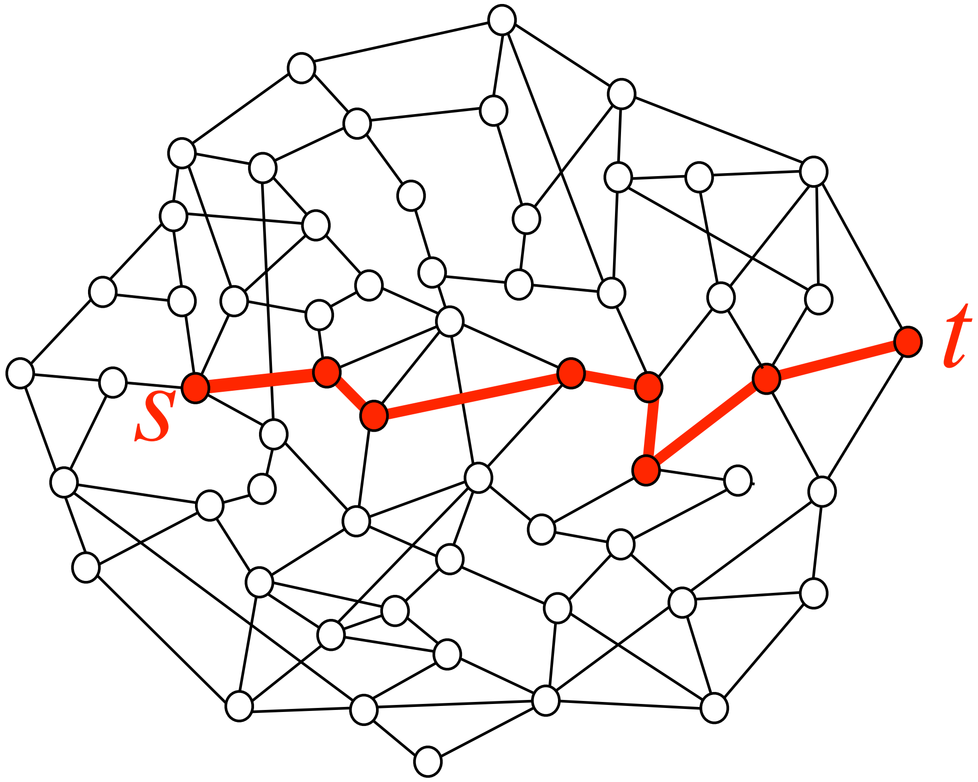
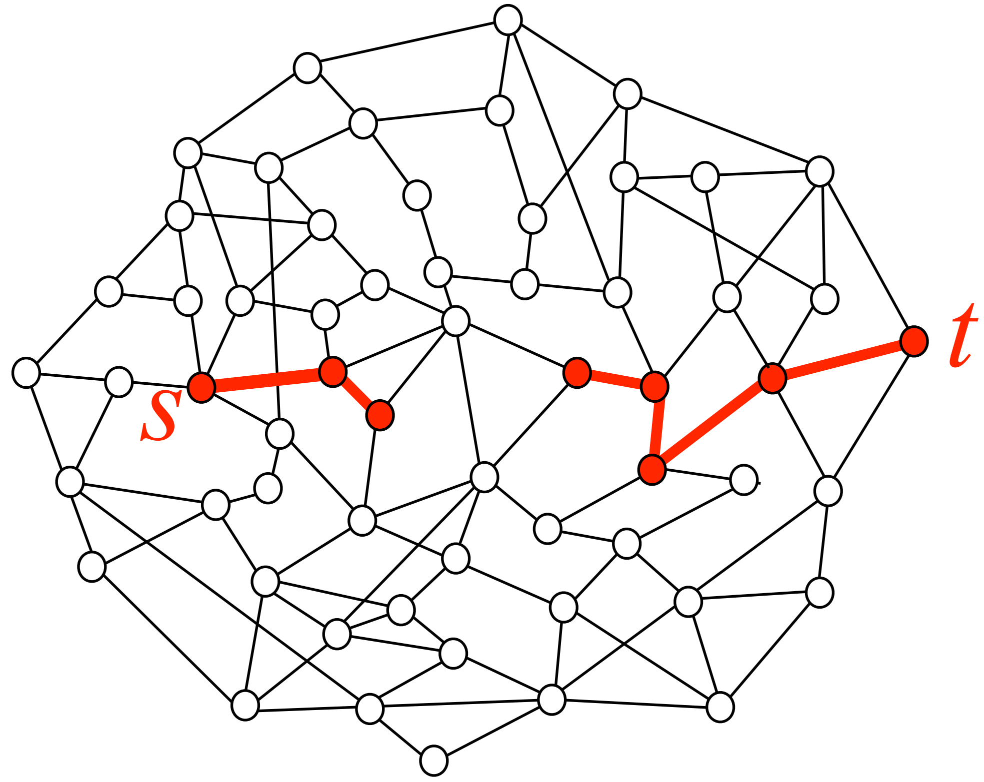
The Replacement Paths Problem



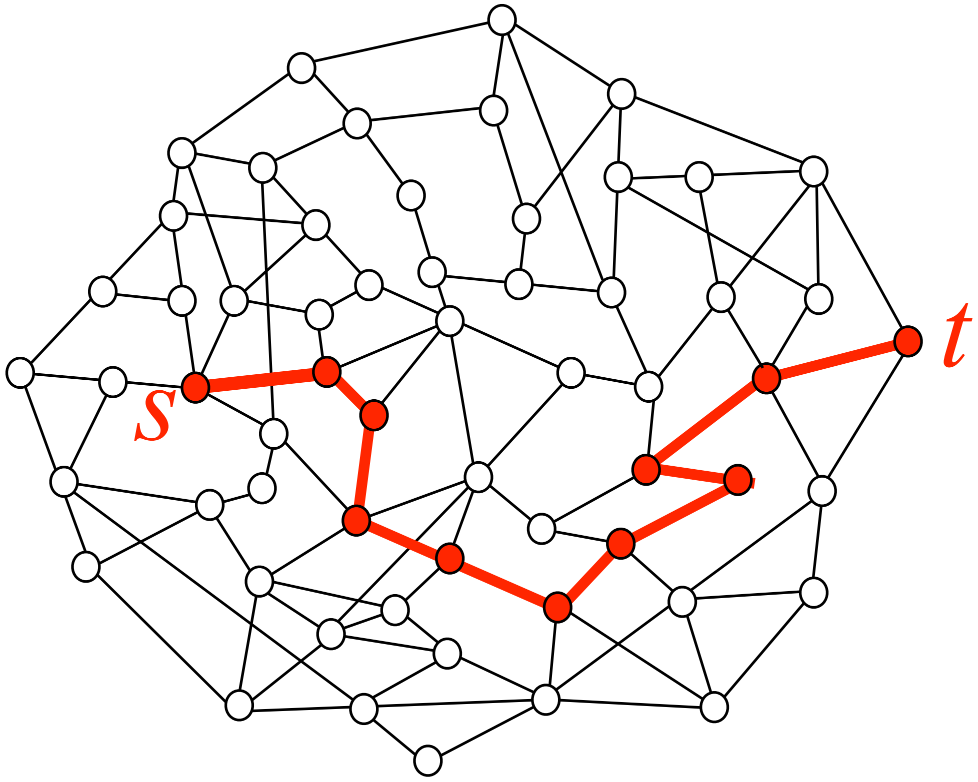
Oren Weimann

Raphy Yuster

The Replacement Paths Problem



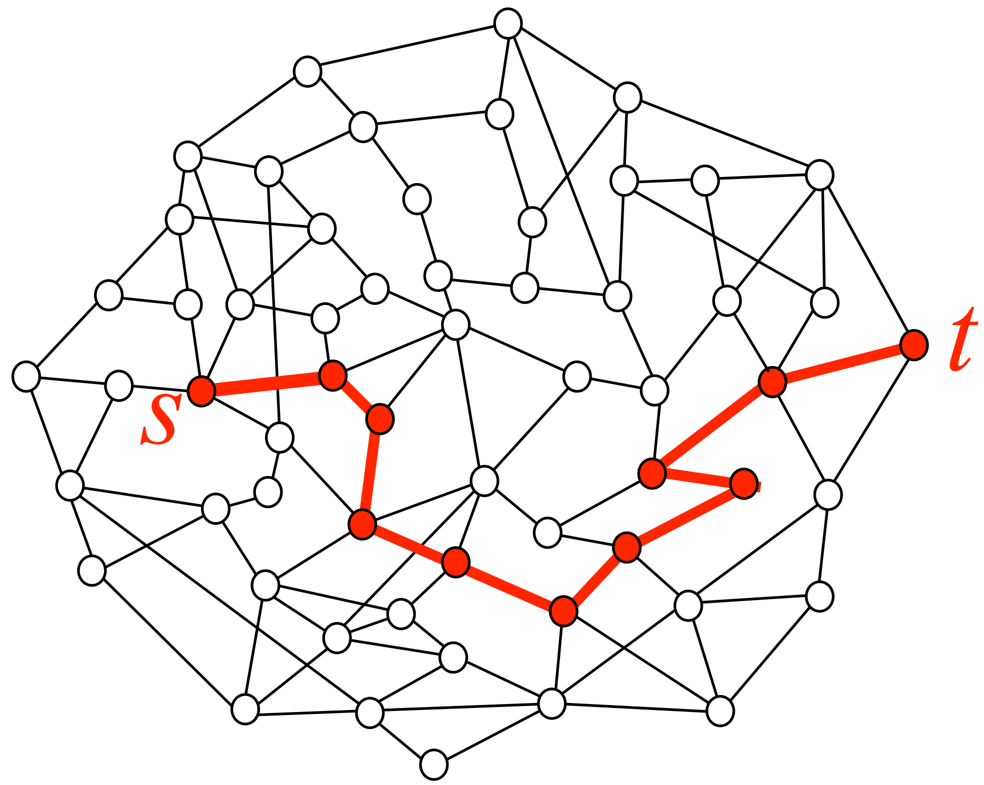
The Replacement Paths Problem



Oren Weimann

Raphy Yuster

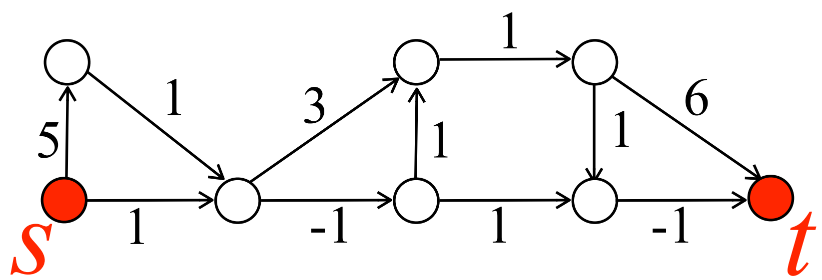
The Replacement Paths Problem



Vickrey pricing of edges owned by selfish agents

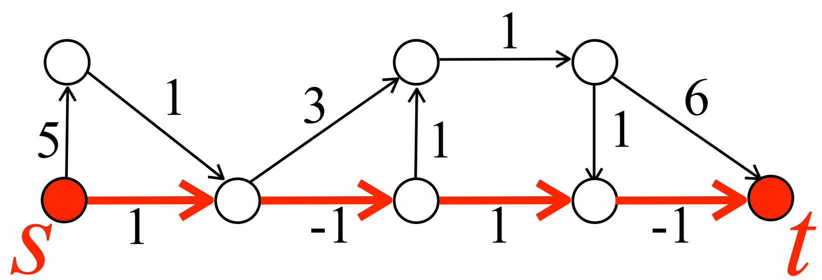
The Replacement Paths Problem

- General directed graphs
- Positive and negative lengths



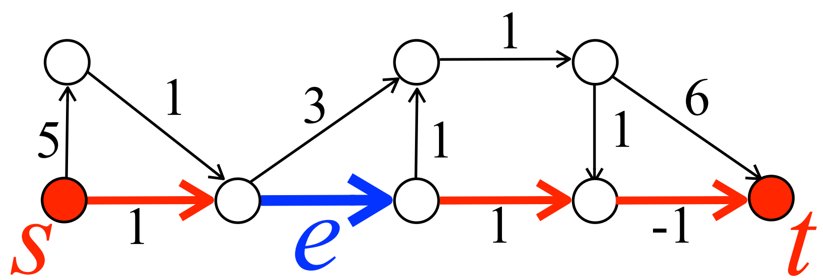
The Replacement Paths Problem

- General directed graphs
- Positive and negative lengths



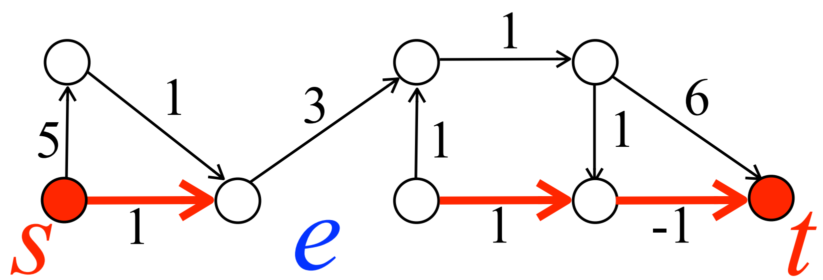
The Replacement Paths Problem

- General directed graphs
- Positive and negative lengths



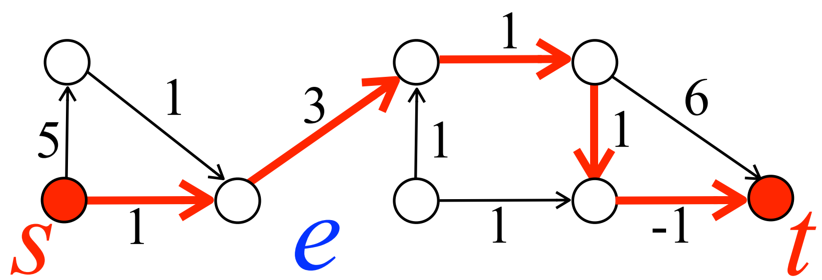
The Replacement Paths Problem

- General directed graphs
- Positive and negative lengths

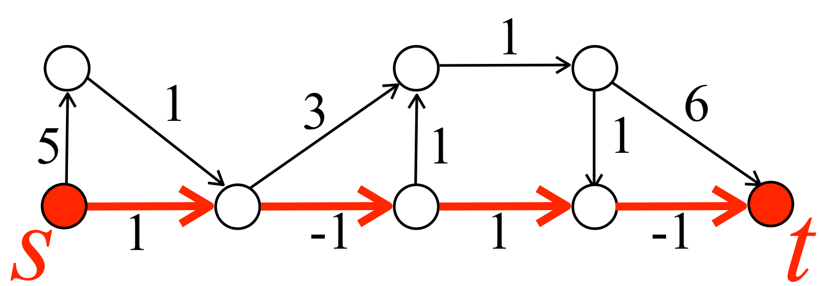


The Replacement Paths Problem

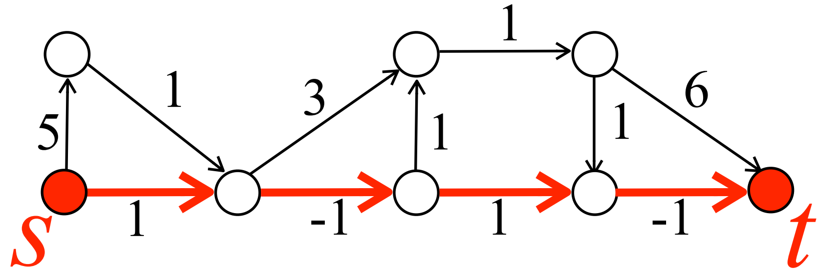
- General directed graphs
- Positive and negative lengths

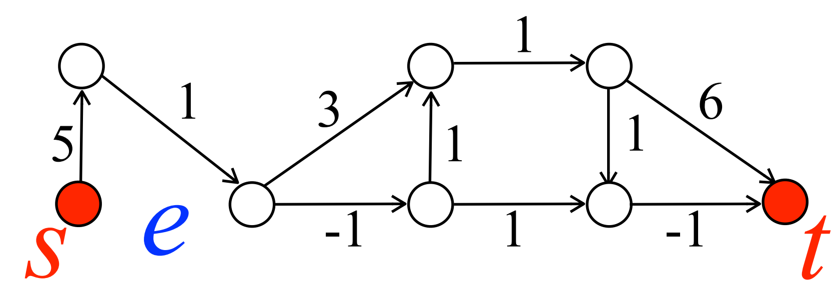


A Naive Solution

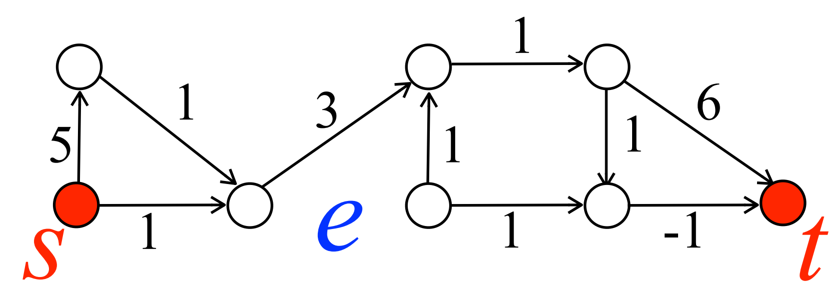


A Naive Solution

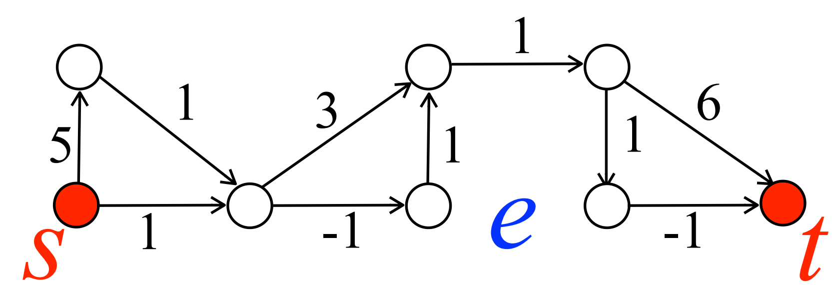




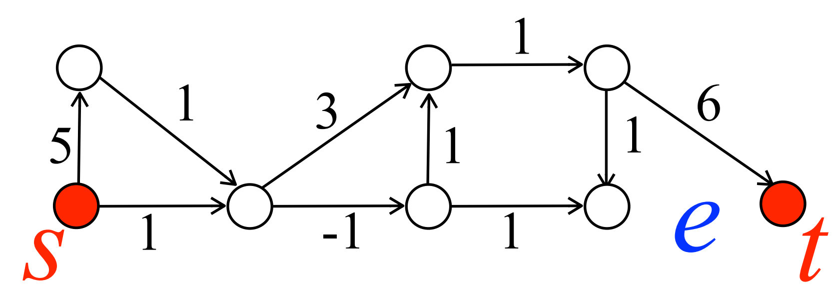
(1)



(2)

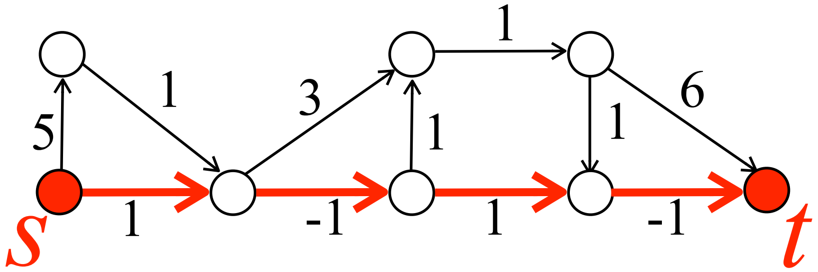


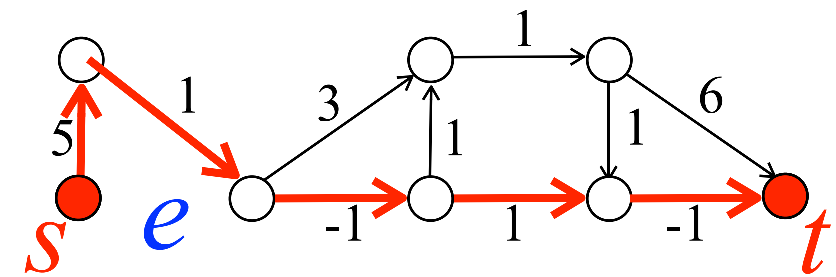
(3)



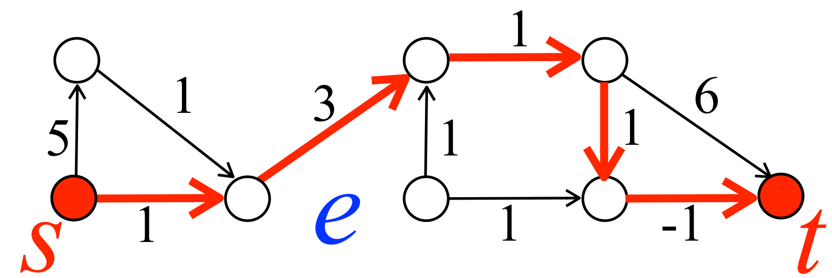
(n)

A Naive Solution

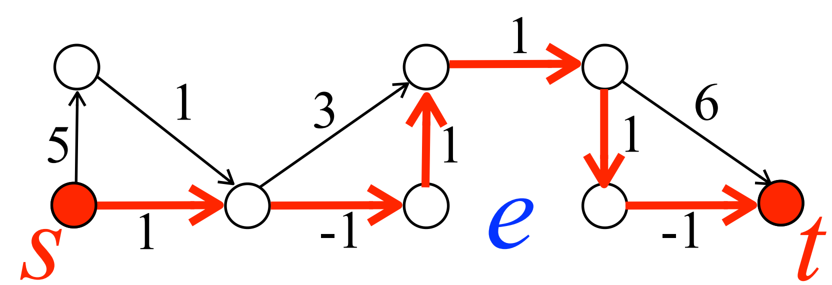




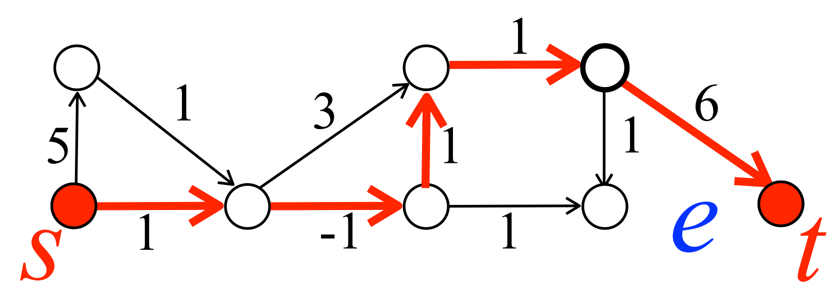
(1)



(2)



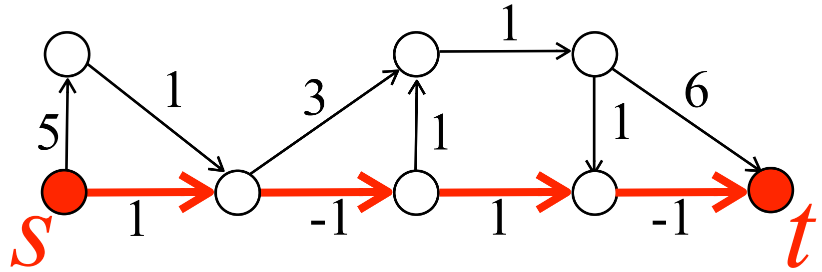
(3)

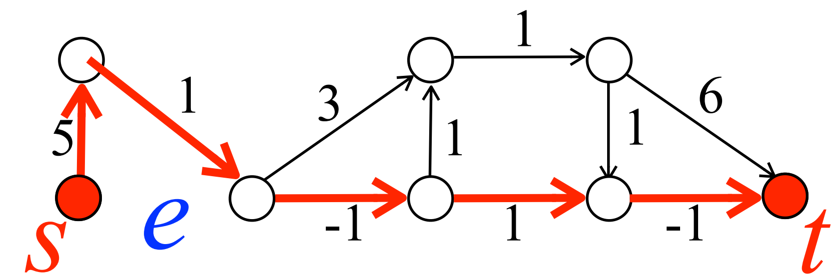


(n)

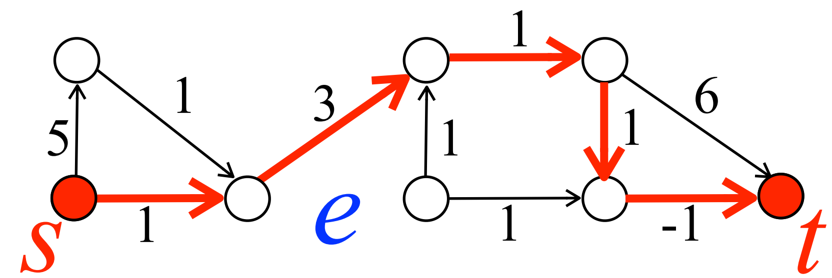
A Naive Solution

$$O(nm + n^2 \log n) = O(n^3)$$

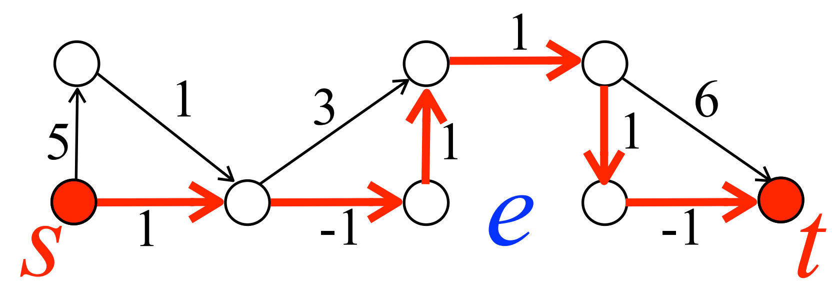




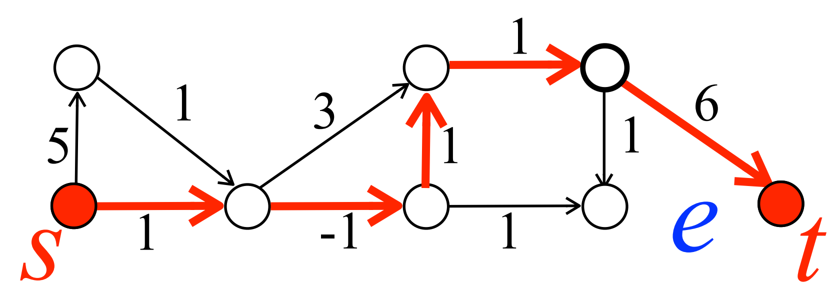
(1)



(2)



(3)



(n)

Related Work

Directed weighted graphs:

- $O(nm + n^2 \log n)$ - naive solution
- $O(nm + n^2 \log\log n)$ - [Gotthilf, Lewenstein 2009]
- $\Omega(m\sqrt{n})$ lower bound - [Hershberger 2003]
- $\tilde{O}(m \log(nM)/\varepsilon)$ for $(1+\varepsilon)$-approximation - [Bernstein 2010]
- $\tilde{O}(Mn^{1+2\omega/3}) = O(Mn^{2.584})$ for weights in $\{-M,\ldots,M\}$ - [Yuster, W.]

Related Work

Directed weighted graphs:

- $O(nm + n^2 \log n)$ - naive solution
- $O(nm + n^2 \log\log n)$ - [Gotthilf, Lewenstein 2009]
- $\Omega(m\sqrt{n})$ lower bound - [Hershberger 2003]
- $\tilde{O}(m \log(nM)/\varepsilon)$ for $(1+\varepsilon)$-approximation - [Bernstein 2010]
- $\tilde{O}(Mn^{1+2\omega/3}) = O(Mn^{2.584})$ for weights in $\{-M,\ldots,M\}$ - [Yuster, W.]

Undirected graphs:

- $O(m + n \log n)$ - [Malik, Mittal, Gupta 1989]
- $O(m\alpha(n,m))$ - [Nardelli, Proietti, Widmayer 2001]

Related Work

Directed weighted graphs:

- $O(nm + n^2 \log n)$ - naive solution
- $O(nm + n^2 \log\log n)$ - [Gotthilf, Lewenstein 2009]
- $\Omega(m\sqrt{n})$ lower bound - [Hershberger 2003]
- $\tilde{O}(m \log(nM)/\varepsilon)$ for $(1+\varepsilon)$-approximation - [Bernstein 2010]
- $\tilde{O}(Mn^{1+2\omega/3}) = O(Mn^{2.584})$ for weights in $\{-M,\ldots,M\}$ - [Yuster, W.]

Undirected graphs:

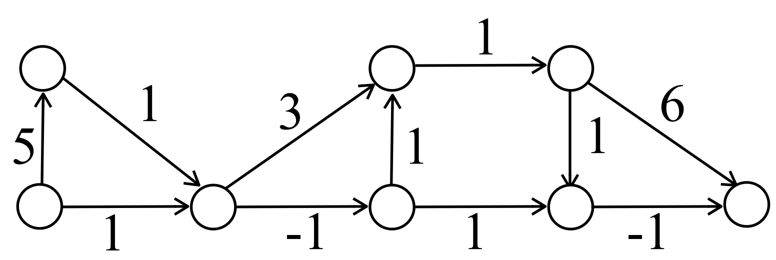
- $O(m + n \log n)$ - [Malik, Mittal, Gupta 1989]
- $O(m\alpha(n,m))$ - [Nardelli, Proietti, Widmayer 2001]

Unweighted graphs:

- $\tilde{O}(m\sqrt{n})$ - [Roditty and Zwick 2005]

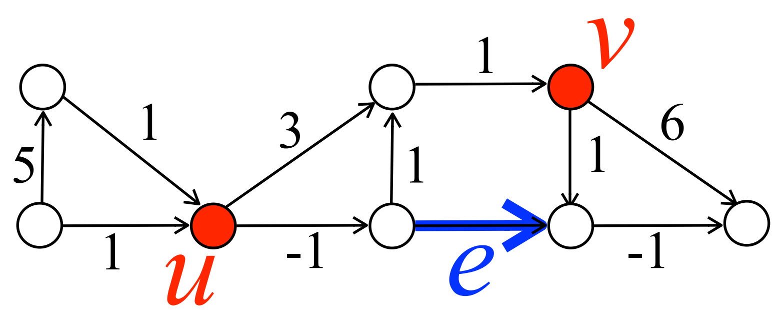
The Replacement Paths Problem

All Pairs



The Replacement Paths Problem

All Pairs

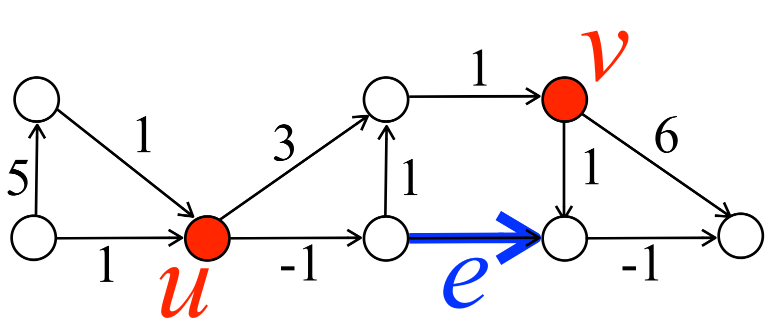


Query: (u,v,e)?

The Replacement Paths Problem

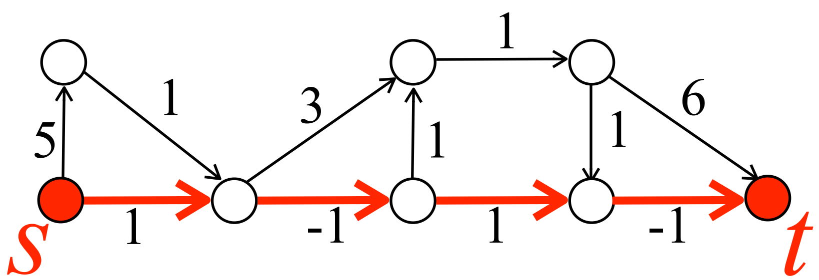
All Pairs

- $\tilde{O}(mn^2)$, constant query time
  [Demetrescu, Thorup, Chowdhury,, Ramachandran 2008]
- $\tilde{O}(mn)$, constant query time
  [Bernstein, Karger 2009]
- Poly., $O(\log n)$ query time, 2 failures
  [Duan, Pettie 2009]
- Poly., $\tilde{O}(f)$ query time, f failures
  constant-approximation
  [Chechick, Langberg, Peleg, Roditty 2009]
- $\tilde{O}(Mn^{1+\omega-\alpha})$, $\tilde{O}(n^{1+\alpha})$ query time, f failures
  [Yuster, W.]



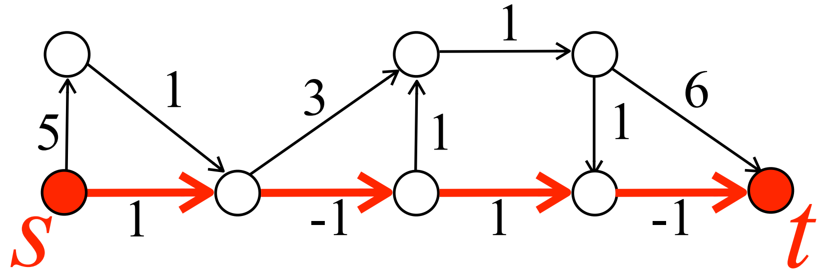
Query: (u,v,e)?

A Replacement Paths Algorithm



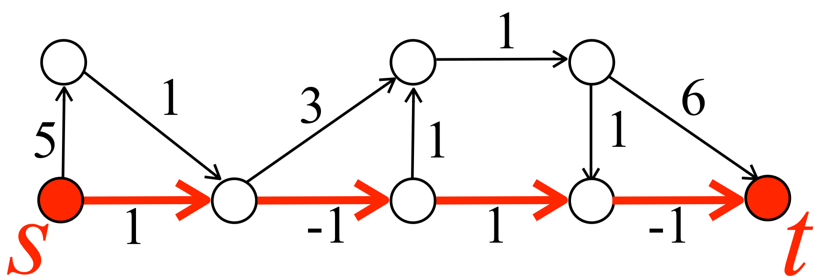
A Replacement Paths Algorithm

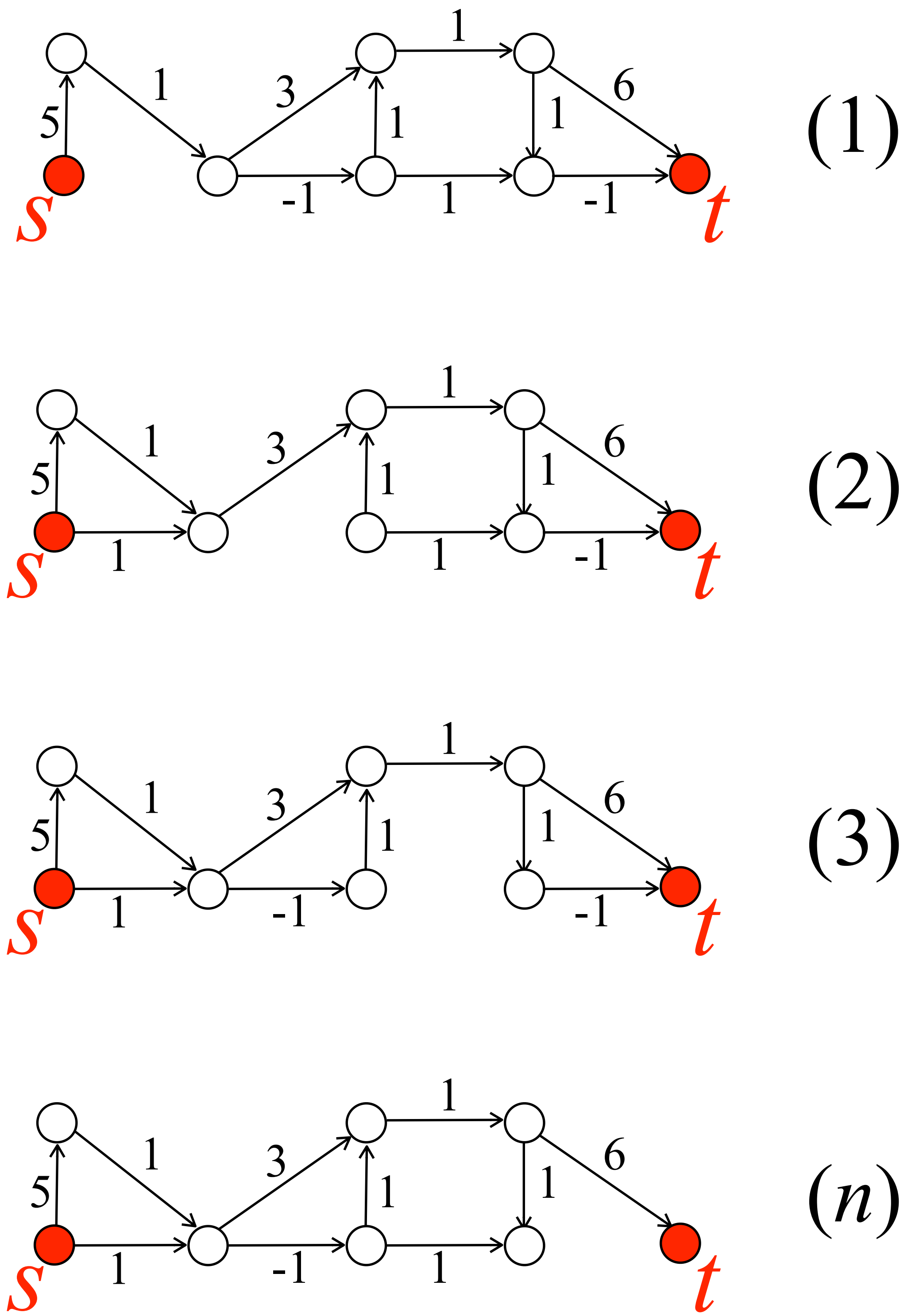
- Success probability 1-1/n



A Replacement Paths Algorithm

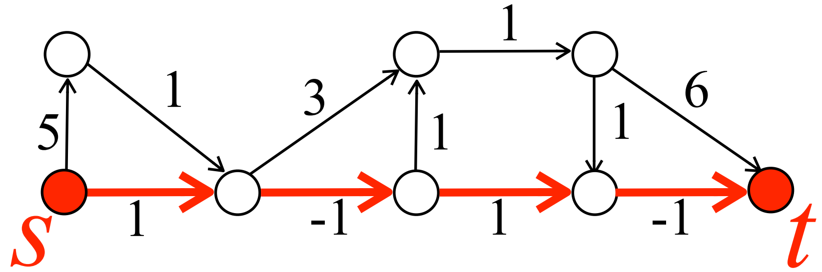
- Success probability 1-1/n





A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$



(1)

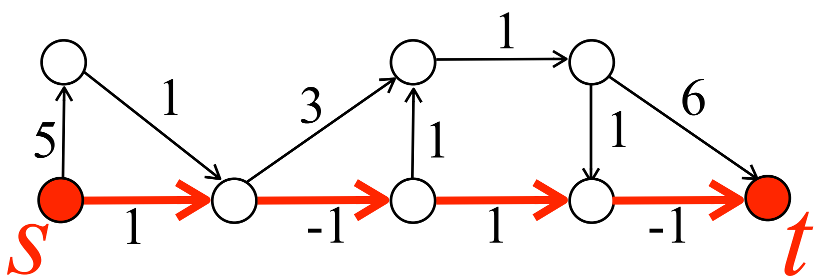
(2)

(3)

(r)

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$



Remove every edge
with probability $n^{\alpha - 1}$

(1)

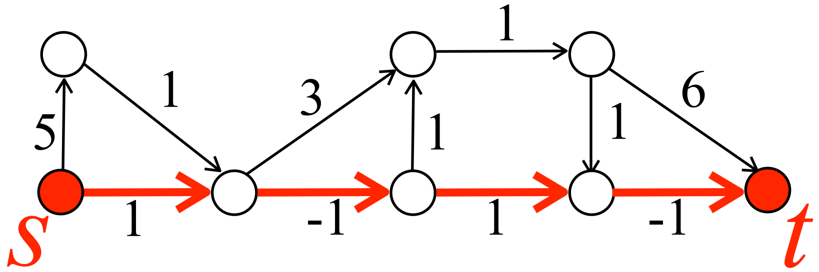
(2)

(3)

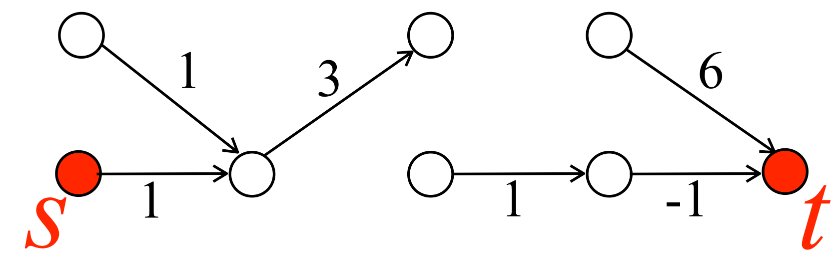
(r)

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$



Remove every edge
with probability $n^{\alpha - 1}$



(1)

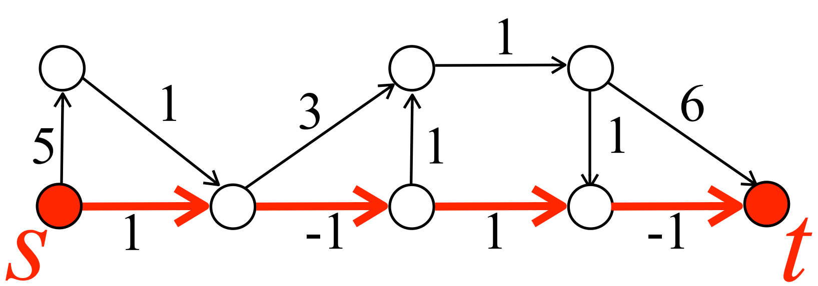
(2)

(3)

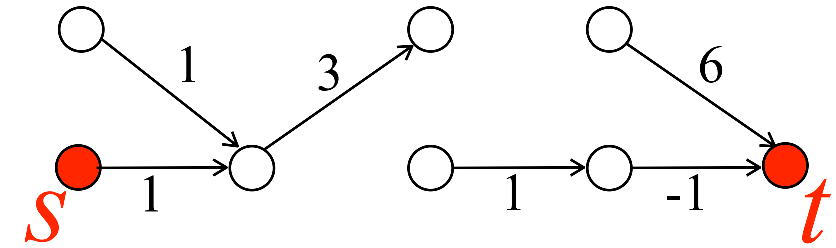
(r)

A Replacement Paths Algorithm

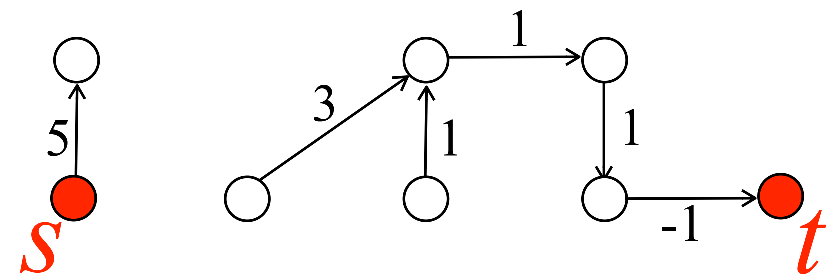
- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$



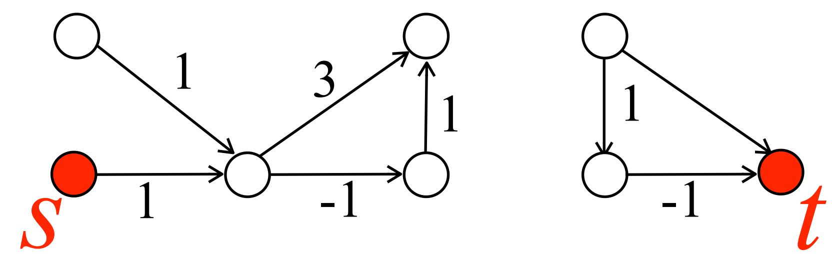
Remove every edge with probability $n^{\alpha - 1}$



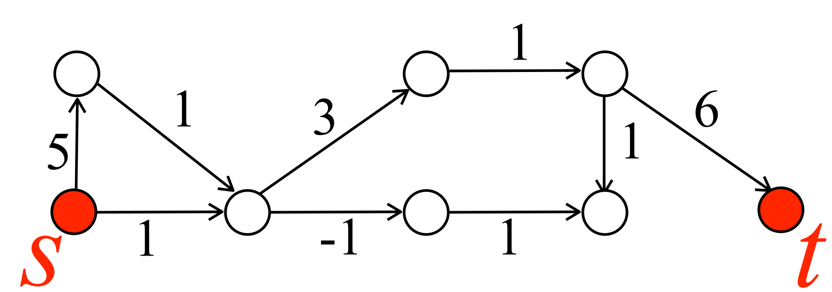
(1)



(2)



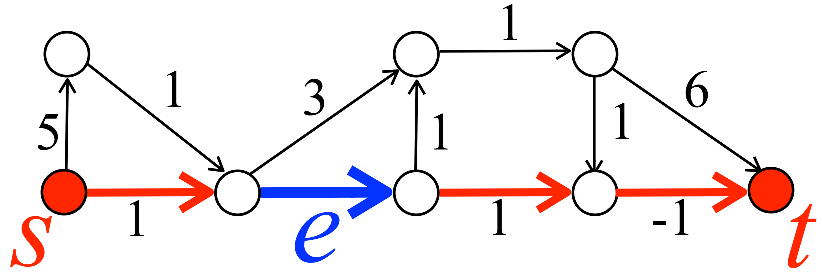
(3)

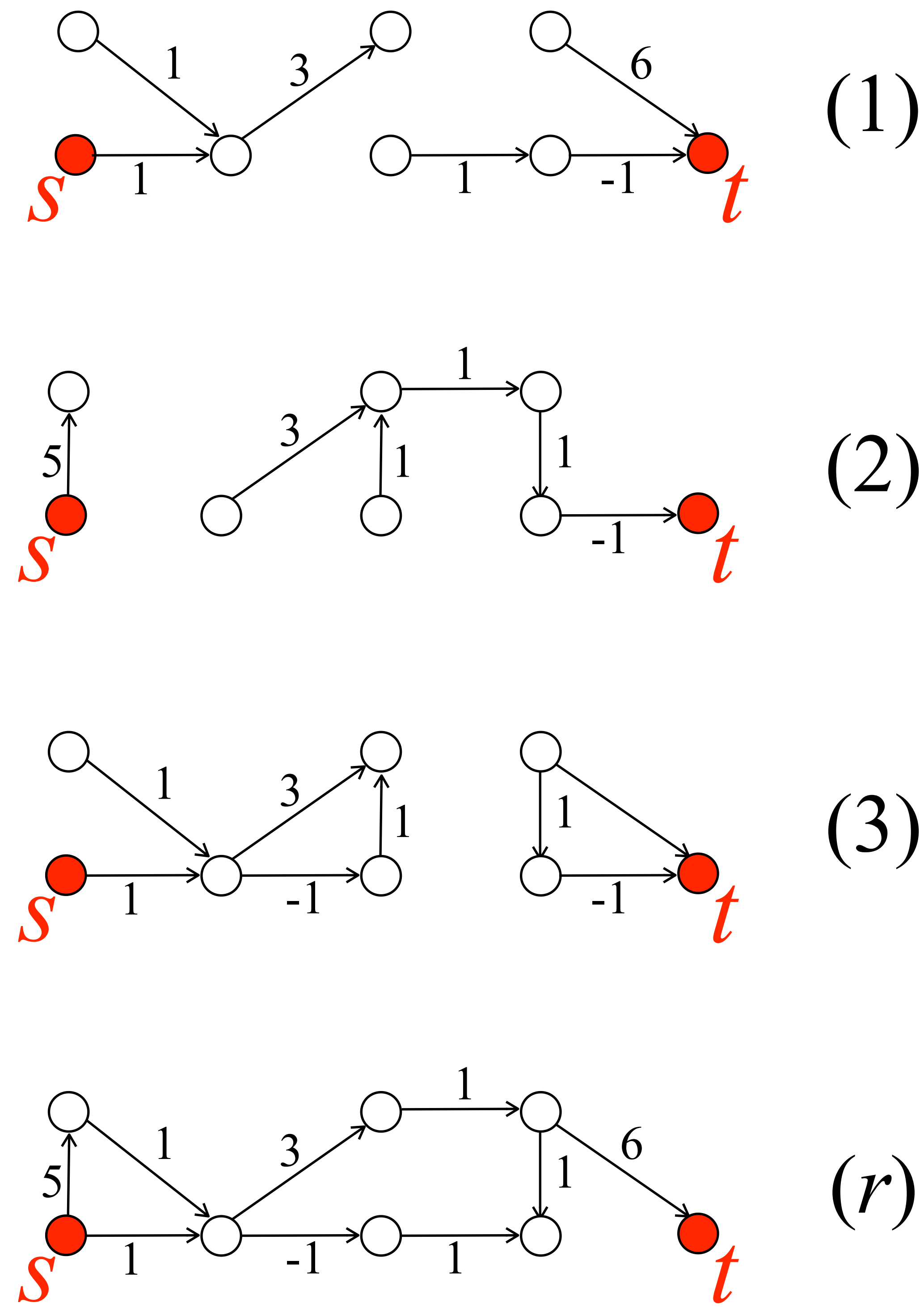


(*r*)

A Replacement Paths Algorithm

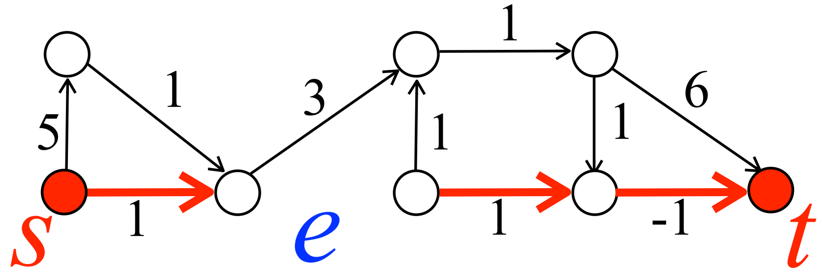
- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

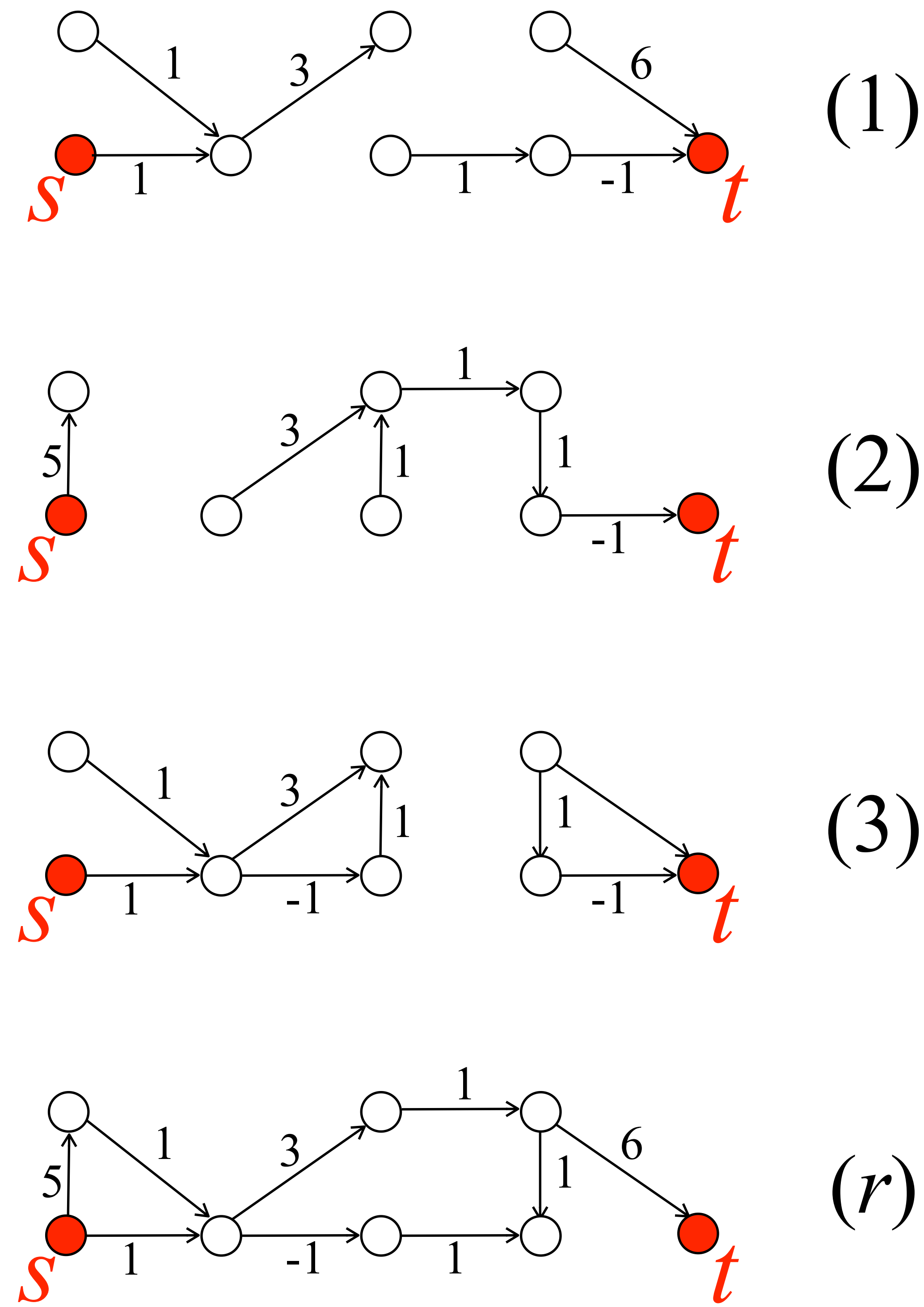




A Replacement Paths Algorithm

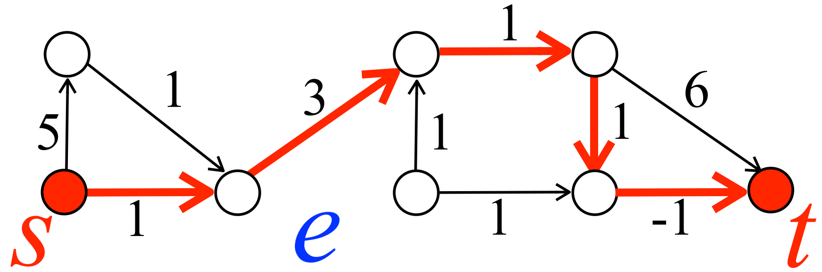
- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

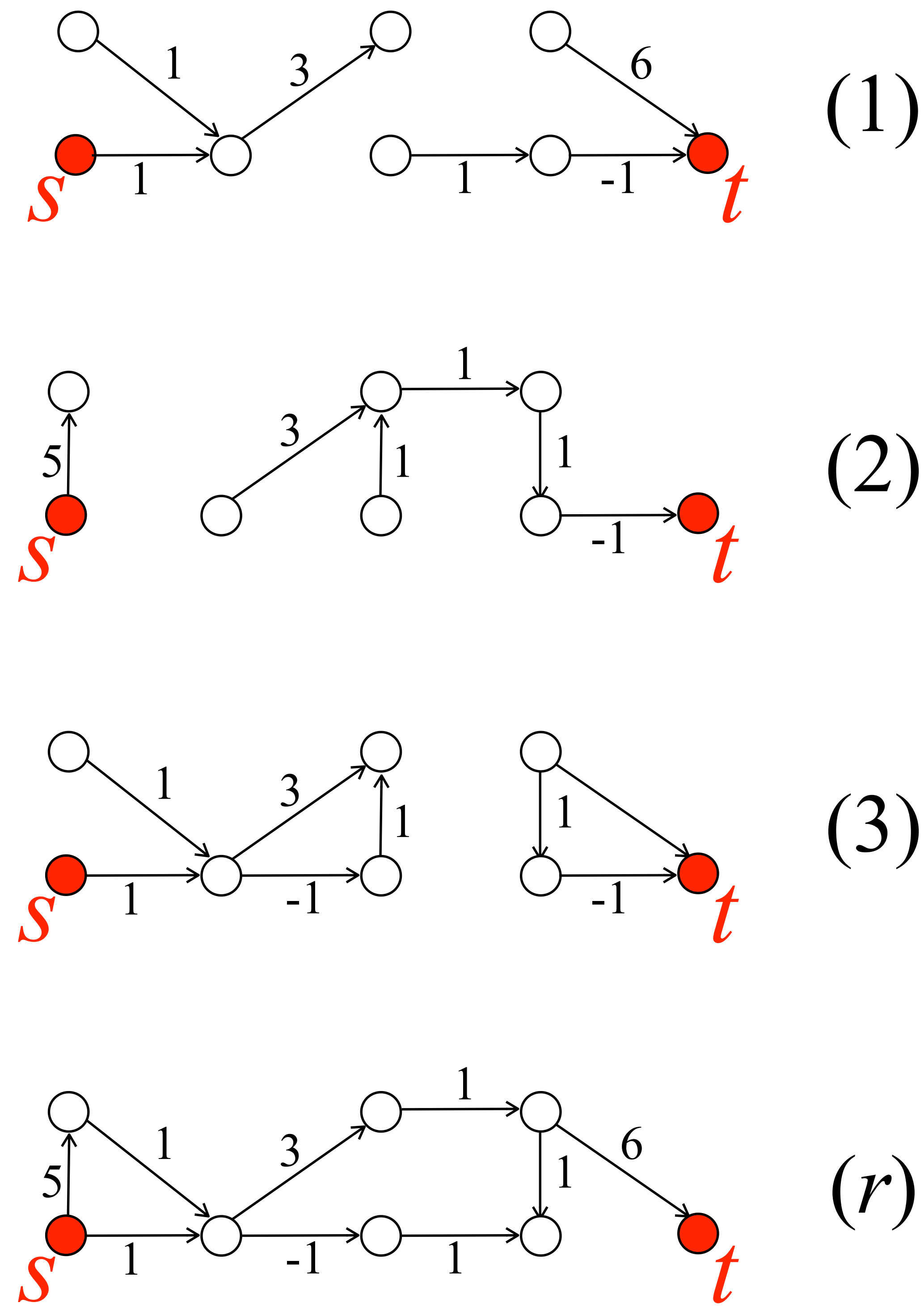




A Replacement Paths Algorithm

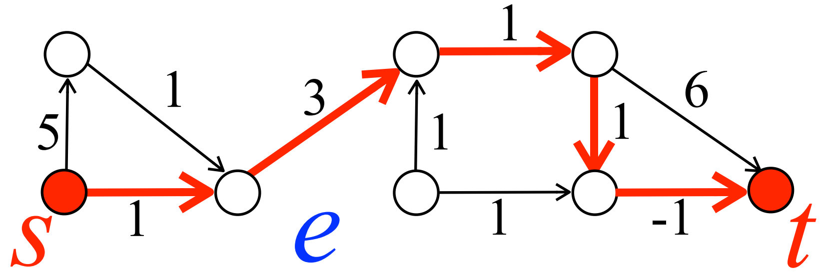
- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

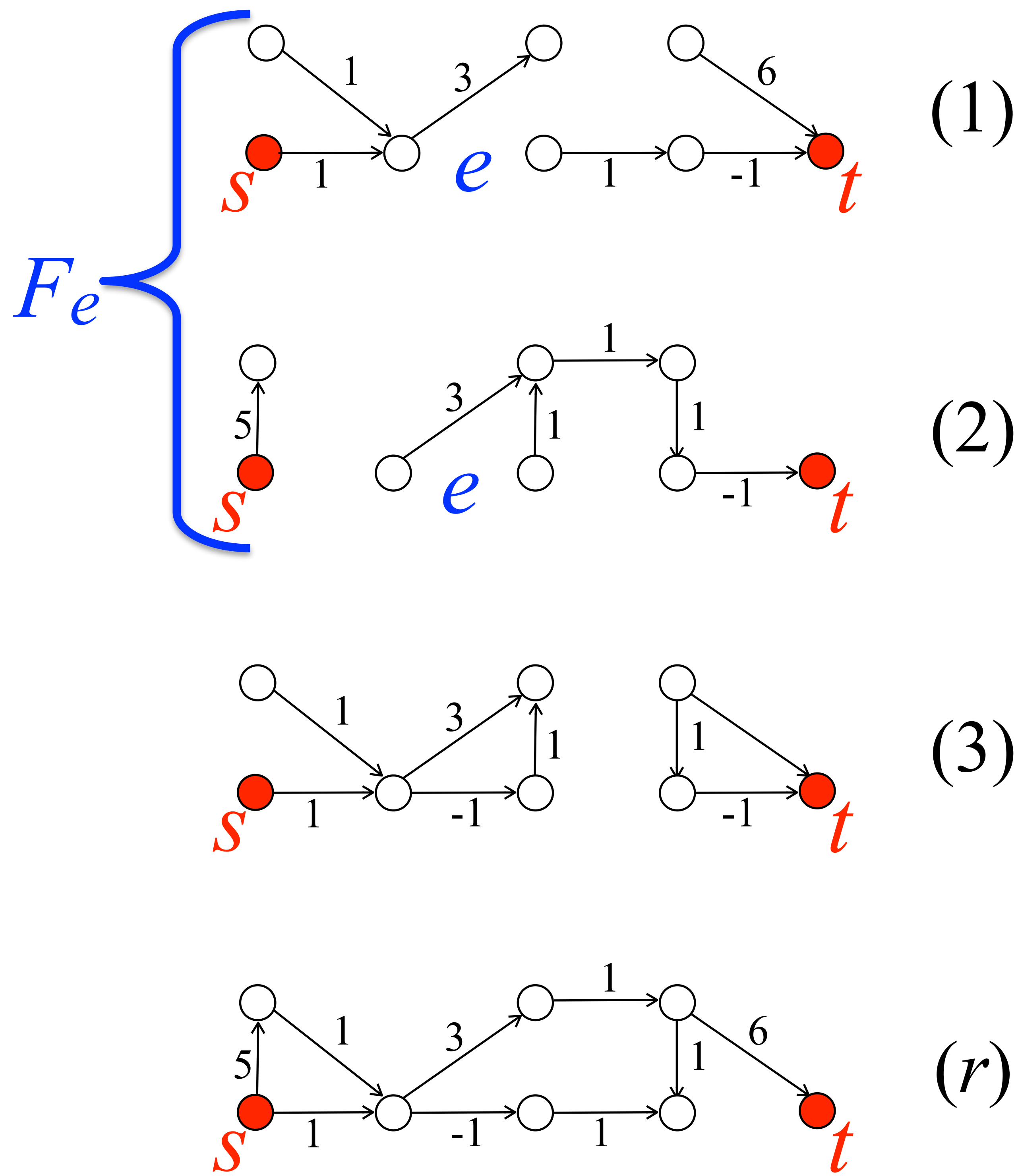




A Replacement Paths Algorithm

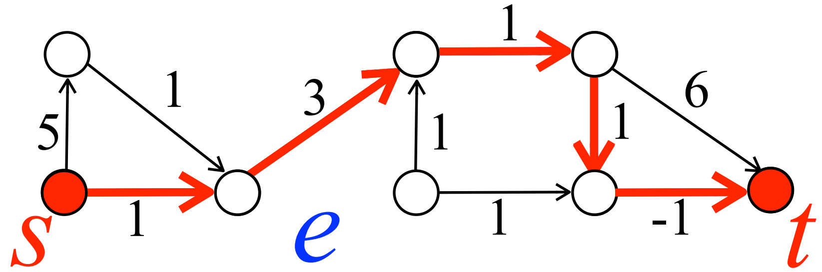
- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$



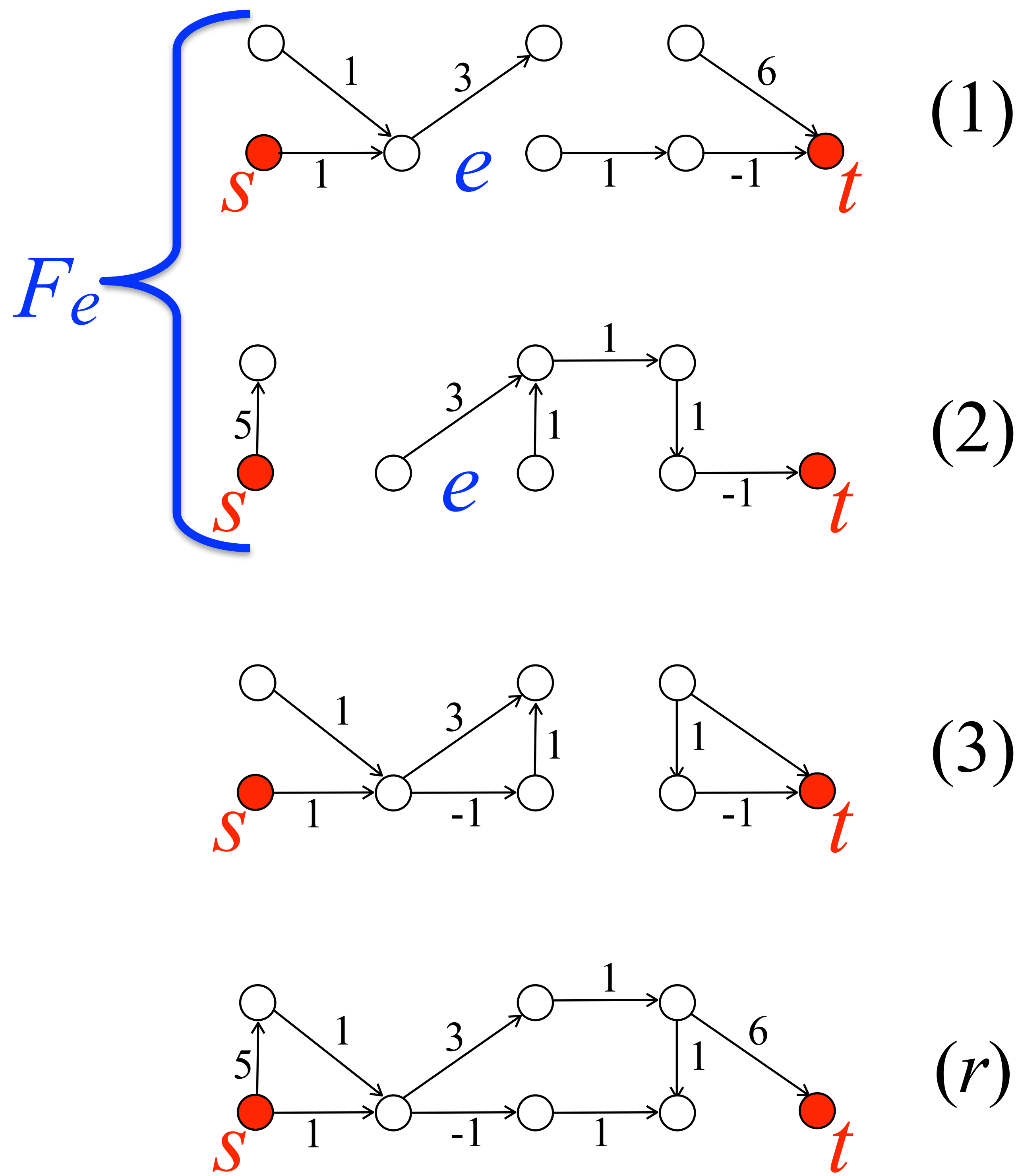


A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

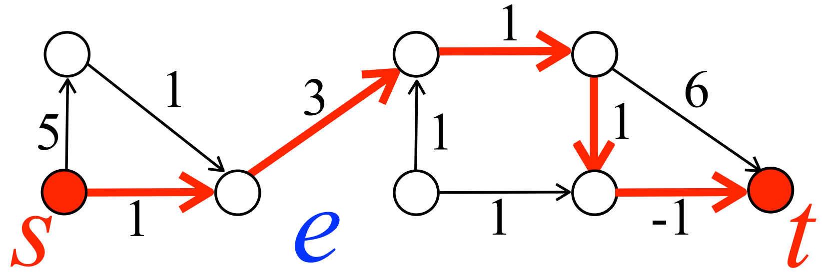


- With high probability
  - short subpaths survive

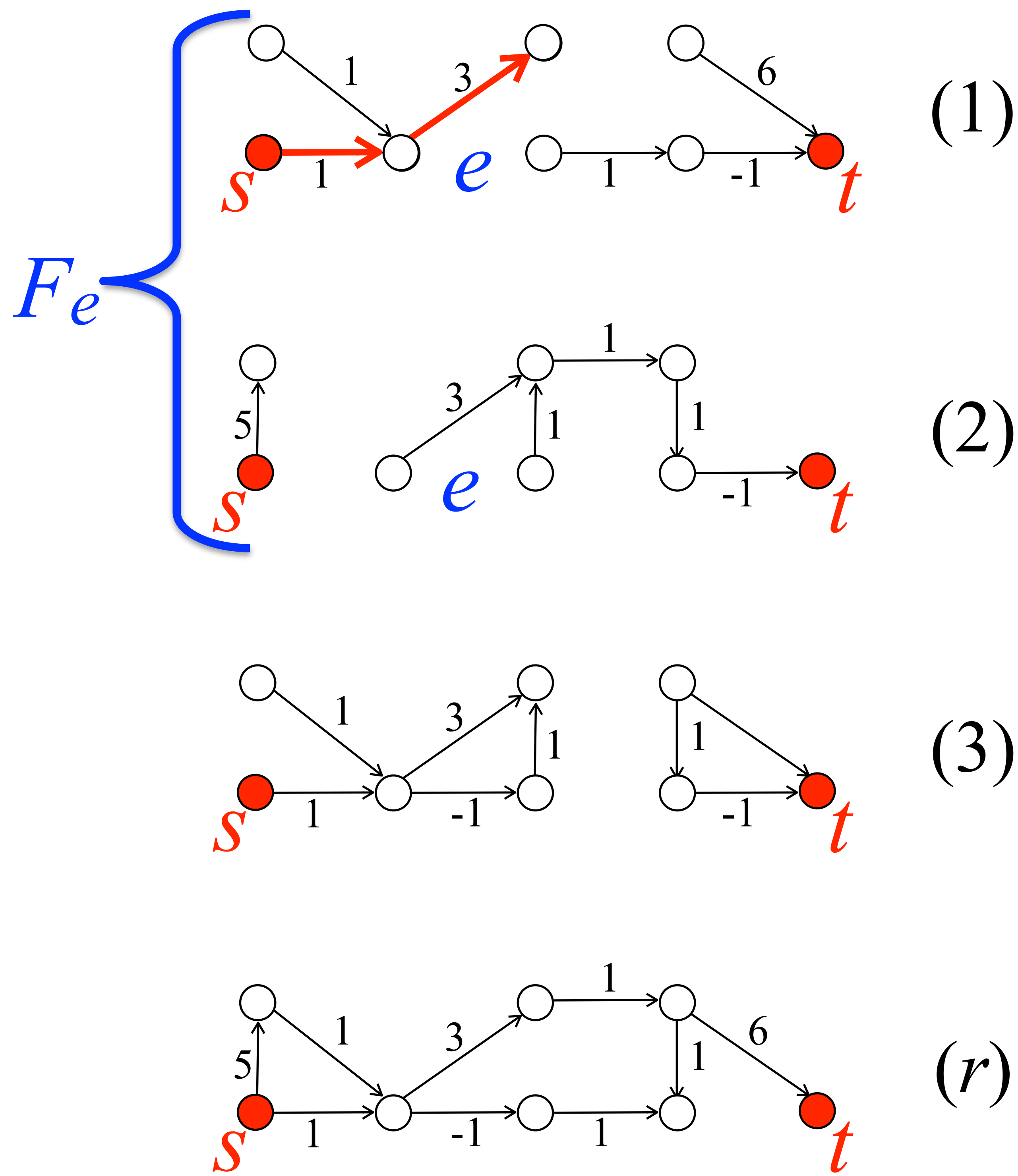


A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

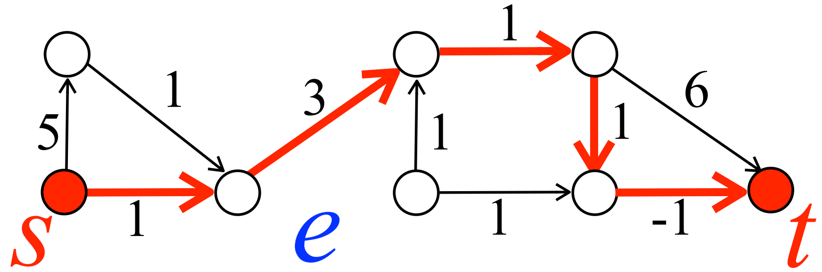


- With high probability
  - short subpaths survive

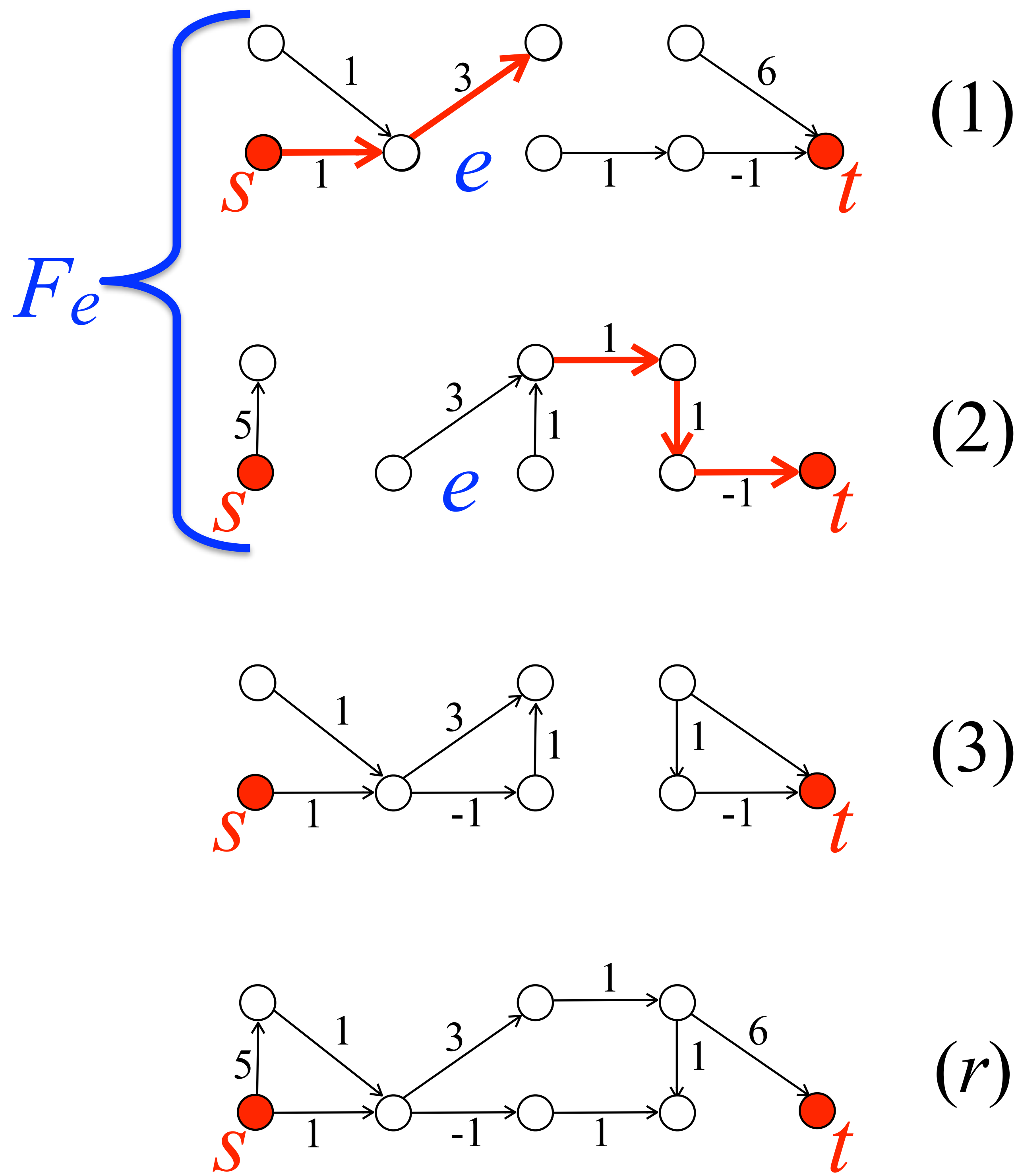


A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

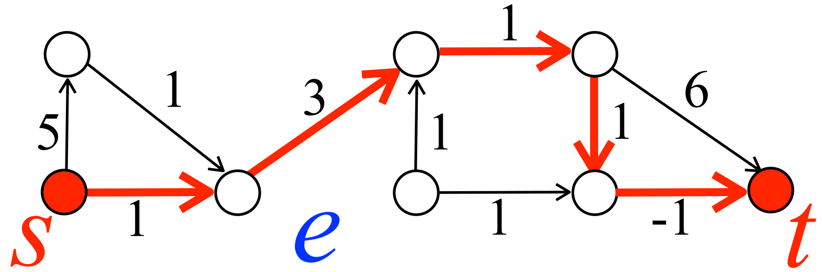


- With high probability
  - short subpaths survive

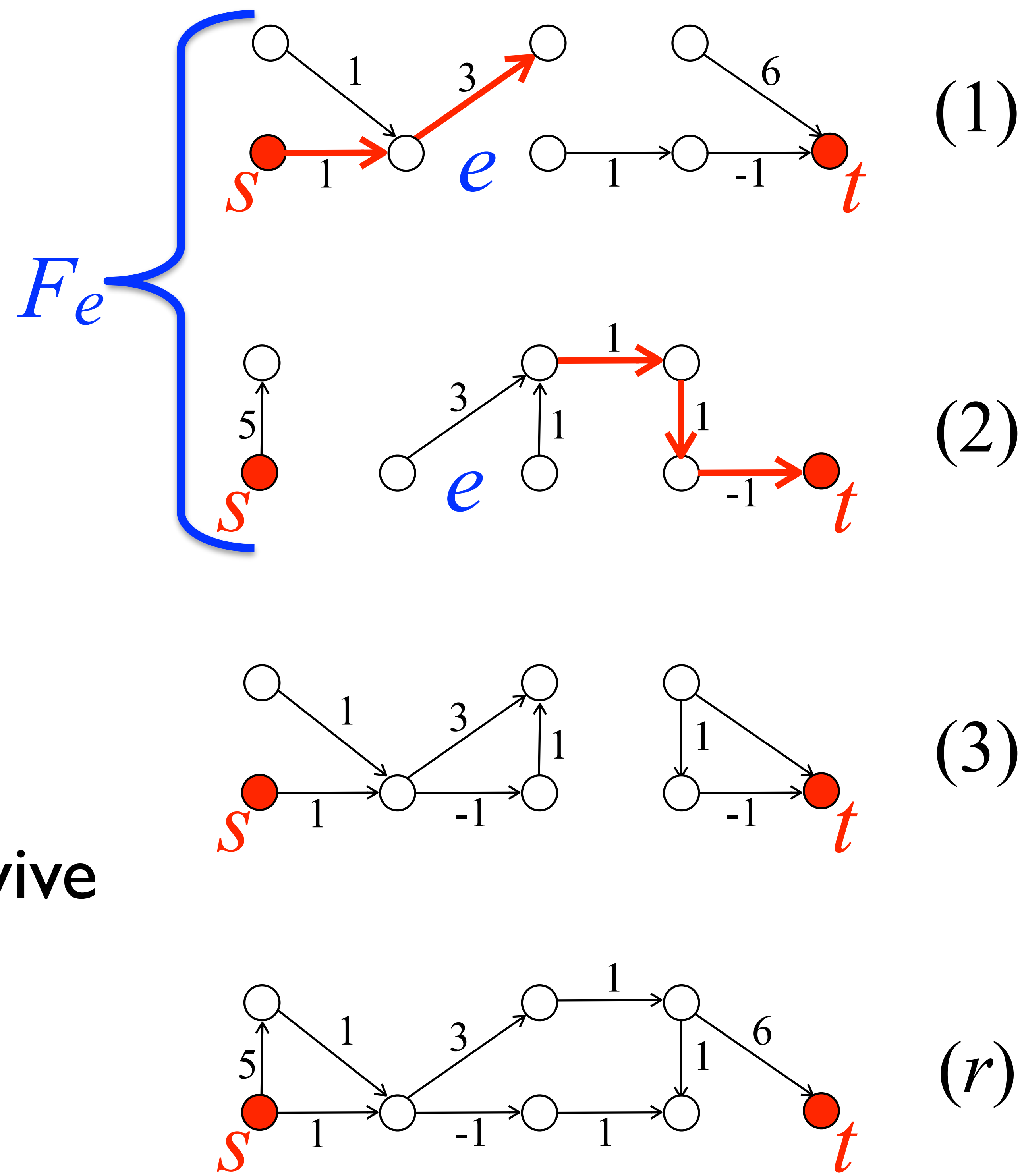


A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

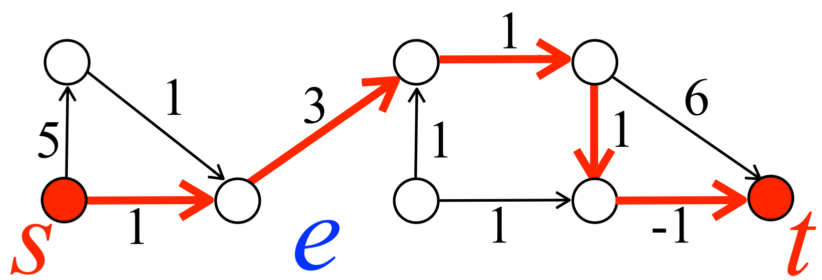


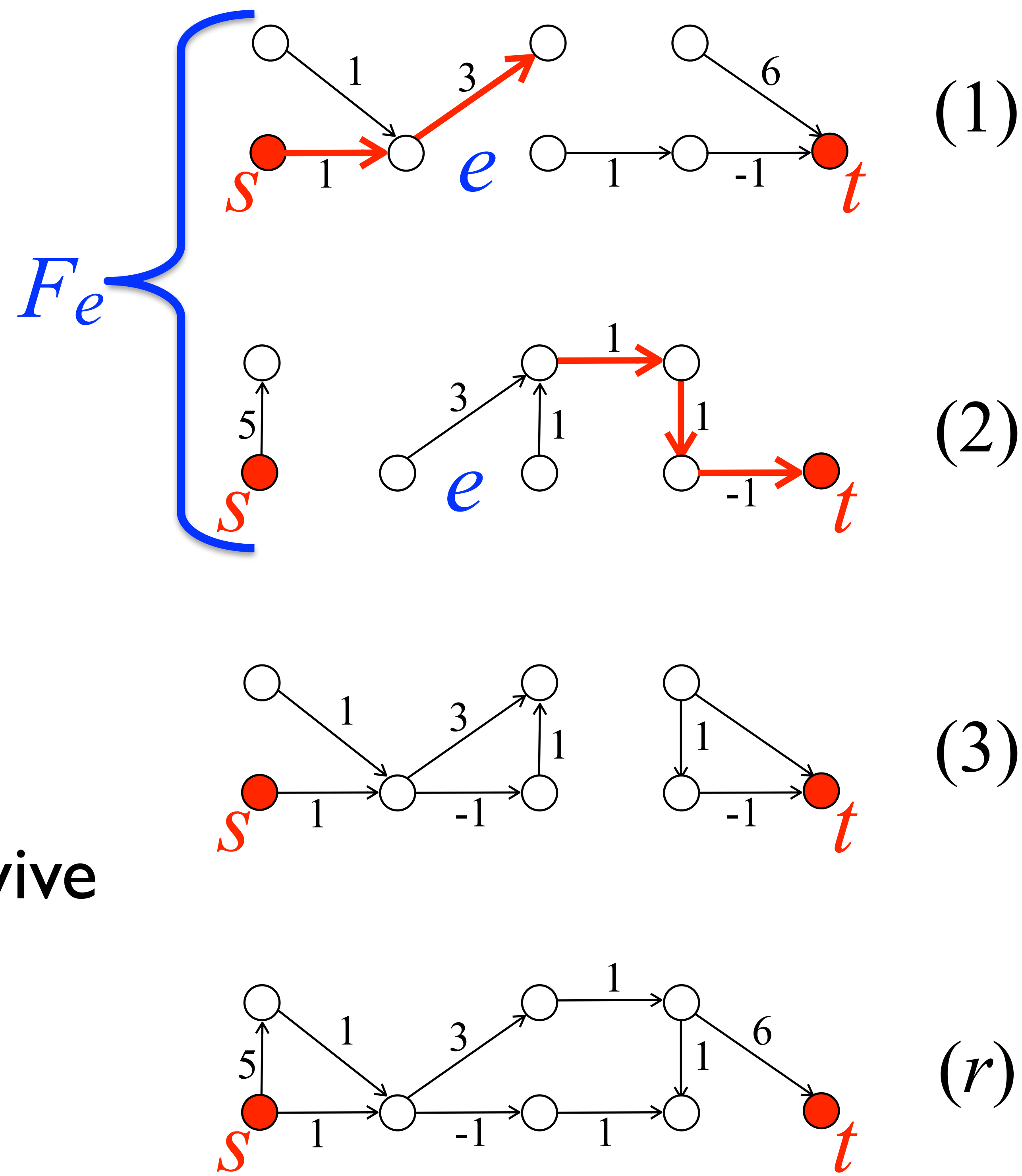
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive



A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

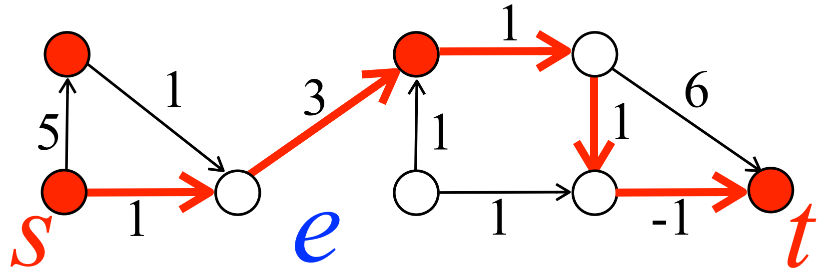




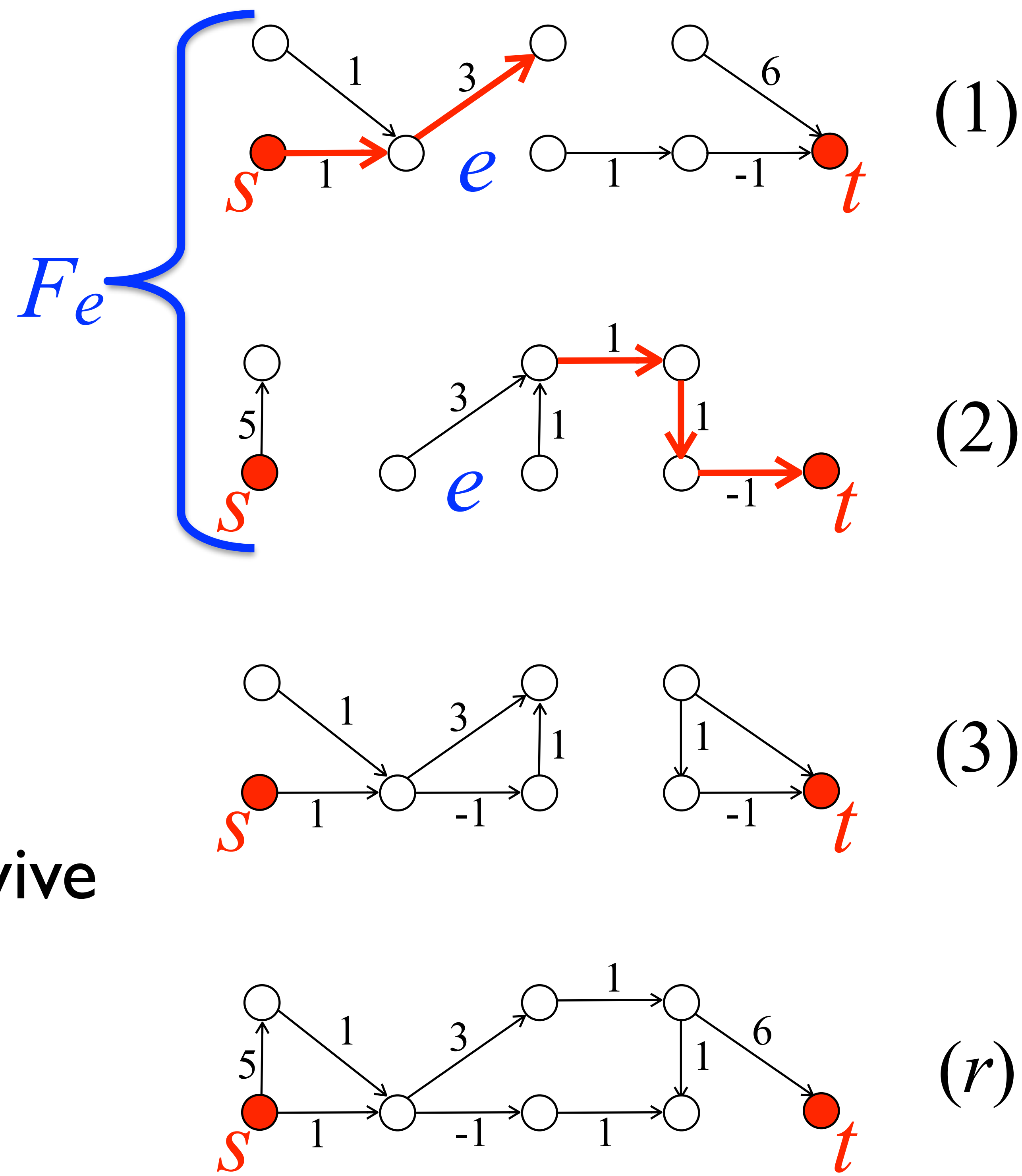
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$

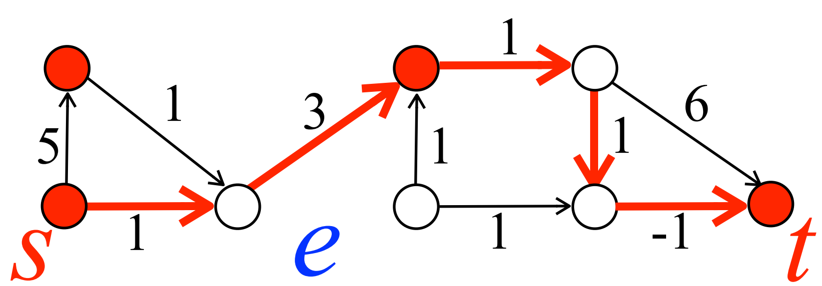


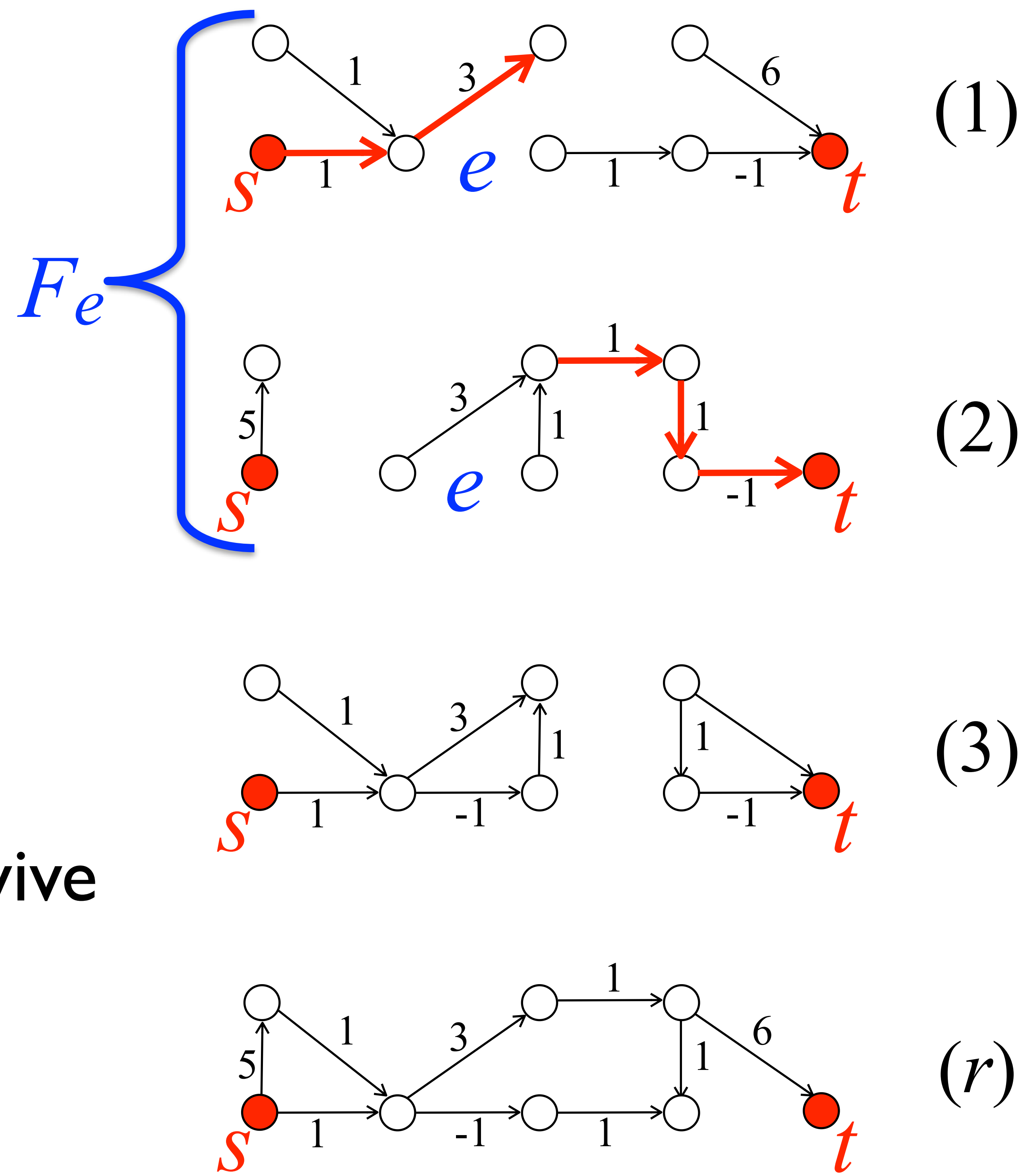
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$



A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$
- Random set $|B| = \tilde{O}(n^{\alpha})$

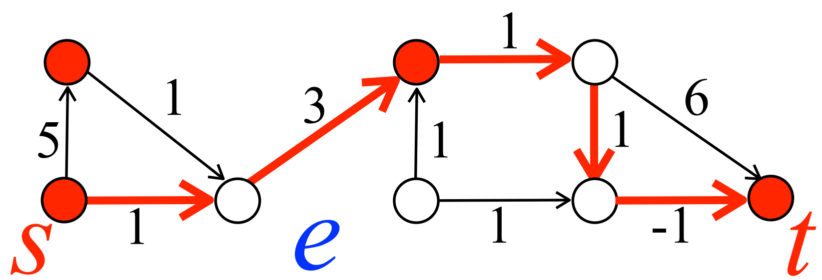


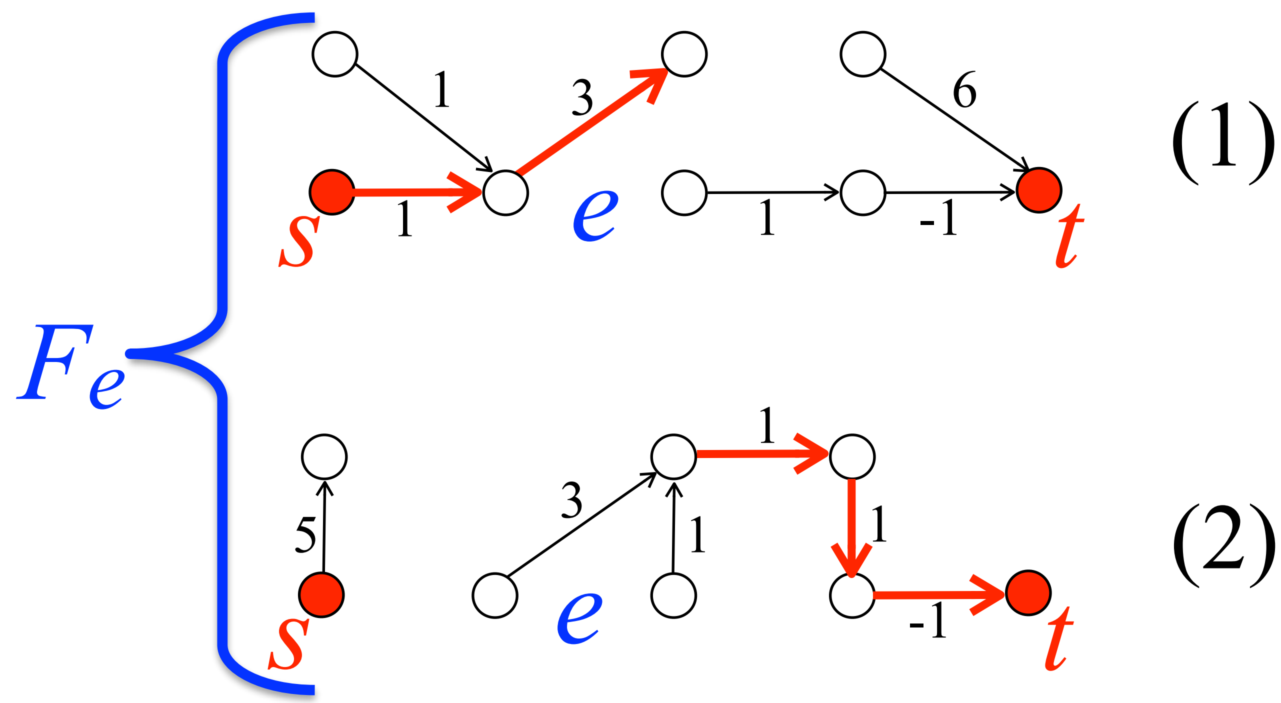


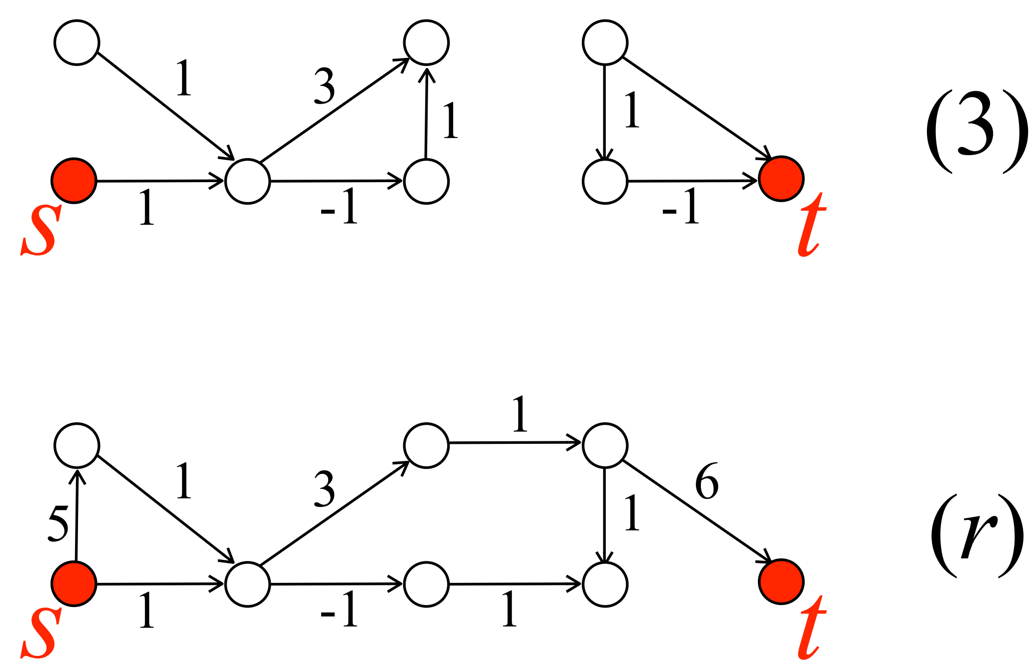
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$
- Random set $|B| = \tilde{O}(n^{\alpha})$



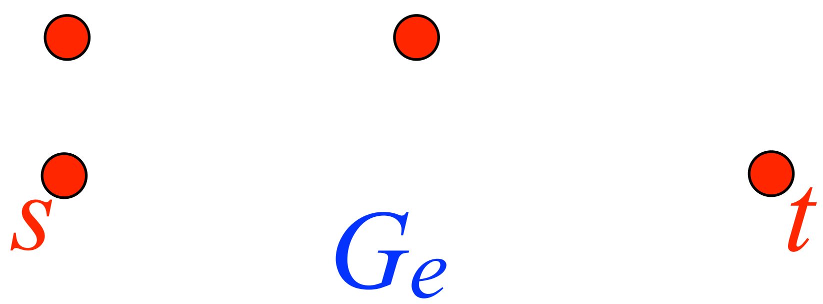


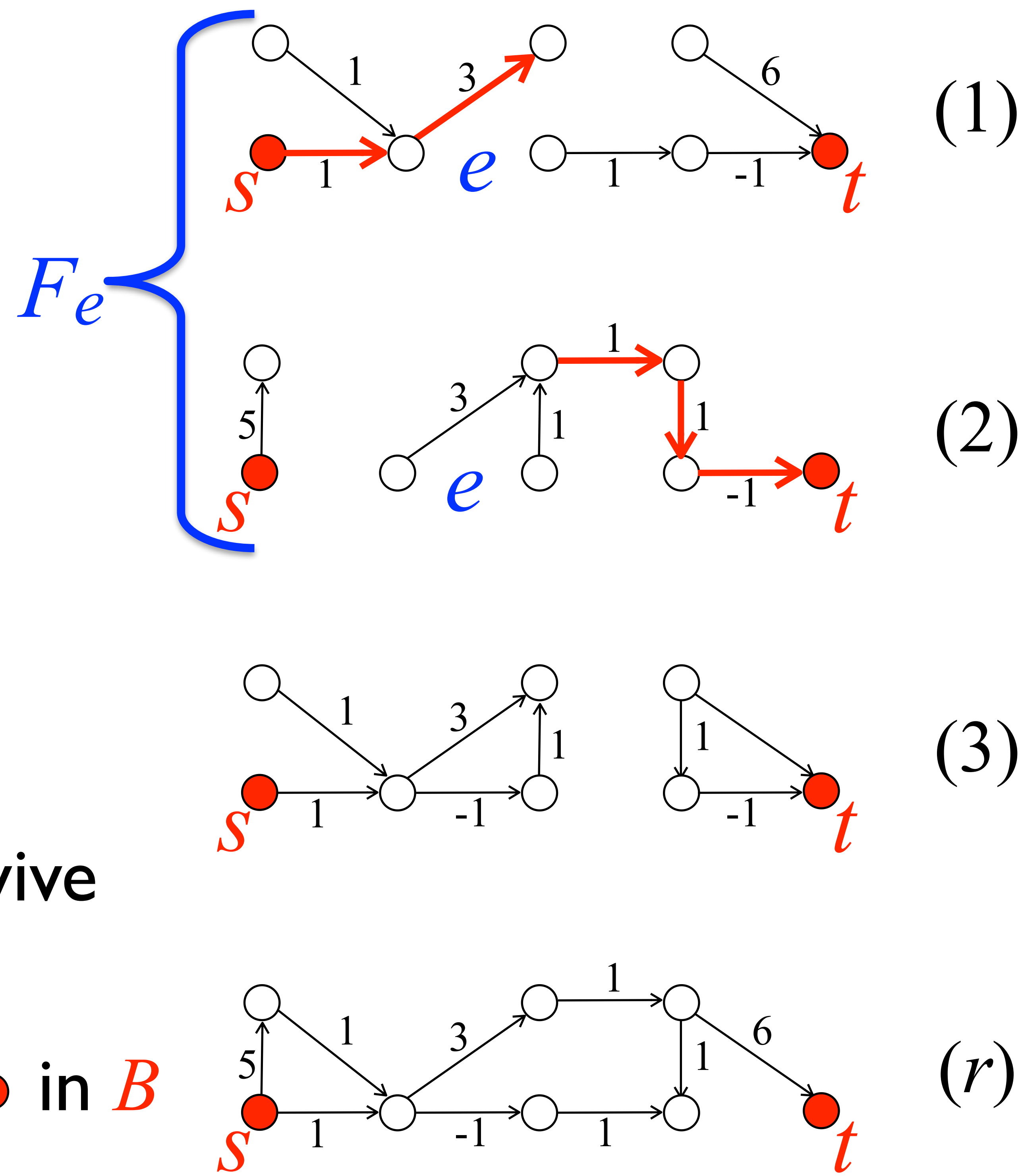


- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$
  - every $(n^{1-\alpha})$-subpath has ● in B

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$
- Random set $|B| = \tilde{O}(n^{\alpha})$

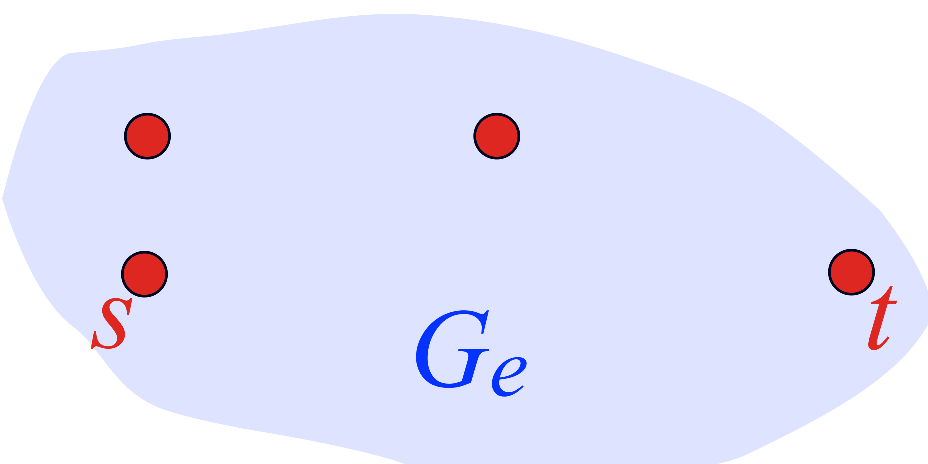




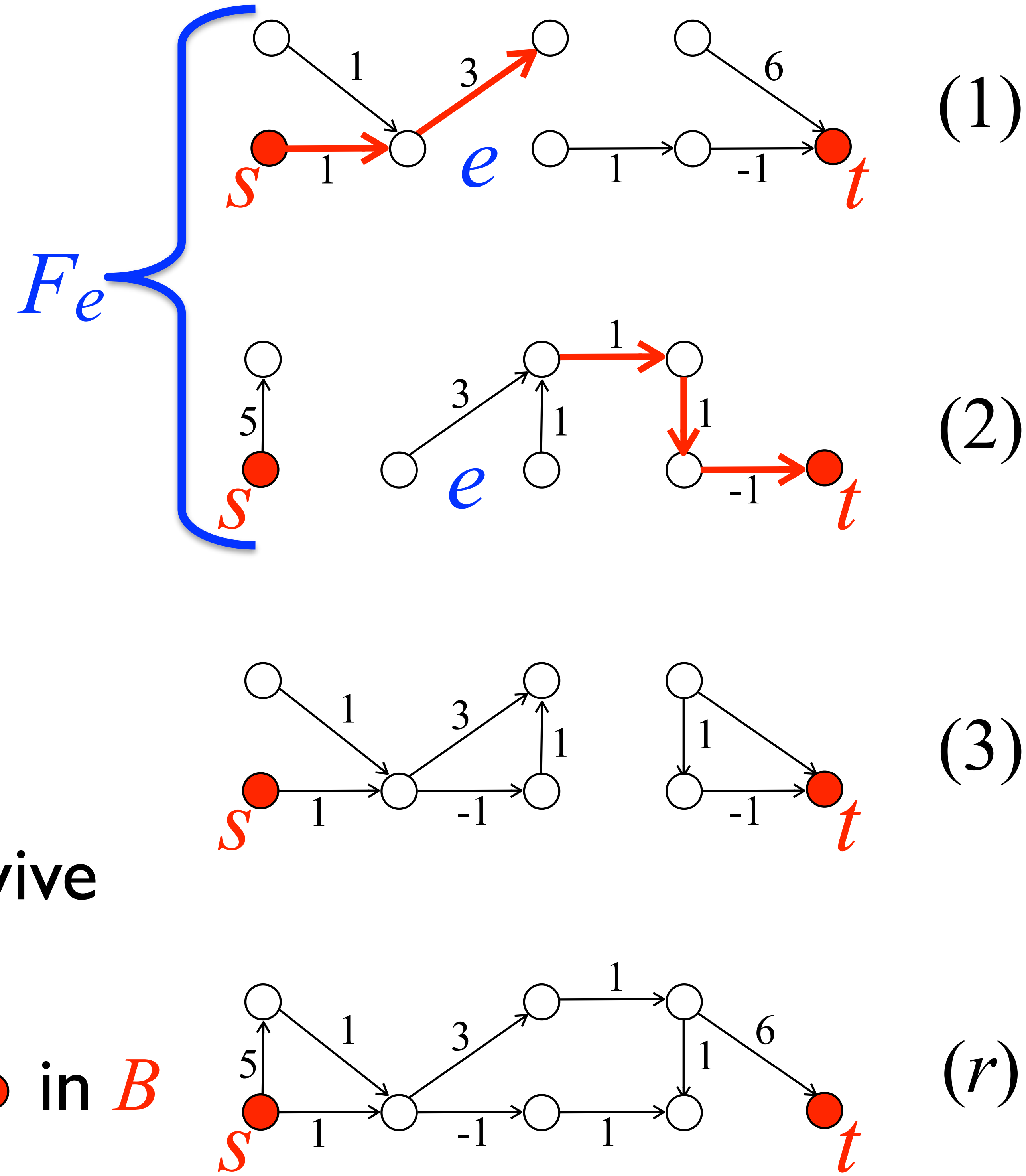
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$
  - every $(n^{1-\alpha})$-subpath has ● in B

A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$
- Random set $|B| = \tilde{O}(n^{\alpha})$

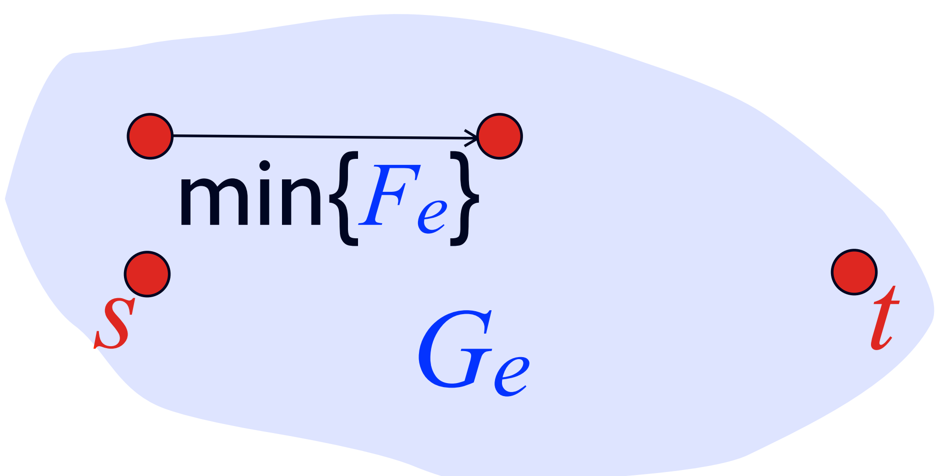


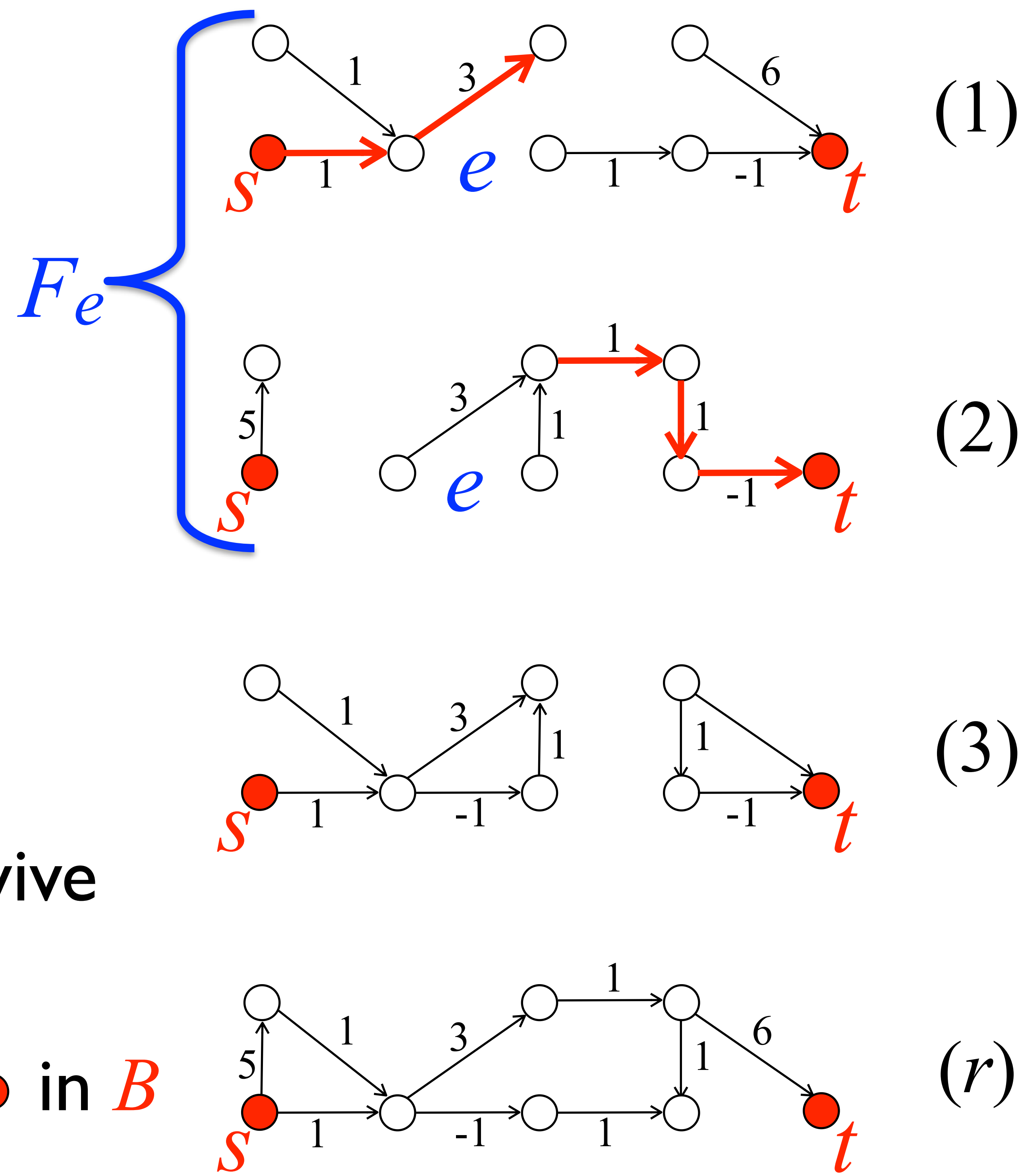
- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$
  - every $(n^{1-\alpha})$-subpath has ● in B



A Replacement Paths Algorithm

- Success probability 1-1/n
- $r = \tilde{O}(n^{1-\alpha})$ for $0 < \alpha < 1$
- Random set $|B| = \tilde{O}(n^{\alpha})$

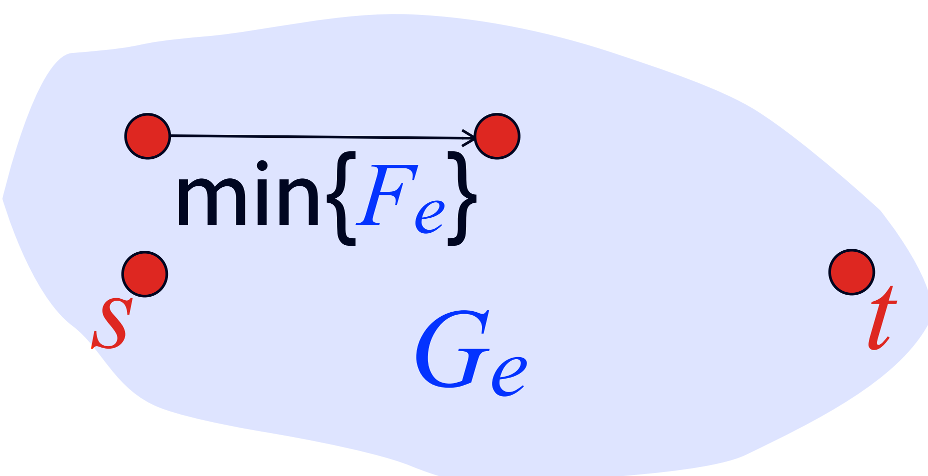


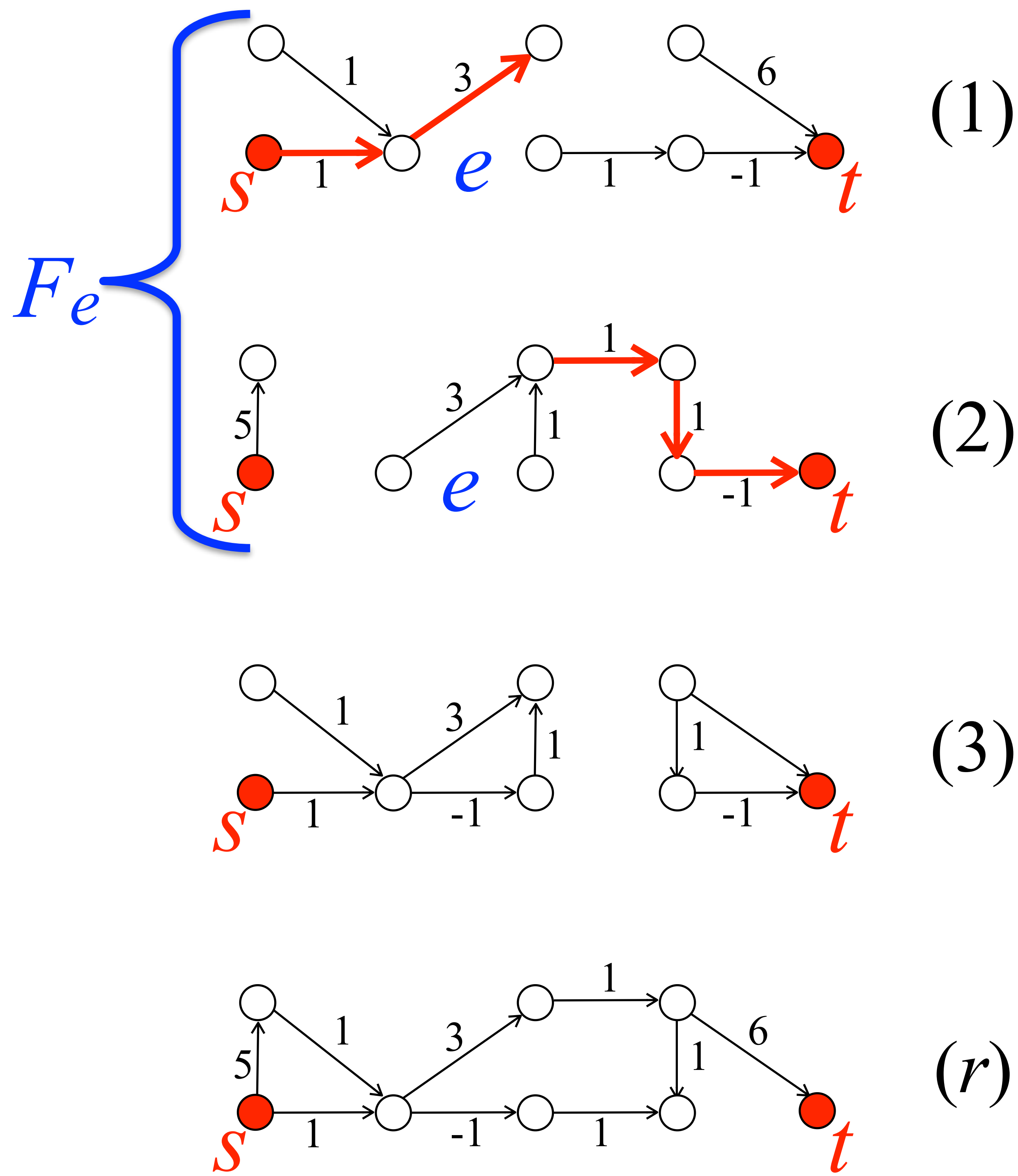


- With high probability
  - short $(n^{1-\alpha})$-subpaths survive
  - $|F_e| = \Theta(\log n)$
  - every $(n^{1-\alpha})$-subpath has ● in B

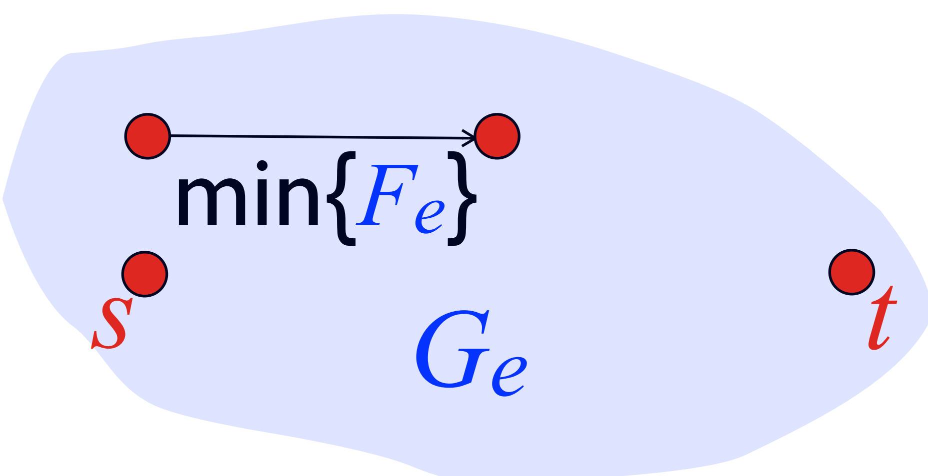
Challenges:

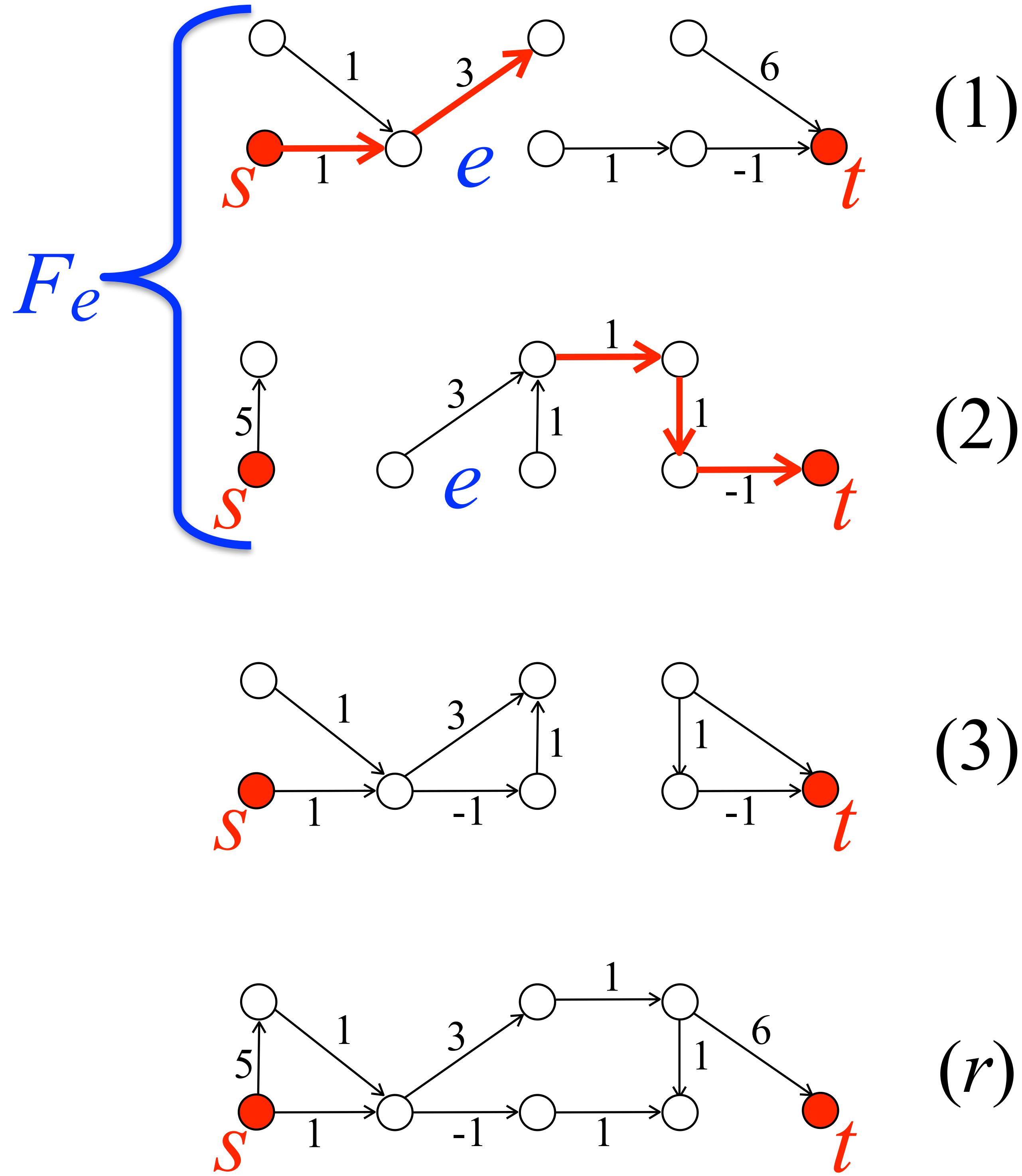
- Construct G_e
- Compute st-paths in all G_e





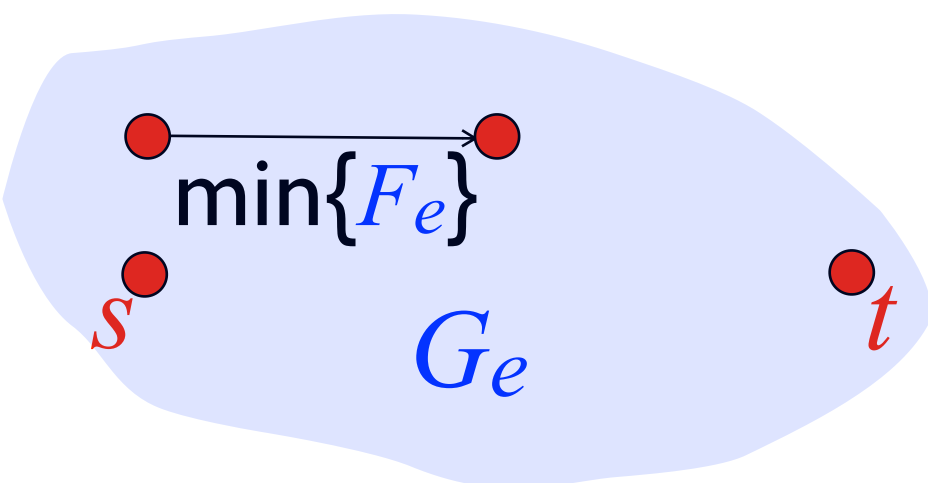
Constructing G_e

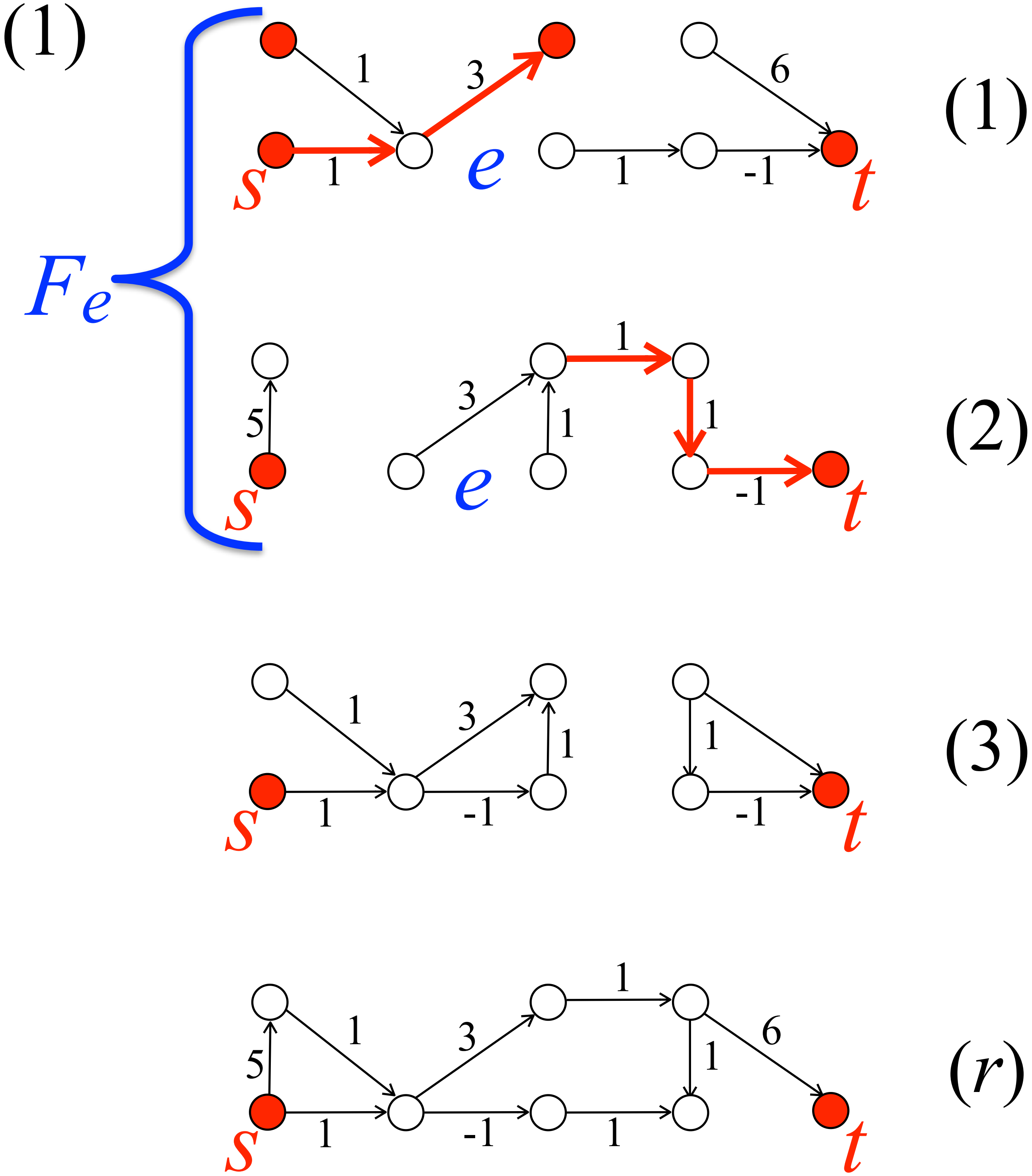




Constructing G_e

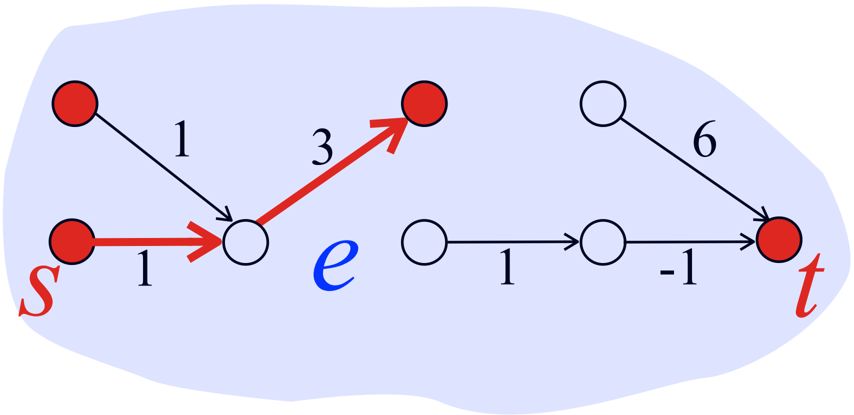
- Focus on ●-to-● distances in (1)





Constructing G_e

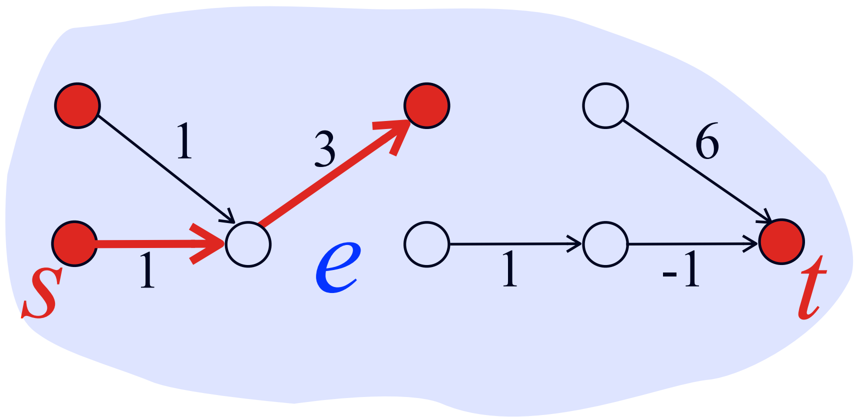
- Focus on ●-to-● distances in (1)



(1)

Constructing G_e

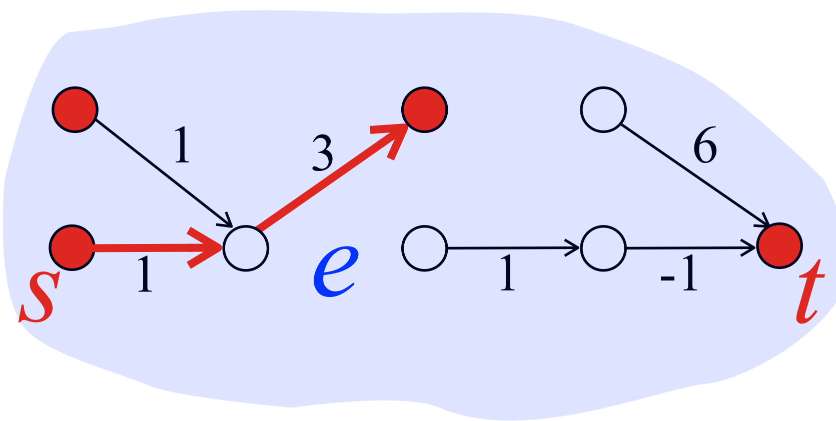
- Focus on ●-to-● distances in (1)
- Lengths in [$-M,\ldots,M$]



(1)

Constructing G_e

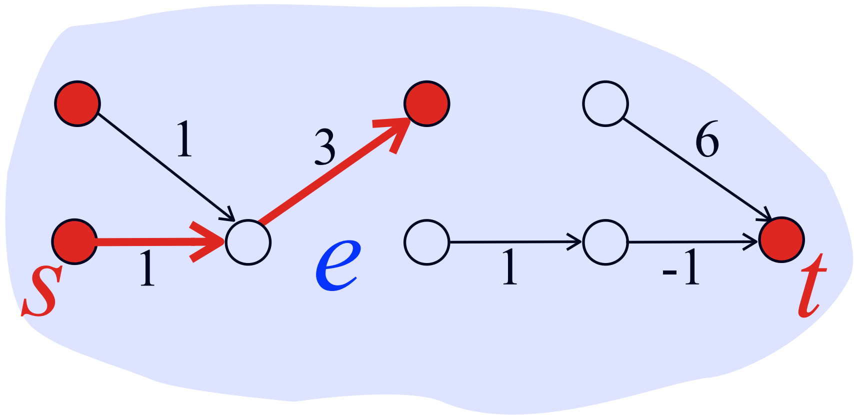
- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 



(1)

Constructing G_e

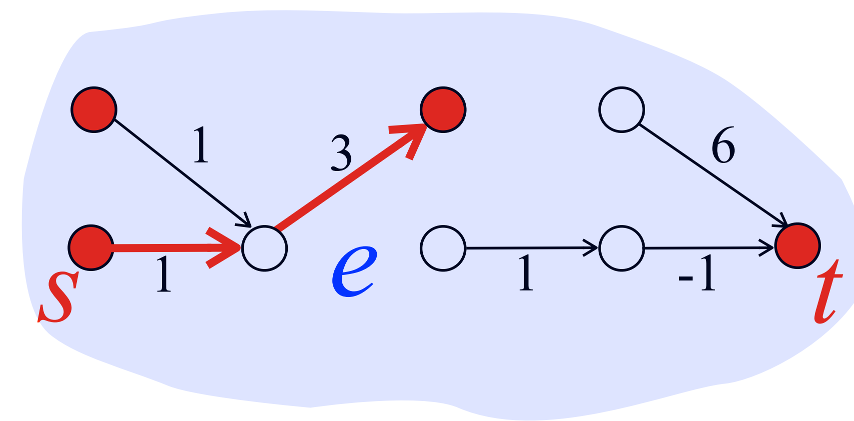
- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B×B subset 



(1)

Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\dots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 

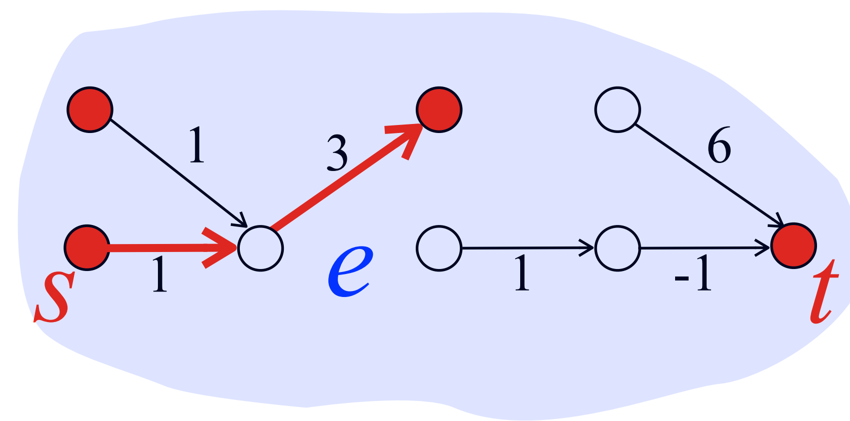


(1)

- $\tilde{O}(Mn^{\omega}$) [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k \{D[u,k]+D[k,v]\}$

Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths
  - Only need B x B subset

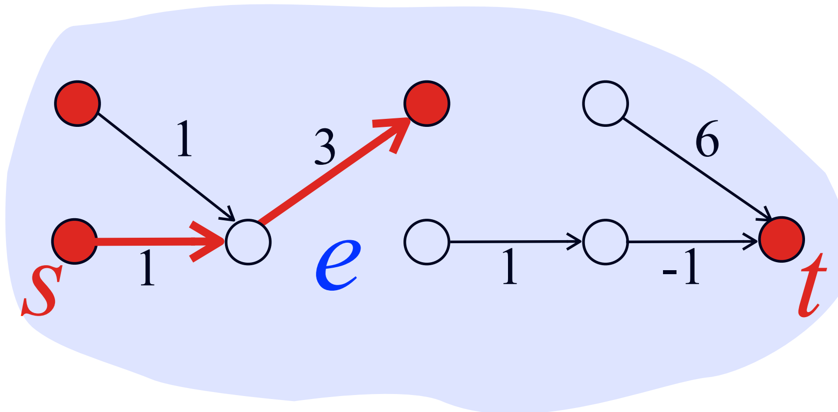


(1)

- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$

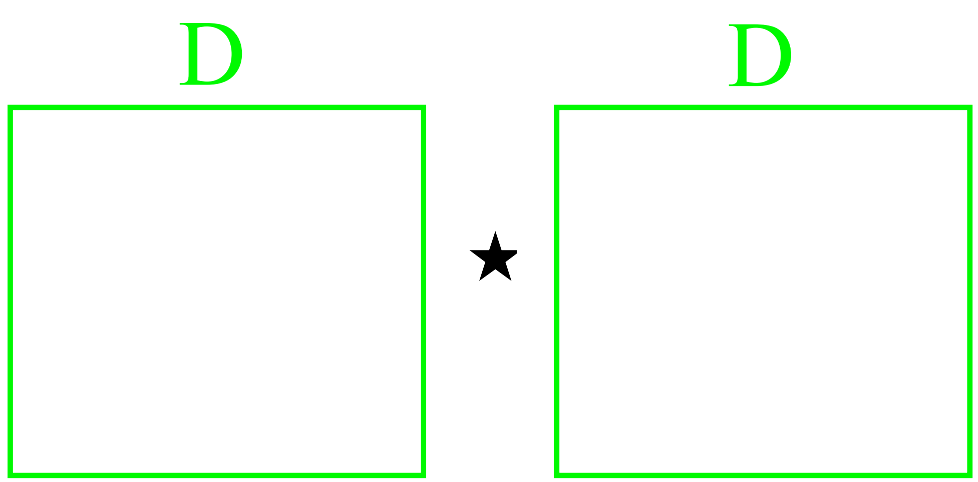
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 



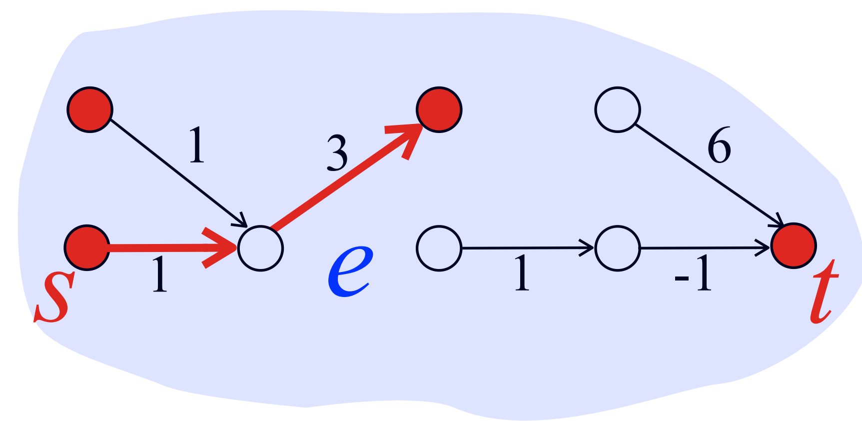
(1)

- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$



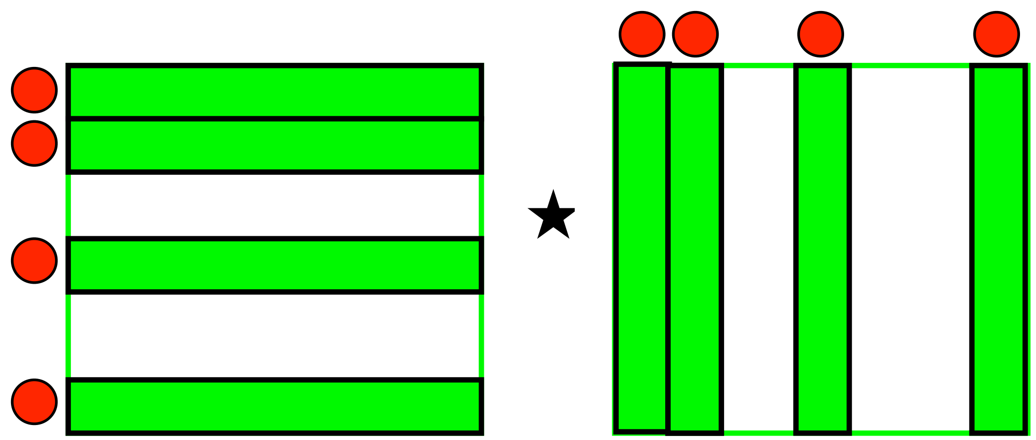
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 



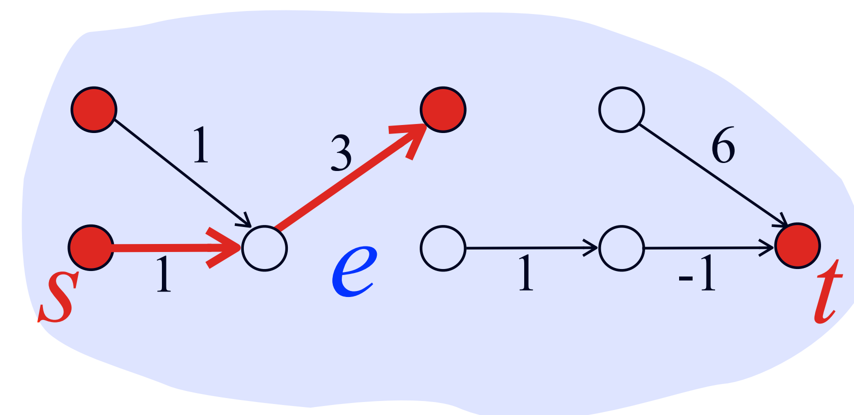
(1)

- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$



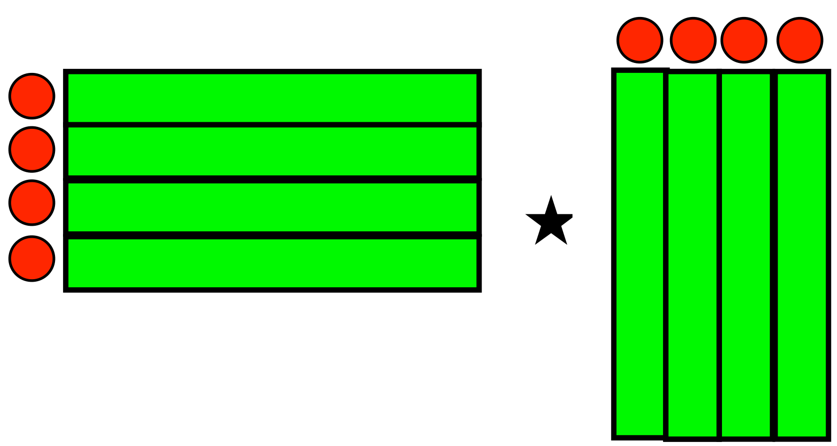
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\dots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 



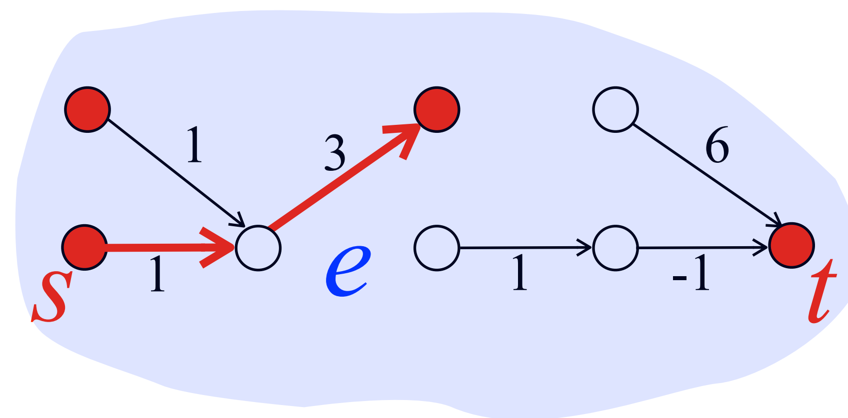
(1)

- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$



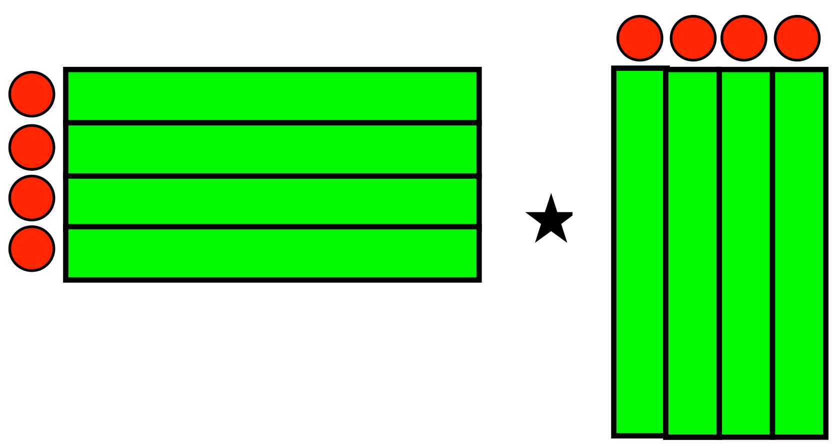
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 



(1)

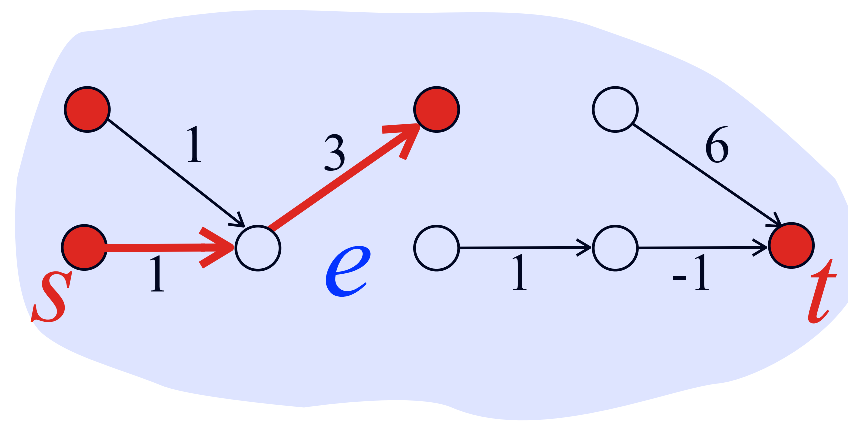
- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$



- Rectangular distance product [Alon, Galil, Margalit 1997]

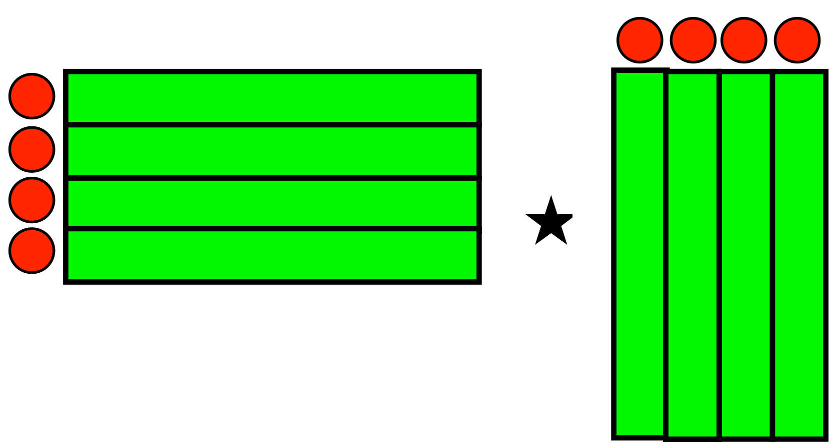
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths
  - Only need B x B subset



(1)

- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$

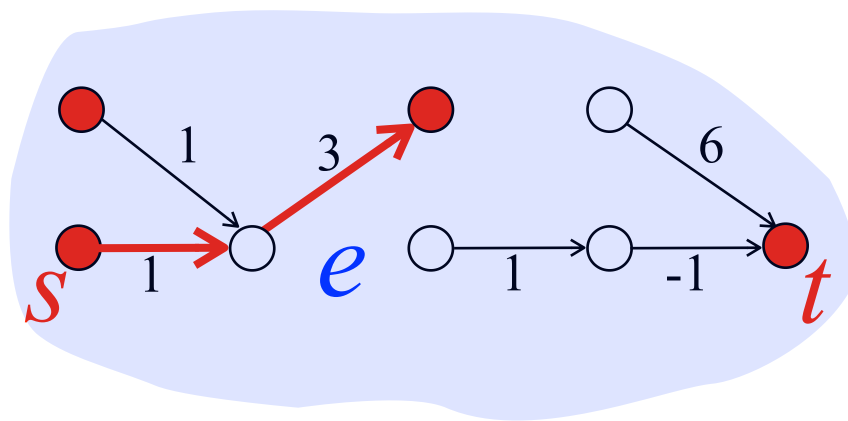


D elements can be as large as nM

- Rectangular distance product [Alon, Galil, Margalit 1997]

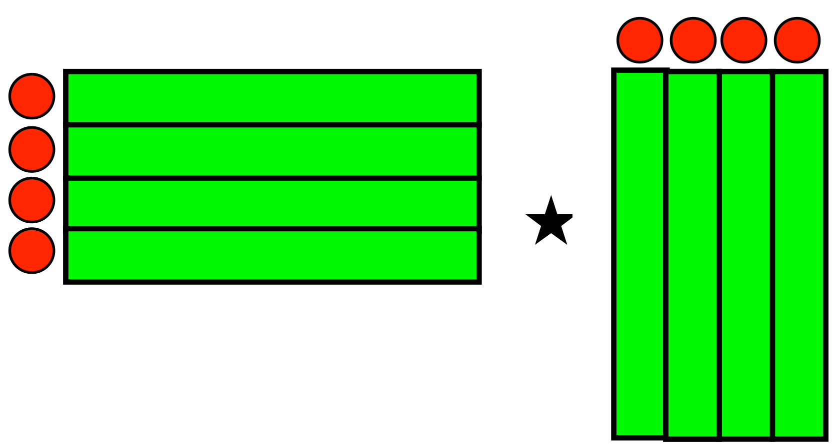
Constructing G_e

- Focus on ●-to-● distances in (1)
- Lengths in $[-M,\ldots,M]$
- $O(n^3)$ All-pairs-shortest-paths 
  - Only need B x B subset 



(1)

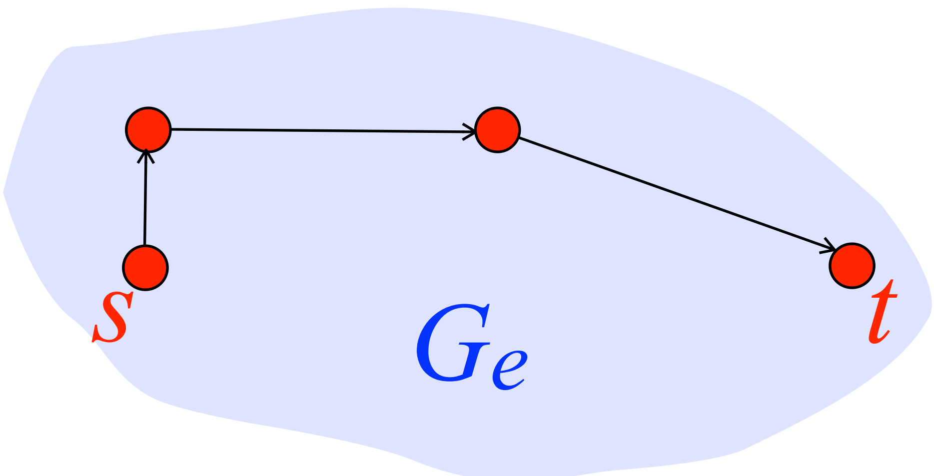
- $\tilde{O}(Mn^{\omega} + |B|^2 n)$ [Yuster-Zwick 2005]:
  Compute a matrix D s.t $dist(u,v) = \min_k\{D[u,k]+D[k,v]\}$



D elements can be as large as nM

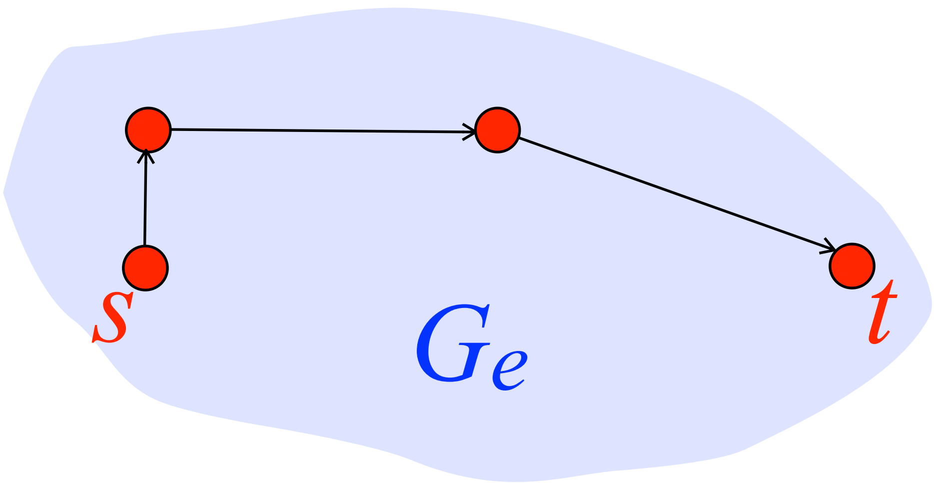
- Rectangular distance product [Alon, Galil, Margalit 1997]
  - Safe to ignore D elements larger than $Mn^{1-\alpha}$

Computing st-paths in all G_e



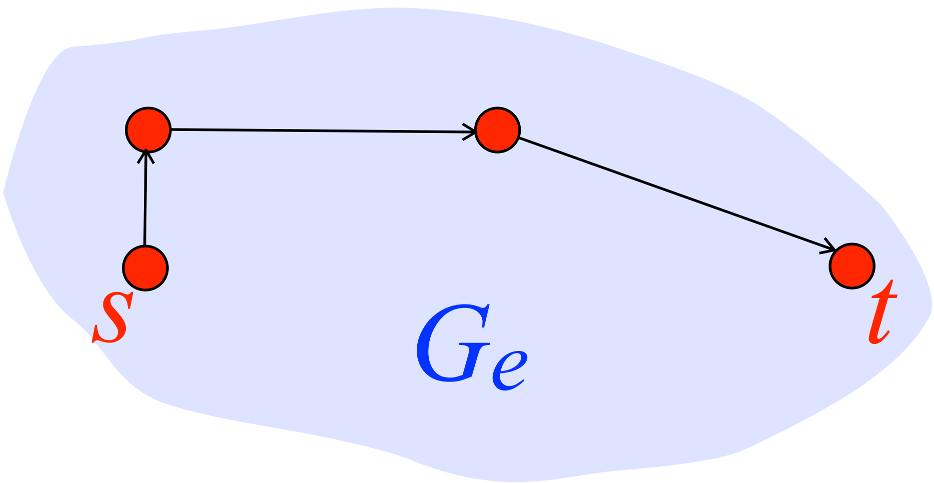
- Unbounded weights.

Computing st-paths in all G_e



- Unbounded weights.
- n executions of Goldberg or Bellman-Ford 

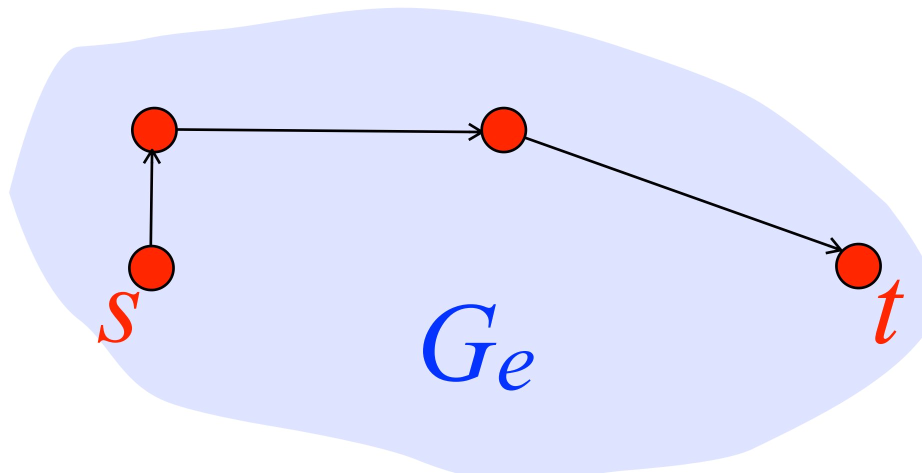
Computing st-paths in all G_e



- Unbounded weights.

- n executions of Goldberg or Bellman-Ford 
- 1 execution of Goldberg, n executions of Dijkstra 

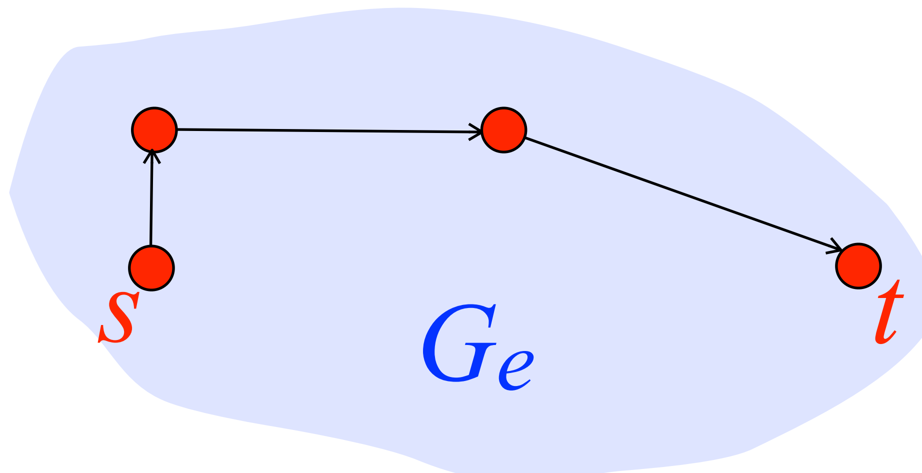
Computing st-paths in all G_e



- Unbounded weights.

- n executions of Goldberg or Bellman-Ford 
- 1 execution of Goldberg, n executions of Dijkstra 
  - Compute shortest paths $d(v)$ from s in G

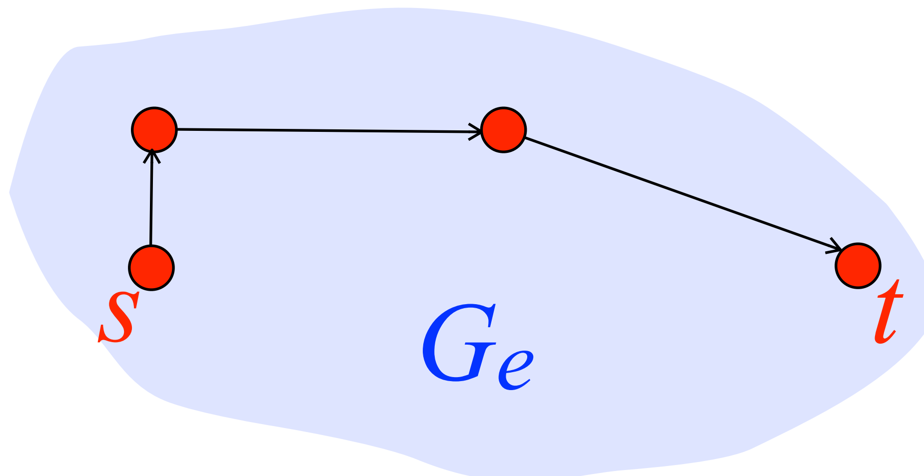
Computing st-paths in all G_e



- Unbounded weights.

- n executions of Goldberg or Bellman-Ford
- 1 execution of Goldberg, n executions of Dijkstra
  - Compute shortest paths $d(v)$ from s in G
  - In every G_e reweigh $w(u,v)$ to $w(u,v) + d(u) - d(v)$

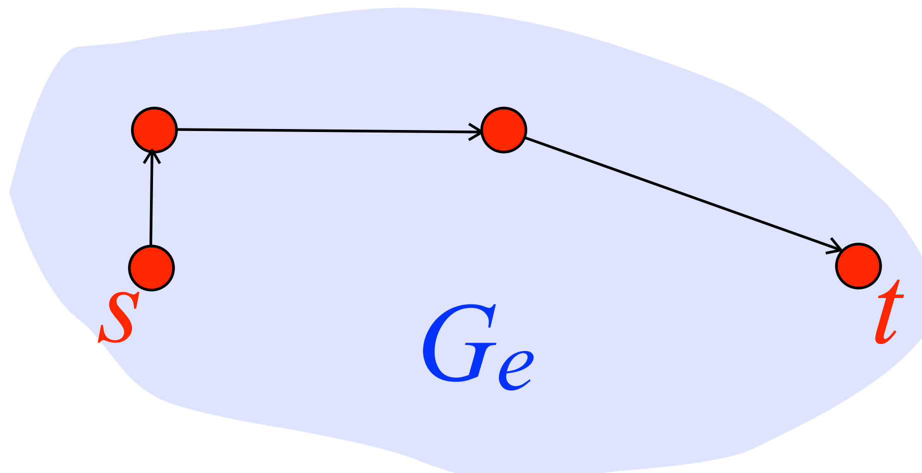
Computing st-paths in all G_e



- Unbounded weights.

- n executions of Goldberg or Bellman-Ford
- 1 execution of Goldberg, n executions of Dijkstra
  - Compute shortest paths $d(v)$ from s in G
  - In every G_e reweigh $w(u,v)$ to $w(u,v) + d(u) - d(v)$
    - Dijkstra on every (positive weighted) G_e

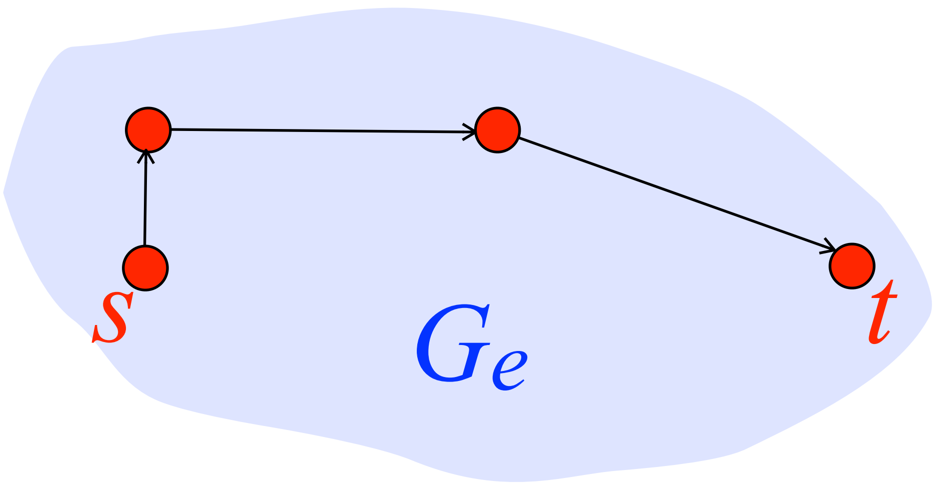
Computing st-paths in all G_e



- Unbounded weights.

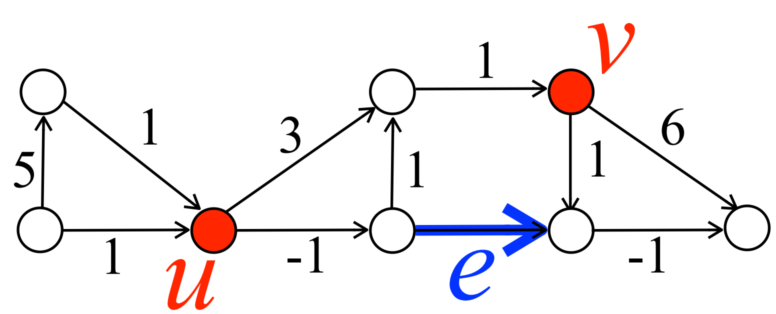
- n executions of Goldberg or Bellman-Ford 
- 1 execution of Goldberg, n executions of Dijkstra 
  - Compute shortest paths $d(v)$ from s in G
  - In every G_e reweigh $w(u,v)$ to $w(u,v) + d(u) - d(v)$
    - Dijkstra on every (positive weighted) G_e
    - $O(n|B|^2)$

Putting it all Together



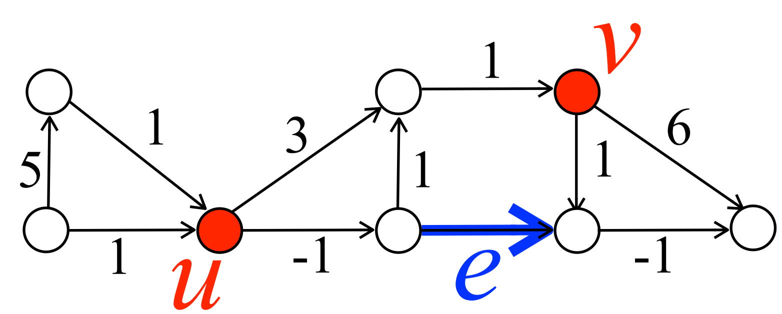
- Replacement paths in $\tilde{O}(Mn^{1+2\omega/3}) = O(Mn^{2.584})$ time.

All-pairs Replacement Paths



Query: (u, v, e)?

All-pairs Replacement Paths

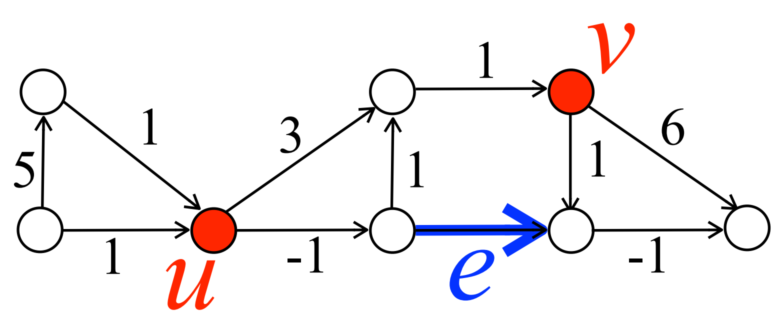


Query: (u,v,e)?

- $\tilde{O}(Mn^{1+\omega-\alpha})$ preprocessing: (not knowing u,v,e)

- $\tilde{O}(n^{1+\alpha})$ query time: (knowing u,v,e)

All-pairs Replacement Paths



Query: (u,v,e)?

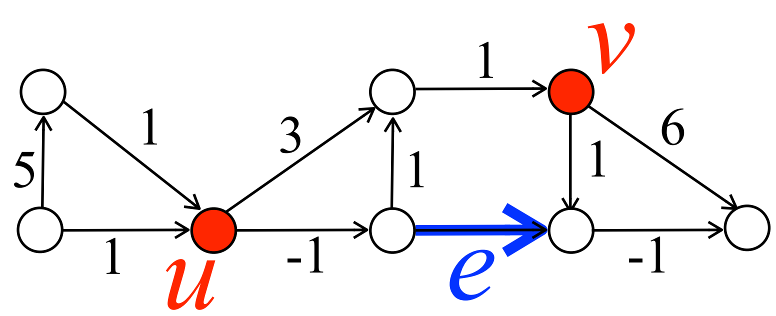
- $\tilde{O}(Mn^{1+\omega-\alpha})$ preprocessing: (not knowing u,v,e)

Construct the random graphs

Pick B, and compute B x B distances

- $\tilde{O}(n^{1+\alpha})$ query time: (knowing u,v,e)

All-pairs Replacement Paths



Query: (u,v,e)?

- $\tilde{O}(Mn^{1+\omega-\alpha})$ preprocessing: (not knowing u,v,e)

Construct the random graphs

Pick B, and compute B x B distances

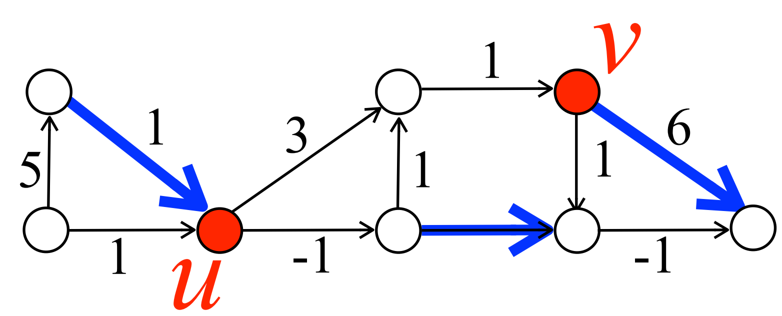
- $\tilde{O}(n^{1+\alpha})$ query time: (knowing u,v,e)

Identify the set F_e

Add u,v to B, and extend B x B distances

Construct G_e and run Dijkstra

Multiple Failures



Query: (u,v,S)?

- $\tilde{O}(Mn^{1+\omega-\alpha})$ preprocessing: (not knowing u,v,e)

Construct the random graphs

Pick B, and compute B x B distances

- $\tilde{O}(n^{1+\alpha})$ query time: (knowing u,v,e)

Identify the set F_e

Add u,v to B, and extend B x B distances

Construct G_e and run Dijkstra

Thank You!