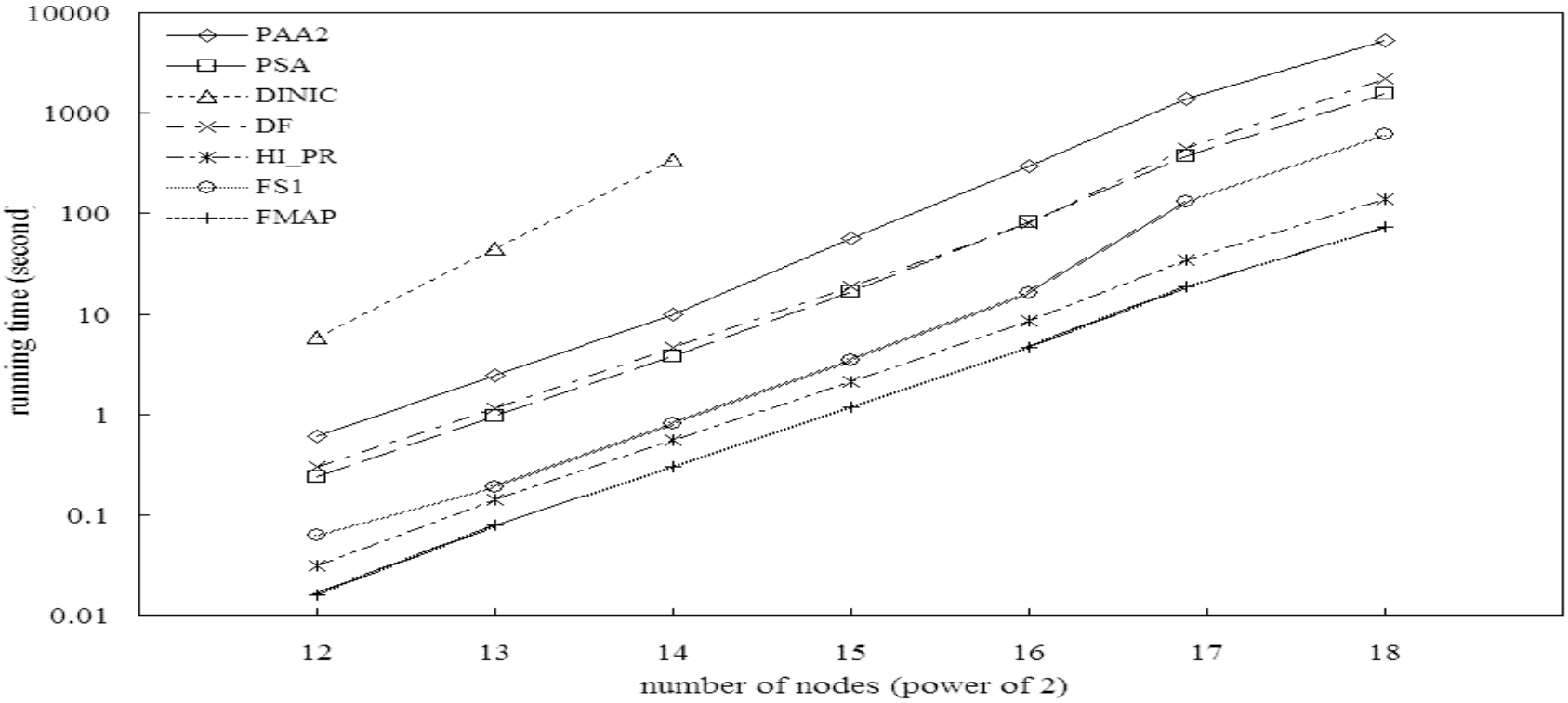
Computational Results(8/11)
G-WIDE



G-WIDE	Running time (second)						
(*n*, *m*)	PAA2	PSA	DINIC	DF	HI_PR	FS1	FMAP
(65025, 314840)	130.57	303.73	*	8.706	0.997	65.64	540.66
(123210, 599289)	549.04	1099.8	*	25.18	2.397	240.6	2506.8
(259308, 1267875)	2464.14	4612.1	*	103.3	7.409	970.7	15309

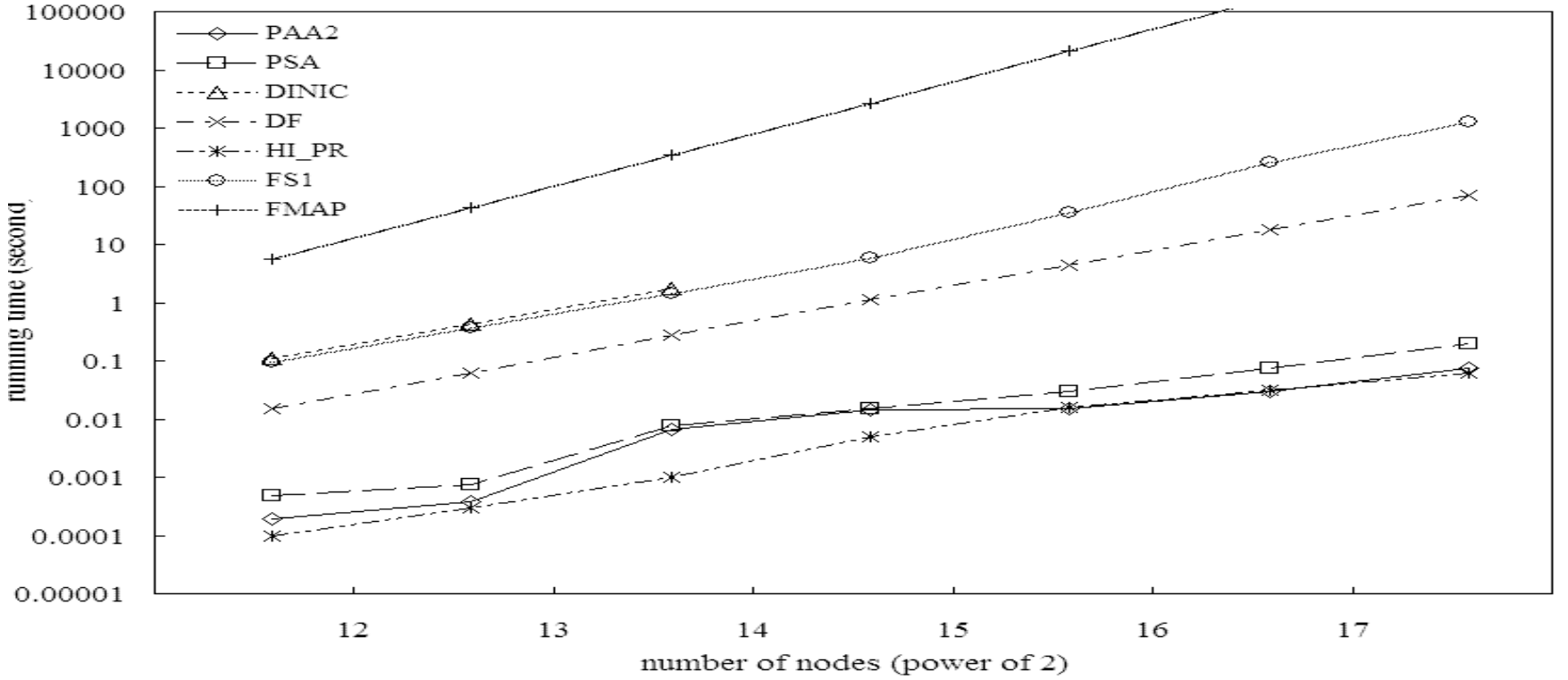
Computational Results(9/11)

AK



AK	Running time (second)						
(*n*, *m*)	PAA2	PSA	DINIC	DF	HI_PR	FS1	FMAP
(65542, 98311)	296.87	83.172	*	80.44	8.61	16.19	4.672
(131078, 196615)	1376.43	405.33	*	451.1	34.313	130.4	18.61
(262150, 393223)	5338.71	1582.5	*	2224	138.45	616.8	74.39

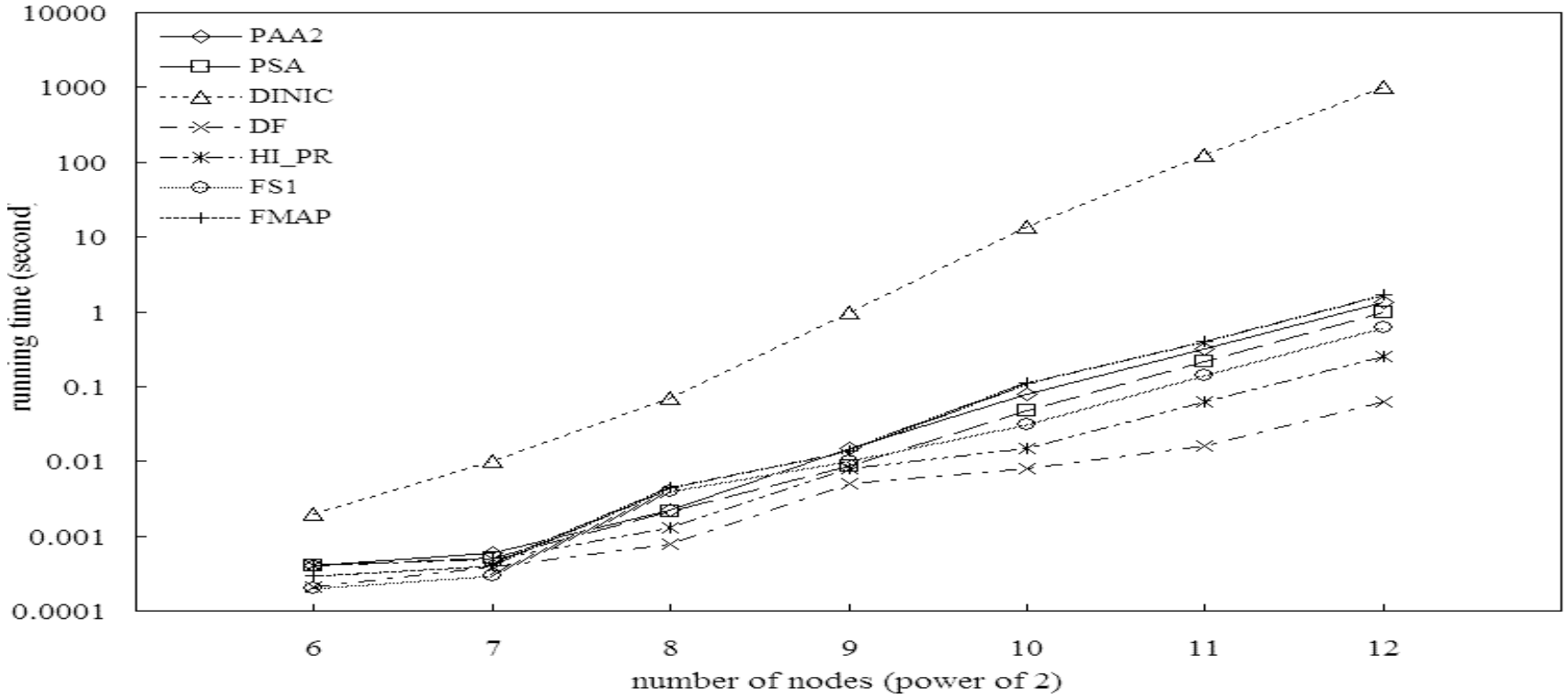
Computational Results(10/11)
WAS-10



WAS-10	Running time (second)						
(*n*, *m*)	PAA2	PSA	DINIC	DF	HI_PR	FS1	FMAP
(49155, 65537)	0.015	0.03	*	4.42	0.016	35.22	21443
(98307, 131073)	0.029	0.074	*	17.73	0.031	263.3	170707
(196611, 262145)	0.074	0.193	*	71.05	0.063	1282	1314449

Computational Results(11/11)

ACU



ACU	Running time (second)						
(*n*, *m*)	PAA2	PSA	DINIC	DF	HI_PR	FS1	FMAP
(1024, 523776)	0.078	0.047	13.5	0.008	0.015	0.031	0.11
(2048, 2096128)	0.313	0.234	124	0.016	0.063	0.141	0.406
(4096, 8386560)	1.36	1.182	1027.1	0.063	0.25	0.61	1.69

Performance Summary(1/2)

fast ⟷ **slow**

- G-LONG

  HI_PR << FS1 < DF << FMAP < DINIC < PSA << PAA2

- G-WIDE

  HI_PR << DF << FS1 < PAA2 < DINIC < PSA << FMAP

- AK

  FMAP < HI_PR << FS1 < PSA < DF < PAA2 << DINIC

- WAS-10

  PAA2 ≒ HI_PR < PSA << DF << FS1 < DINIC << FMAP

- ACU

  DF < HI_PR < FS1 < PSA < PAA2 < FMAP << DINIC

Performance Summary(2/2)

- HI_PR performs the best in most cases
- PAA2 performs the best in WAS-10
  - Only one time of augmentation
- In ACU family
  - PAA2 and PSA only need two times of augmentation

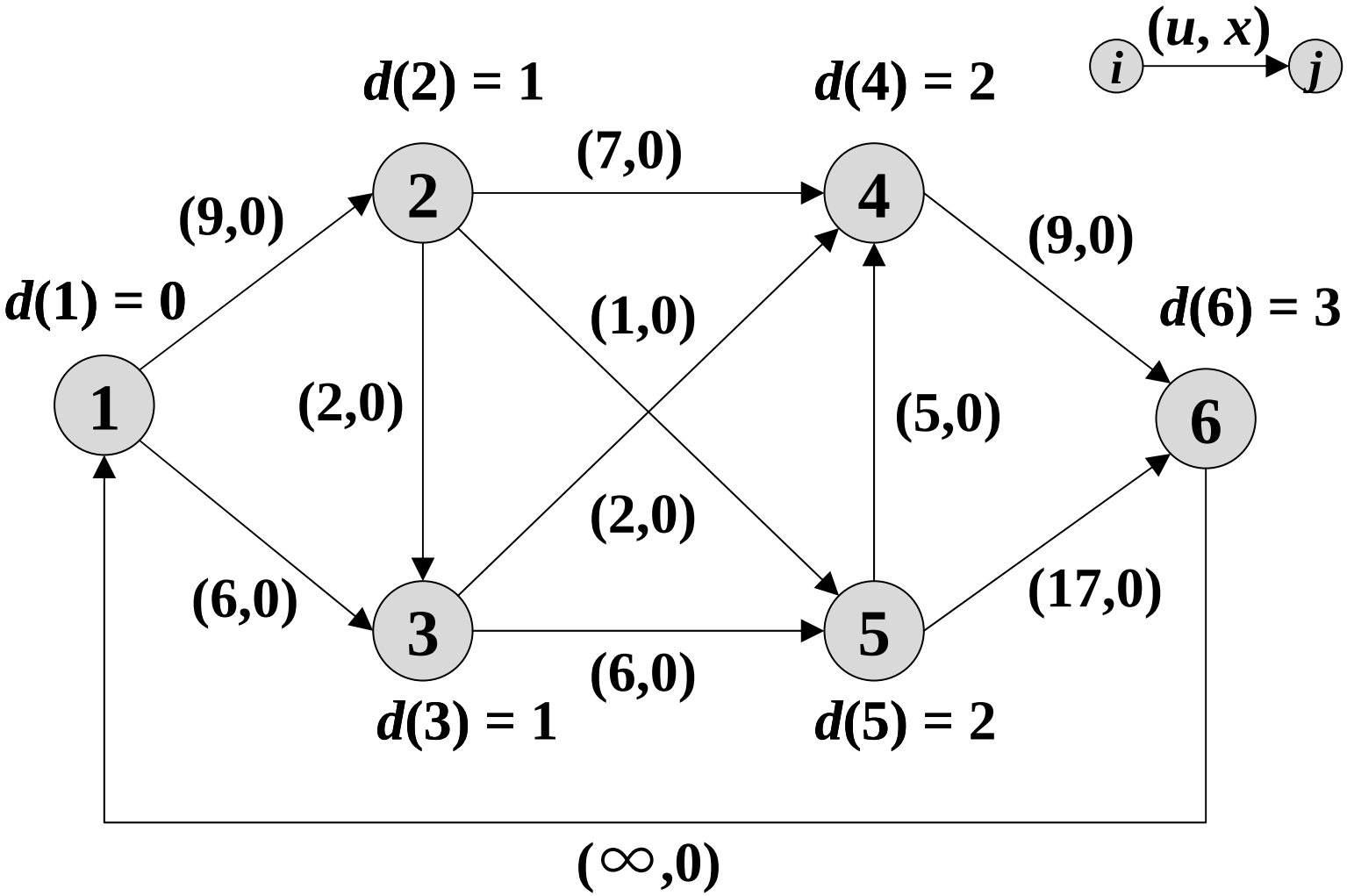
MLSDP Algorithm(1/2)

- LSPD method to solve the min-cost problem
  - Nondegenerate pivot by solving NNLS subproblem
- Max-flow problem → min-cost problem by adding an artificial arc (t, s) with infinite capacity
- LSDP for the max-flow problem
  - NNLS subproblem can be solved by Kirchhoff's laws to obtain improving direction δ
- LSDP for max-flow in layered networks
  - Polynomial-time algorithm

MLSDP Algorithm(2/2)

- Use BFS and distance labels to construct a layered network
- Treating each admissible arc as 1Ω, calculate the electronic flow δ_{ij} by KCL/KVL
- Calculate maximum step length θ
- Augment flows by $\Delta x = \delta \times \theta$
- Saturate at least one arc after each augmenting iteration
- Theoretical Complexity：$O(nm^4)$

MLSDP Example(1/7)



Kirchhoff's law

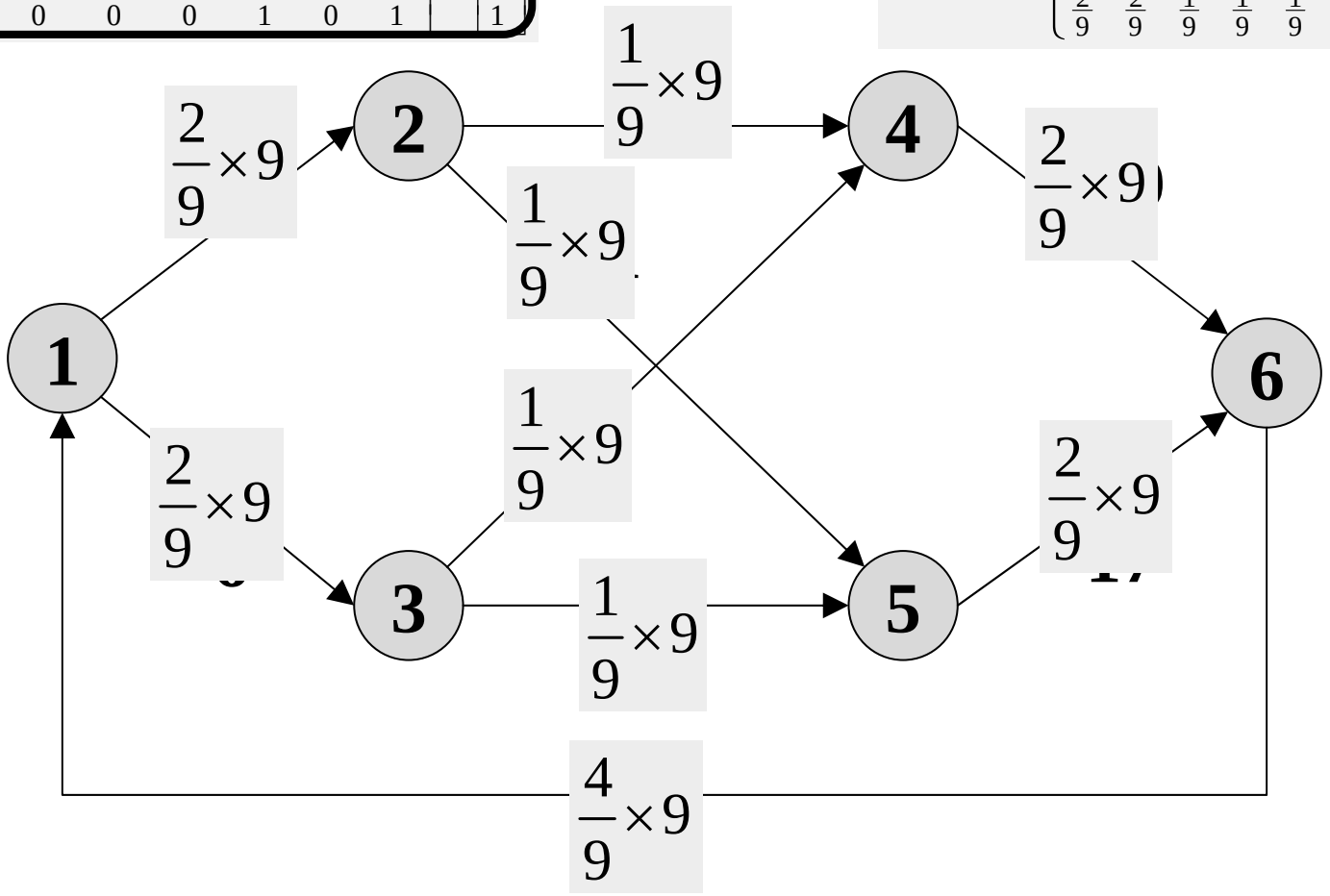
$$\left[\begin{array}{ccccccccc|c|c}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & & 0 \\
0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & & 0 \\
0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & = & 0 \\
\hline
-1 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & & 0 \\
-1 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & & 0 \\
-1 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & & 1
\end{array}\right]$$

KCL

KVL

$$(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7, \delta_8, \delta_9) = \left(\frac{2}{9}, \frac{2}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{2}{9}, \frac{2}{9}, \frac{4}{9}\right)$$

$$\theta = \min\left\{\frac{9}{\frac{2}{9}}, \frac{6}{\frac{2}{9}}, \frac{7}{\frac{1}{9}}, \frac{1}{\frac{1}{9}}, \frac{2}{\frac{1}{9}}, \frac{6}{\frac{1}{9}}, \frac{9}{\frac{2}{9}}, \frac{17}{\frac{2}{9}}\right\} = 9$$



Kirchhoff's law

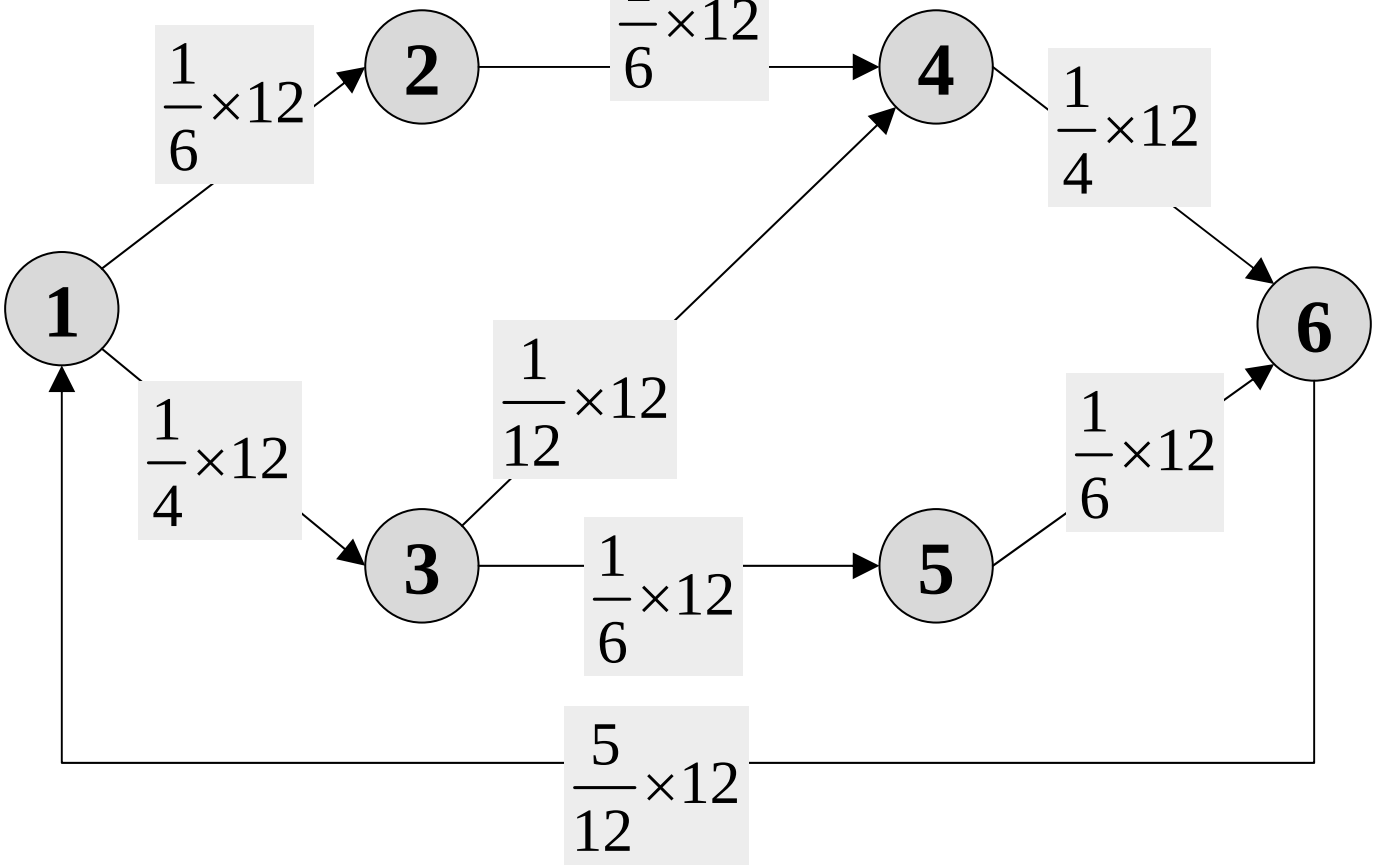
$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 1 & -1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

KCL

KVL

$$(\delta_1,\delta_2,\delta_3,\delta_4,\delta_5,\delta_6,\delta_7,\delta_8) = \left(\frac{1}{6},\frac{1}{4},\frac{1}{6},\frac{1}{12},\frac{1}{6},\frac{1}{4},\frac{1}{6},\frac{5}{12}\right)$$

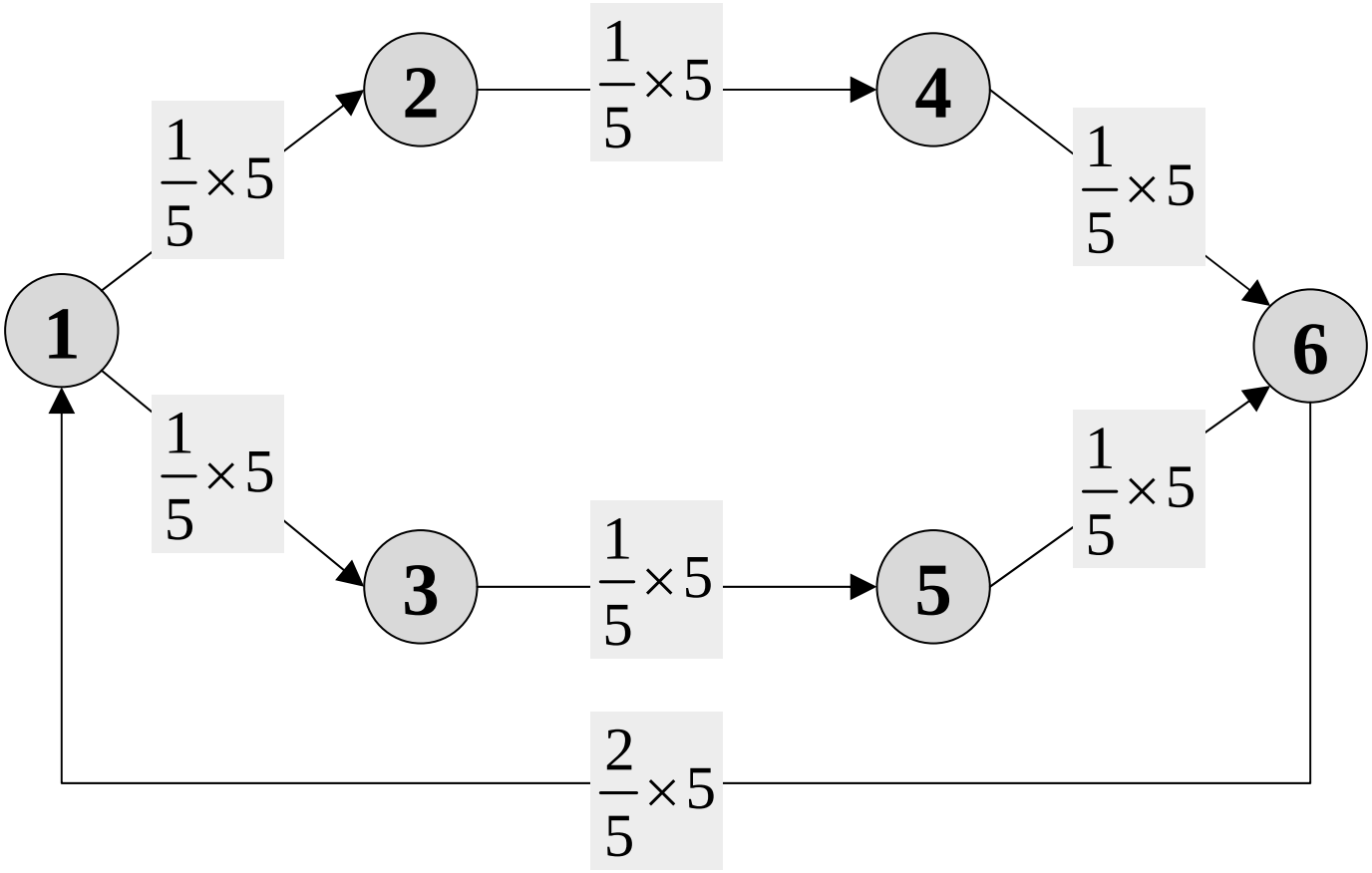
$$\theta = \min\left\{\frac{7}{\frac{1}{6}},\frac{4}{\frac{1}{4}},\frac{6}{\frac{1}{6}},\frac{1}{\frac{1}{12}},\frac{5}{\frac{1}{6}},\frac{7}{\frac{1}{4}},\frac{15}{\frac{1}{6}}\right\} = 12$$



MLSDP Example(4/7)

$$(\delta_1,\delta_2,\delta_3,\delta_4,\delta_5,\delta_6,\delta_7)=\left(\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{2}{5}\right)$$

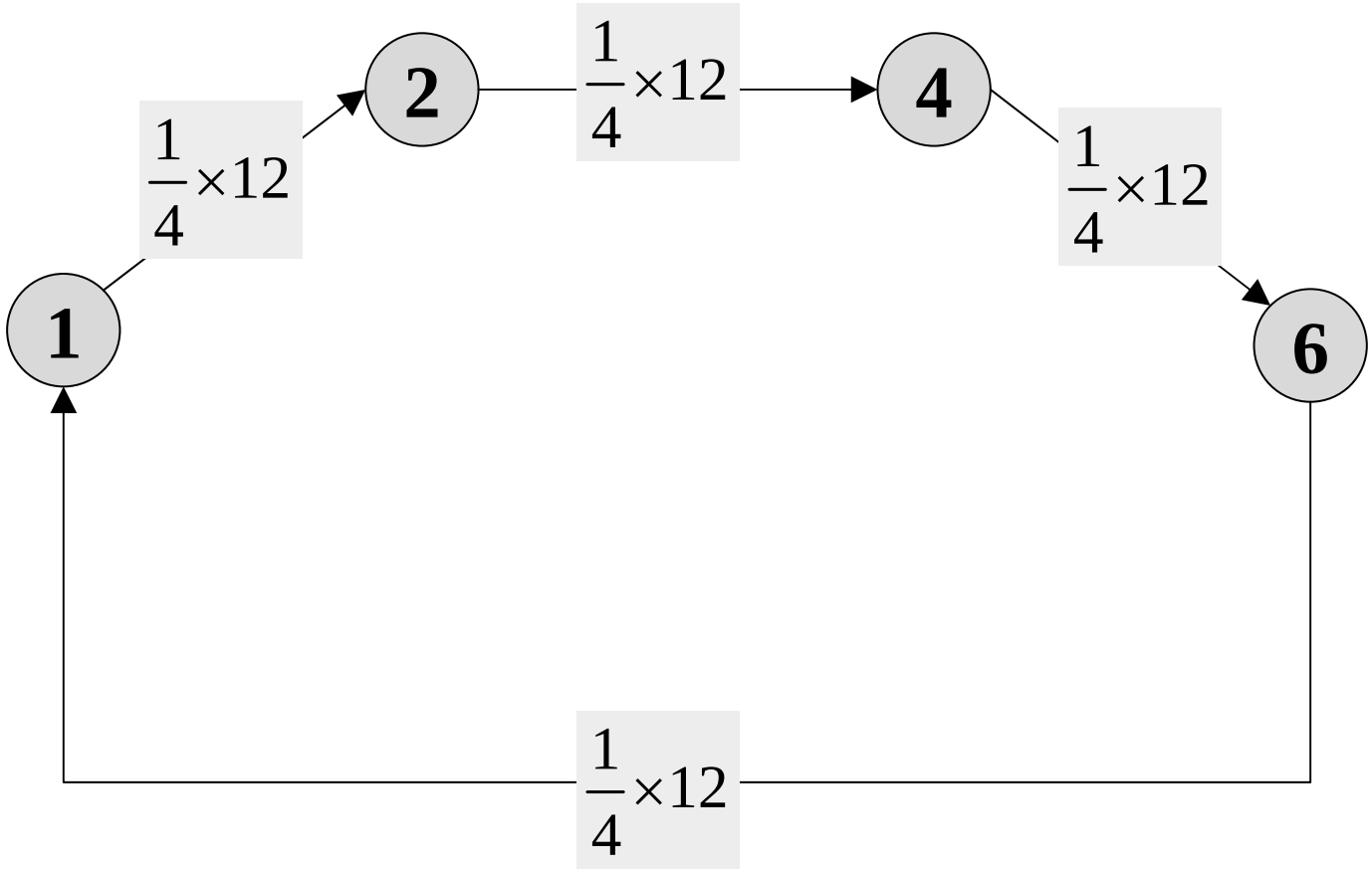
$$\theta=\min\left\{\frac{5}{\frac{1}{5}},\frac{1}{\frac{1}{5}},\frac{4}{\frac{1}{5}},\frac{3}{\frac{1}{5}},\frac{4}{\frac{1}{5}},\frac{11}{\frac{1}{5}}\right\}=5$$



MLSDP Example(5/7)

$$(\delta_1, \delta_2, \delta_3, \delta_4) = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$$

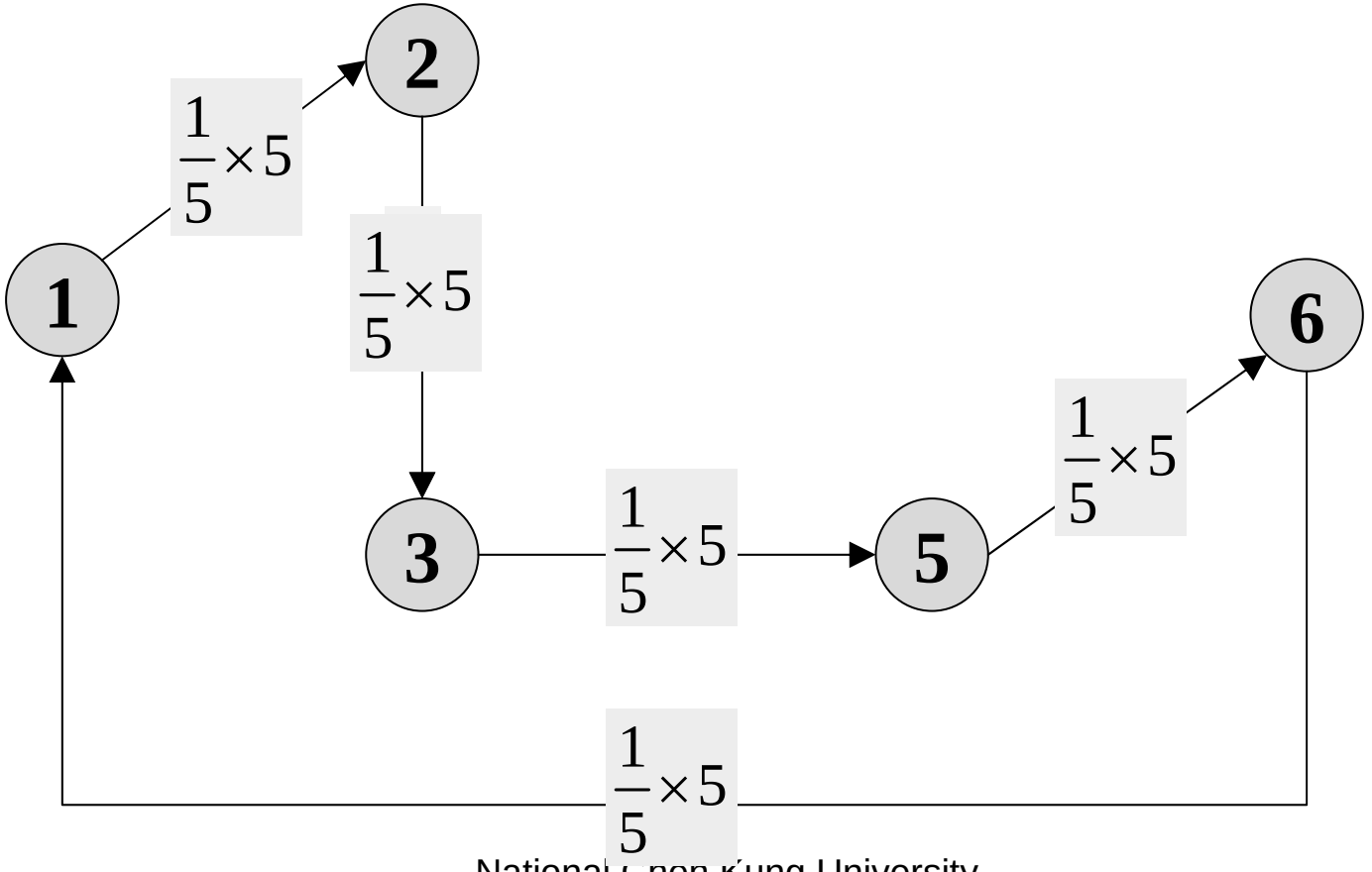
$$\theta = \min\left\{\frac{4}{\frac{1}{4}}, \frac{3}{\frac{1}{4}}, \frac{3}{\frac{1}{4}}\right\} = 12$$



MLSDP Example(6/7)

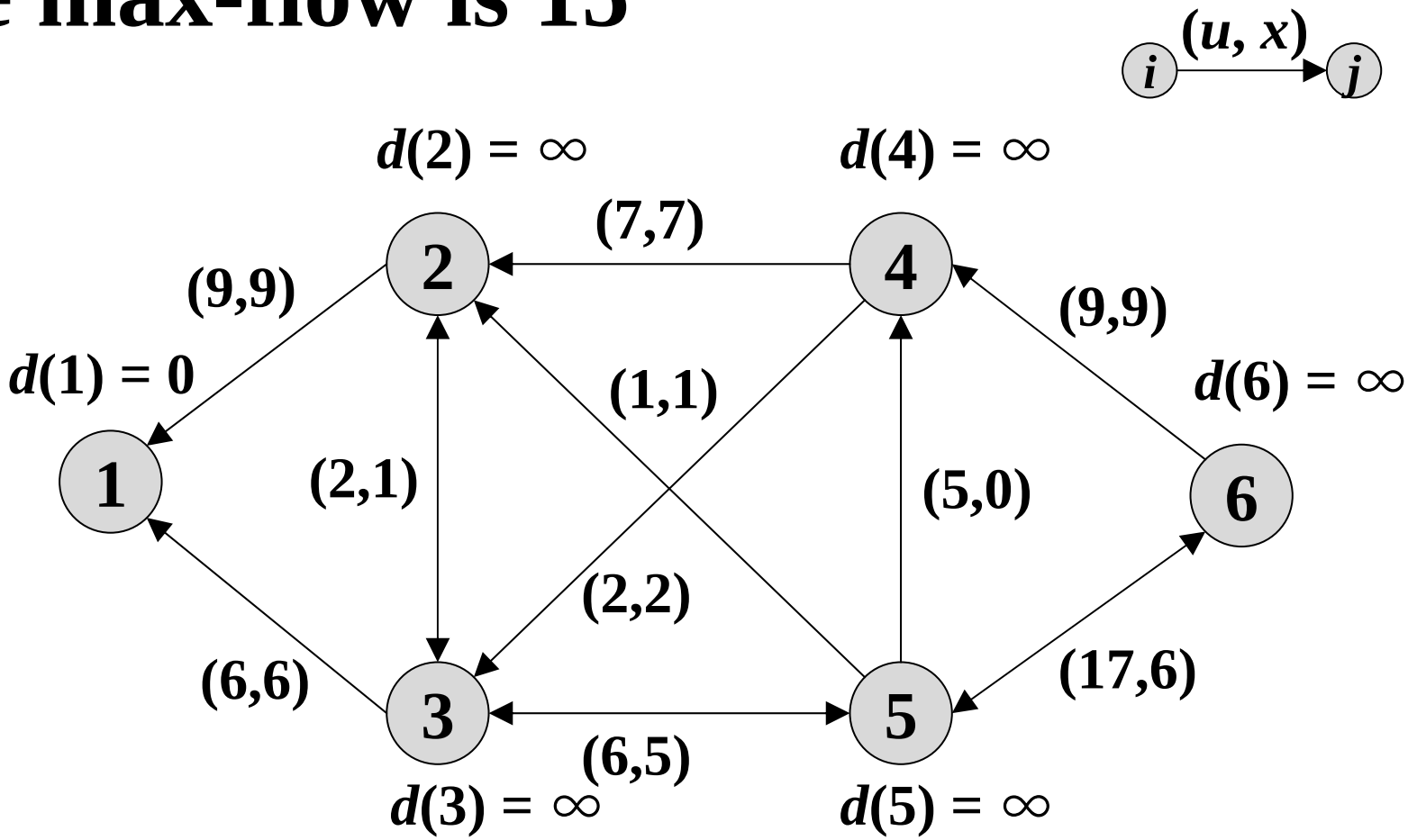
$$(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5) = \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$$

$$\theta = \min\left\{\frac{1}{\frac{1}{5}}, \frac{2}{\frac{1}{5}}, \frac{2}{\frac{1}{5}}, \frac{12}{\frac{1}{5}}\right\} = 5$$



MLSDP Example(7/7)

The max-flow is 15



Complexity Analysis

- The MLSDP algorithm constructs at most n-1 layered networks in the residual network
- Find the max-flow in a given layered network : $O(m^4)$
  - Calculate electron flow δ by Kirchhoff's law : $O(m^3)$
  - Augment flow : $O(m)$
  - Remove at least one arc after augmenting flow
- Theoretical complexity of MLSDP : $O(nm^4)$

Speed-up Techniques

- KCL/KVL form a sparse linear system
  - Using UMFPACK sparse linear solver instead of Gauss-Jordan elimination
- Improving the $O(m^3)$ complexity for KCL/KVL?
  - Flow decomposition? Only path flow, NO cycle flow?
  - q path flows, $O(q^3)$? ➔ future research!
    - $q = m - n + 1$ (KVL)

Computational Results(1/2)

- Test algorithms
  - PAA2
  - MLSDP using Gauss-Jordan elimination
  - MLSDP using UMFPACK (MLSDP2)
- Tested network families
  - G-LONG & G-WIDE
  - AK
  - WAS-10
  - ACU

Computational Results(2/2)

PAA and MLSDP

Network		Running time (second)			# of augmentation	
Family	(*n*, *m*)	PAA2	MLSDP	MLSDP2	PAA2	MLSDP
G-LONG	(756, 3240)	0.026	1503.65	30.041	486	758
	(1225, 5376)	0.067	10418.3	106.161	862.4	1289.8
G-WIDE	(676, 3003)	0.02	547.1	10.716	462.8	660.2
	(1024, 4608)	0.062	4018.27	61.473	737.2	1612.4
AK	(2054, 3079)	0.281	263.703	20.968	1026	770
	(4102, 6151)	1.094	3947.48	165.327	2050	1538
WAS-10	(1539, 2049)	0.008	0.282	0.093	1	1
	(3075, 4097)	0.016	1.171	0.407	1	1
ACU	(512, 130816)	0.016	2.579	0.047	2	2
	(1024, 523776)	0.079	20.344	0.234	2	2

Contributions(1/2)

- PAA proportionally distribute flows to arcs in an admissible subnetwork
- Theoretical complexity of PAA : $O(n^2m)$
- Implementations of PAA
  - Advanced data structure
  - Forward and backward augmentation
  - Scaling phase
- FSA fairly distribute flows to arcs in an admissible subnetwork
- Theoretical complexity of FSA : $O(nm^2)$

Contributions(2/2)

- The MLSDP algorithm can be bounded in $O(nm^4)$ time
- Use UMFPACK sparse linear solver
- Converting fractional flows to integral flows in $O(m^2)$ time
- FSA and MLSDP have fewer iterations than PAA in AK family
- PAA may be suitable for WAS-10 、ACU or symmetric networks

Future Research

- Augment flows along at most *k* outgoing arcs in PAA algorithm
- PAA for fair networks
- Path flow augmentations in MLSDP

NCKU IIM

Thank you for listening

Q & A