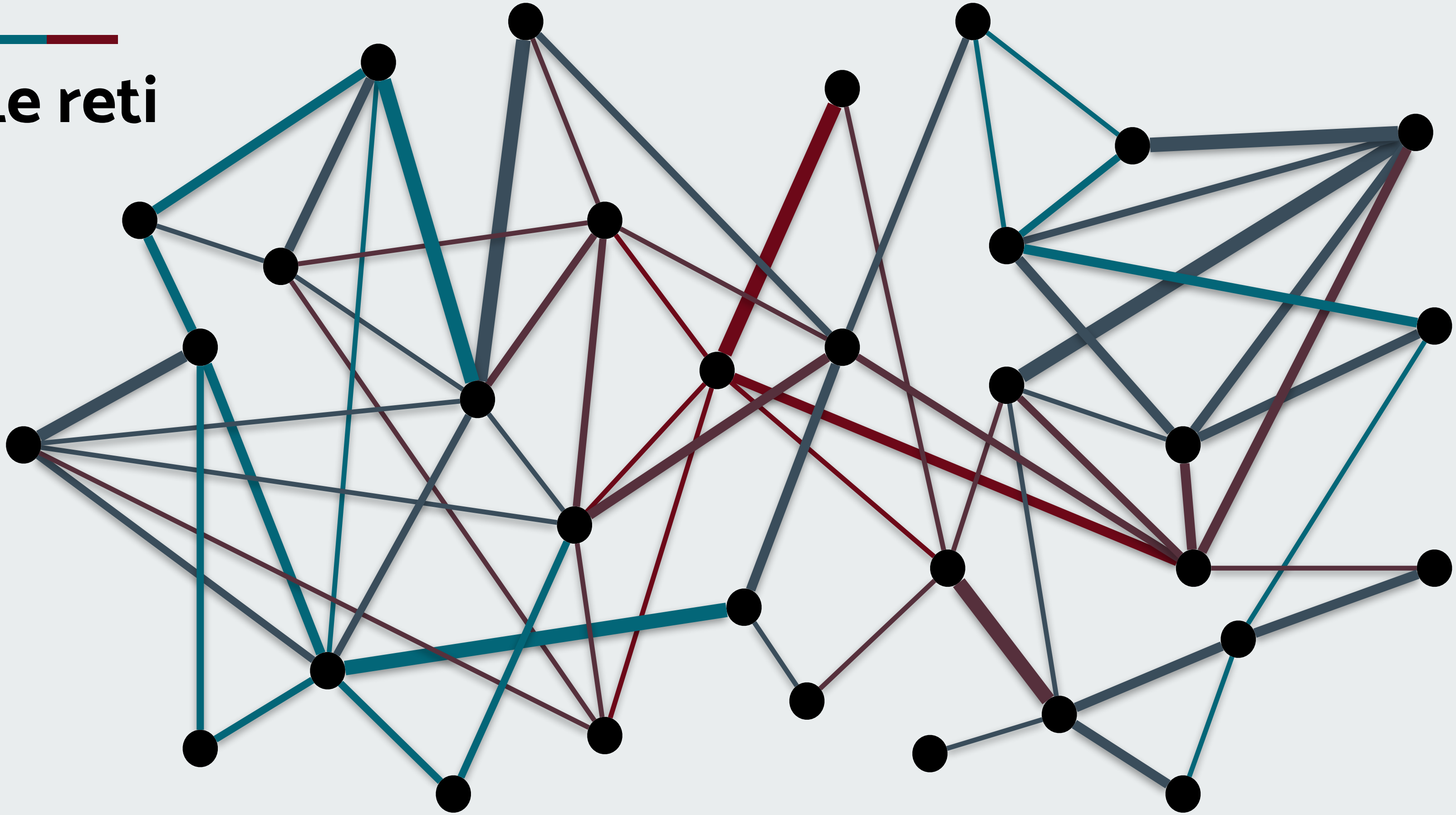
Achieving Max Flow in Strongly Polynomial Time for Sparse Networks

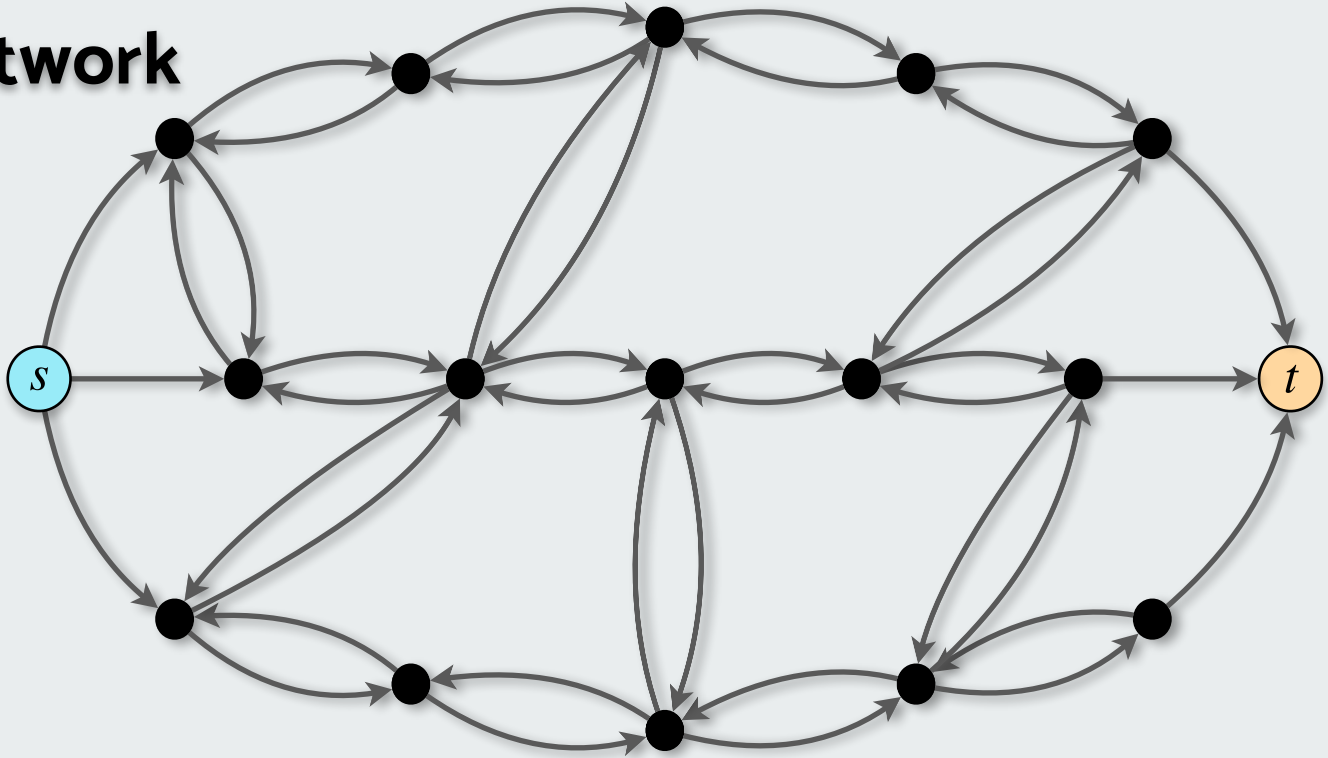
Beyond the Edmonds-Karp Algorithm

Armando Coppola - 2003964
Relatore - Paul Joseph Wollan
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di Laurea in Informatica
Sapienza Università di Roma
Anno Accademico 2023/24

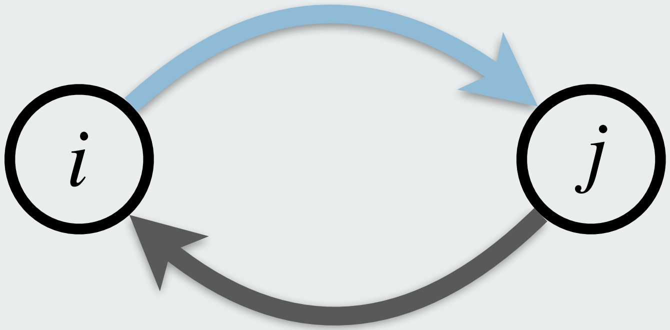
Le reti



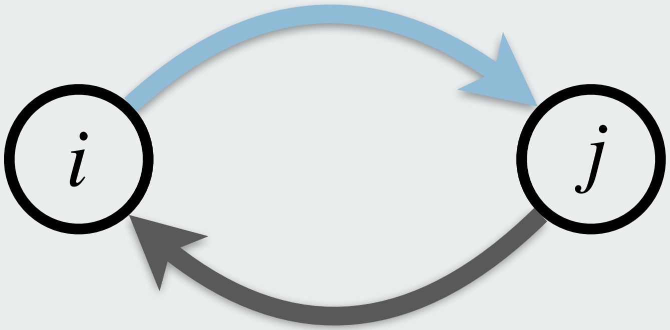
Il network



Com’è fatto un network



Com'è fatto un network



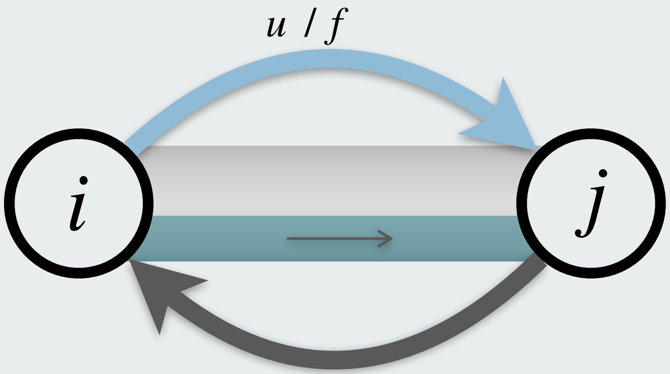
Vincolo di capacità

$$f(i,j) \le u(i,j)$$

Conservazione del flusso

$$\sum_{j:(j,i)\in E} f(j,i) = \sum_{j:(i,j)\in E} f(i,j)$$

Com'è fatto un network



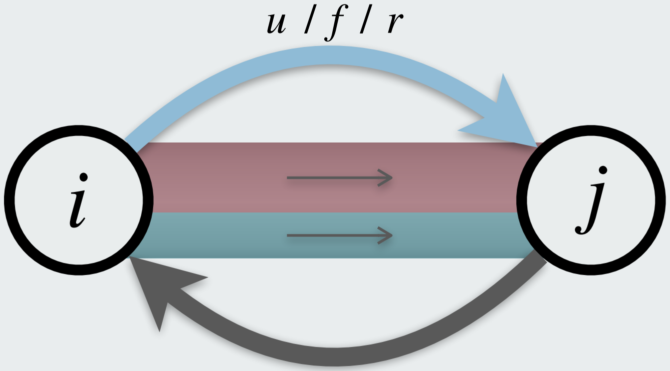
Vincolo di capacità

$$f(i,j) \le u(i,j)$$

Conservazione del flusso

$$\sum_{j:(j,i)\in E} f(j,i) = \sum_{j:(i,j)\in E} f(i,j)$$

Com'è fatto un network



Vincolo di capacità

$$f(i,j) \le u(i,j)$$

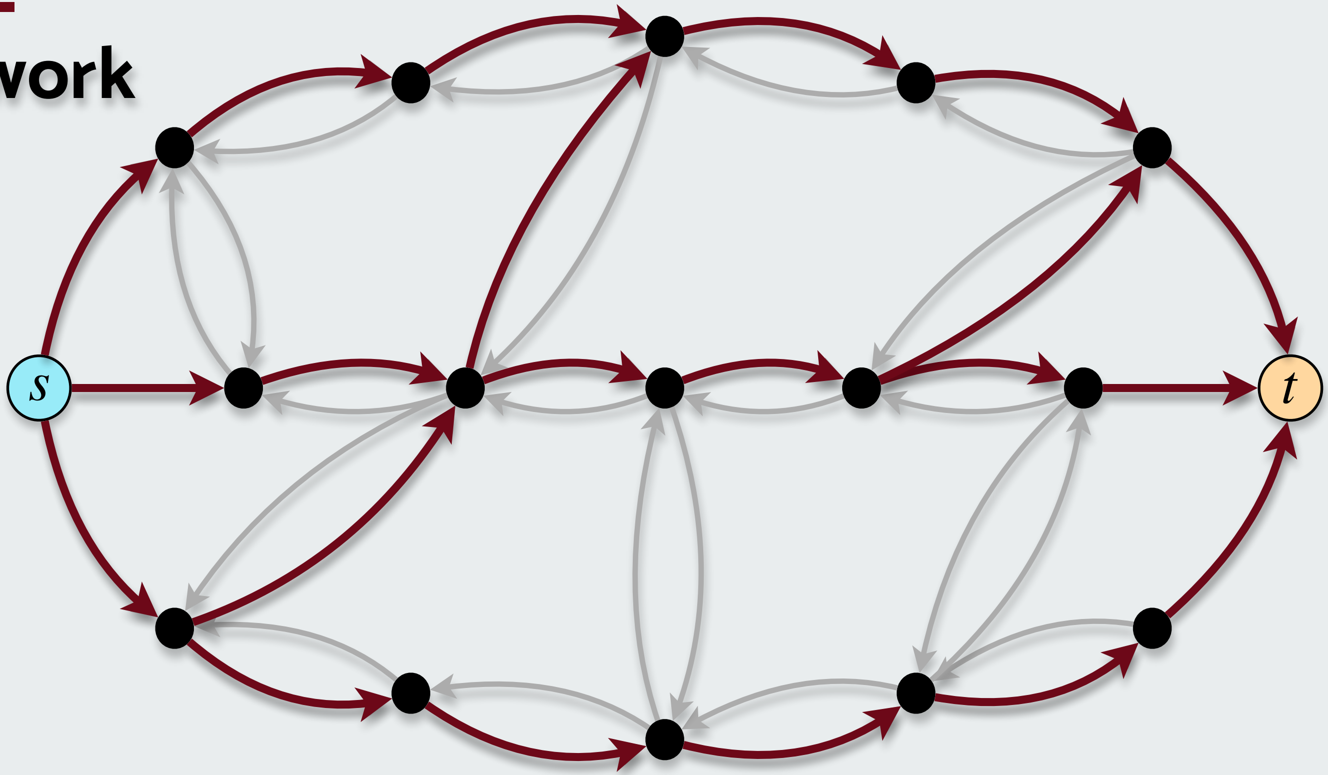
Conservazione del flusso

$$\sum_{j:(j,i)\in E} f(j,i) = \sum_{j:(i,j)\in E} f(i,j)$$

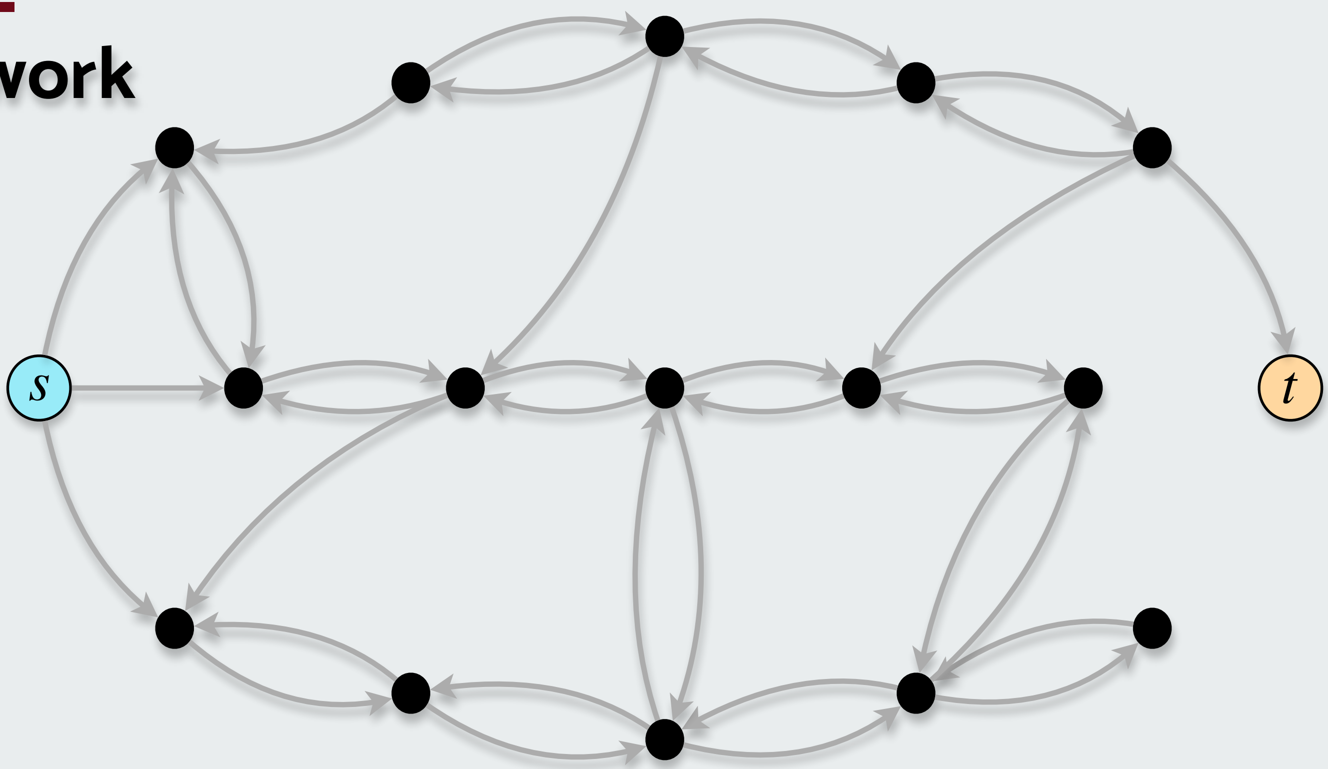
Capacità residua

$$r(i,j) = u(i,j) - f(i,j) + f(j,i)$$

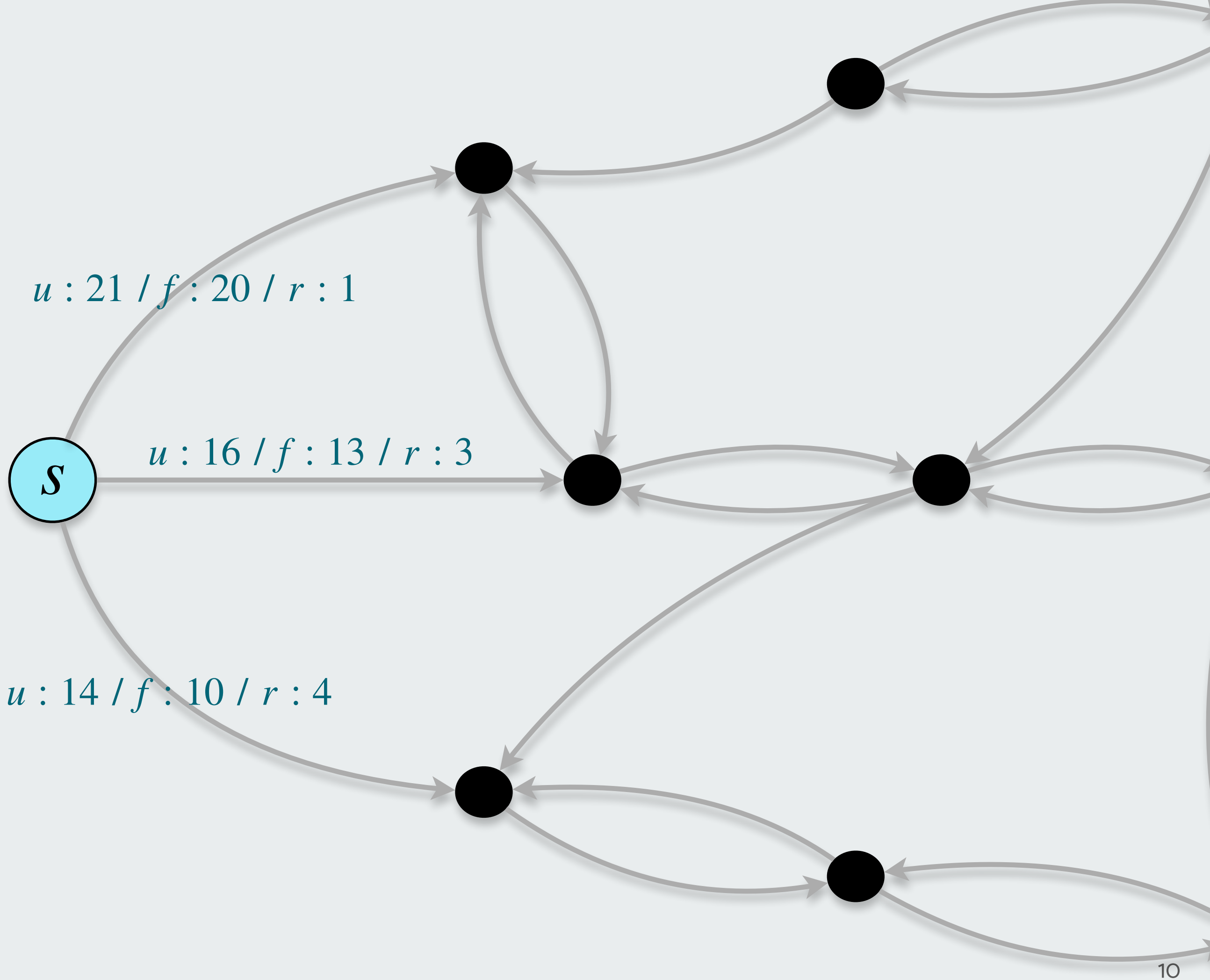
Network



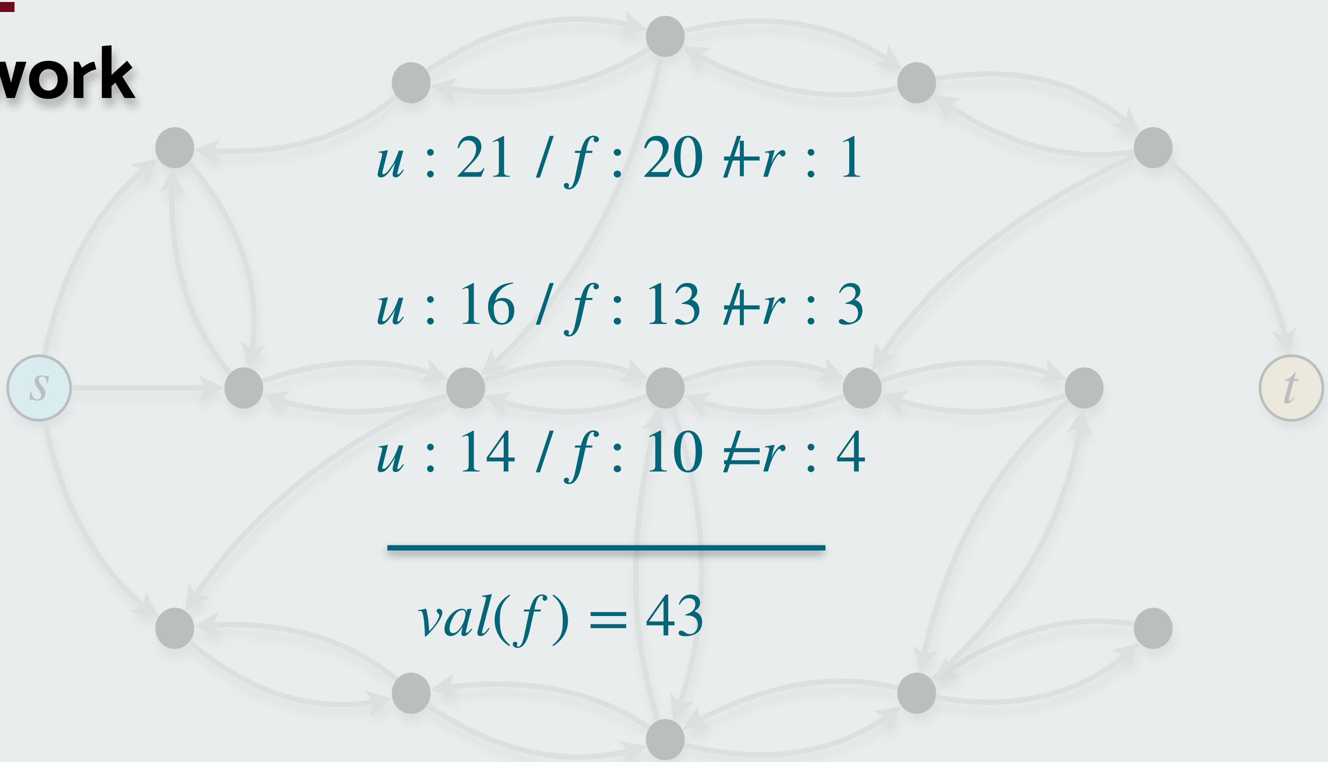
Network



Network



Network



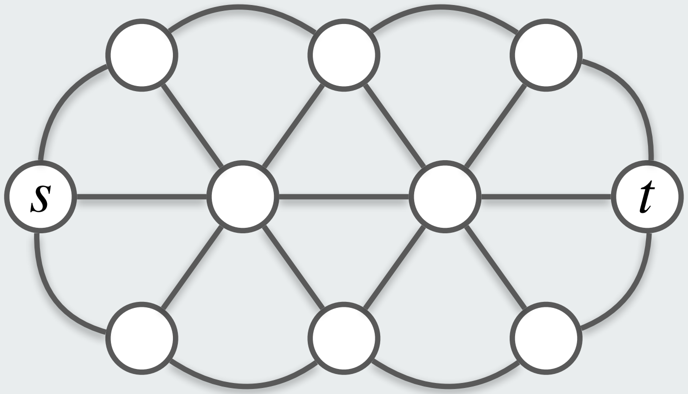
$$u : 21 \;/\; f : 20 \Rightarrow r : 1$$

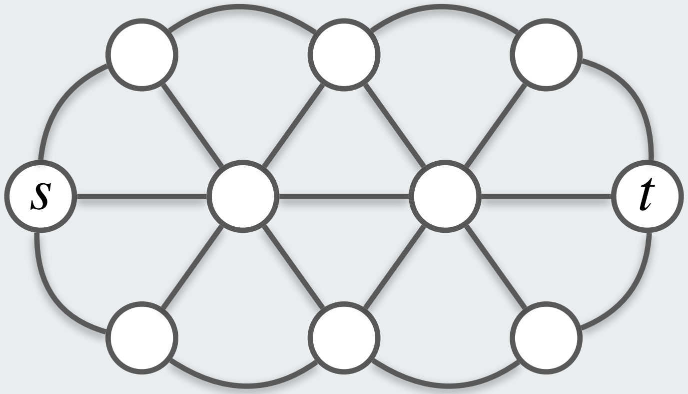
$$u : 16 \;/\; f : 13 \Rightarrow r : 3$$

$$u : 14 \;/\; f : 10 \Rightarrow r : 4$$

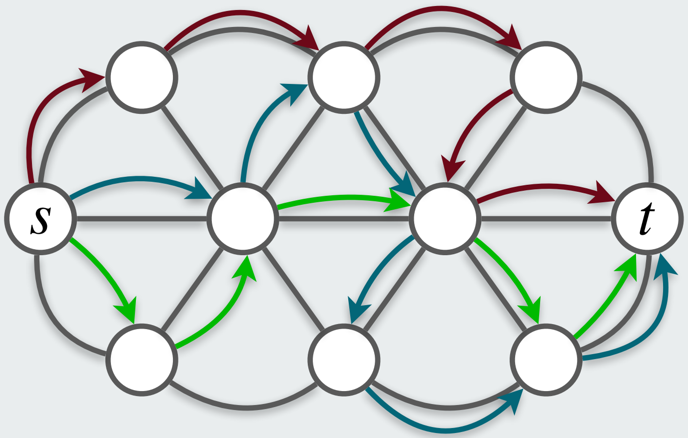
$$val(f) = 43$$

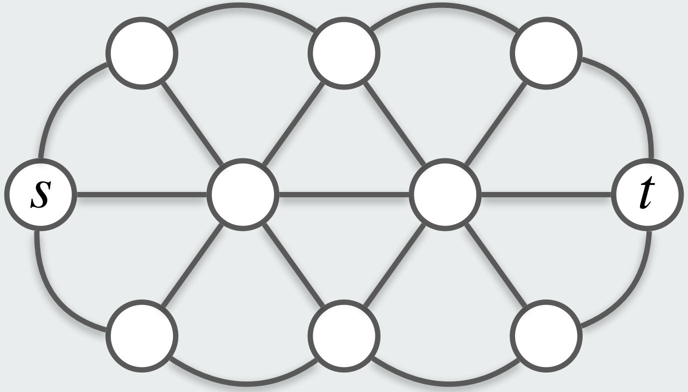
Ford-Fulkerson & Edmonds-Karp





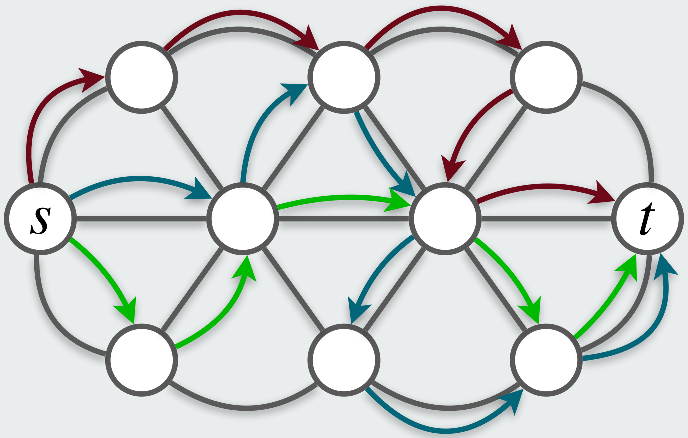
Ford-Fulkerson & Edmonds-Karp



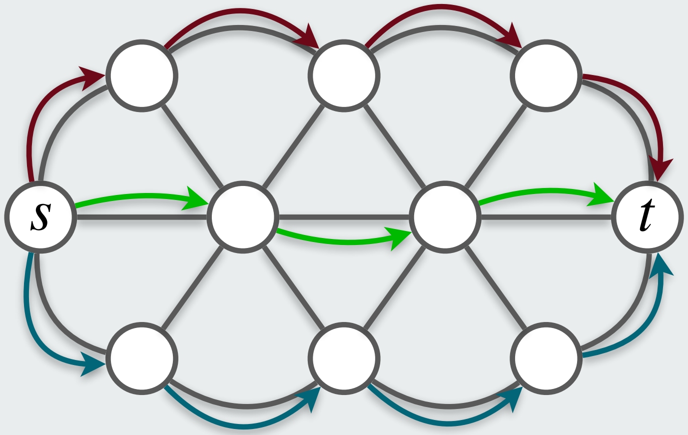


$O(m \cdot f_{max})$

Ford-Fulkerson & Edmonds-Karp

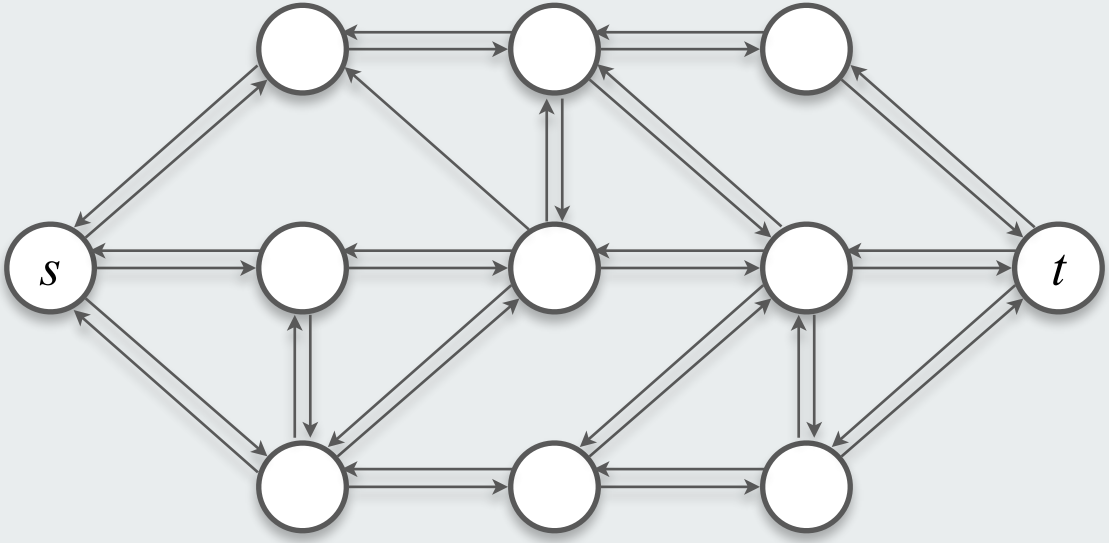


$O(m \cdot f_{max})$

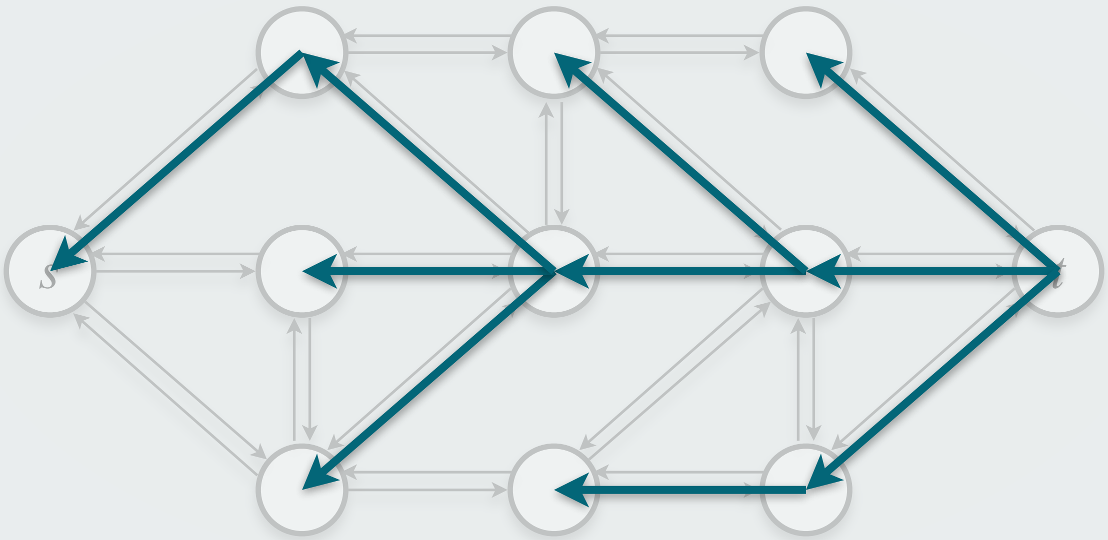


$O(m^2 \cdot n)$

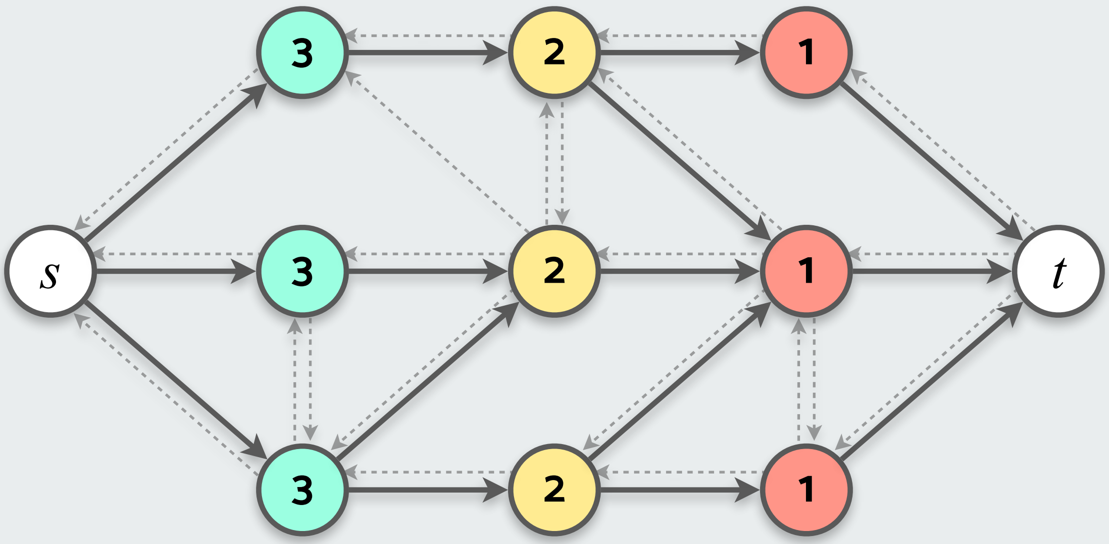
Algoritmo di Dinitz



Algoritmo di Dinitz

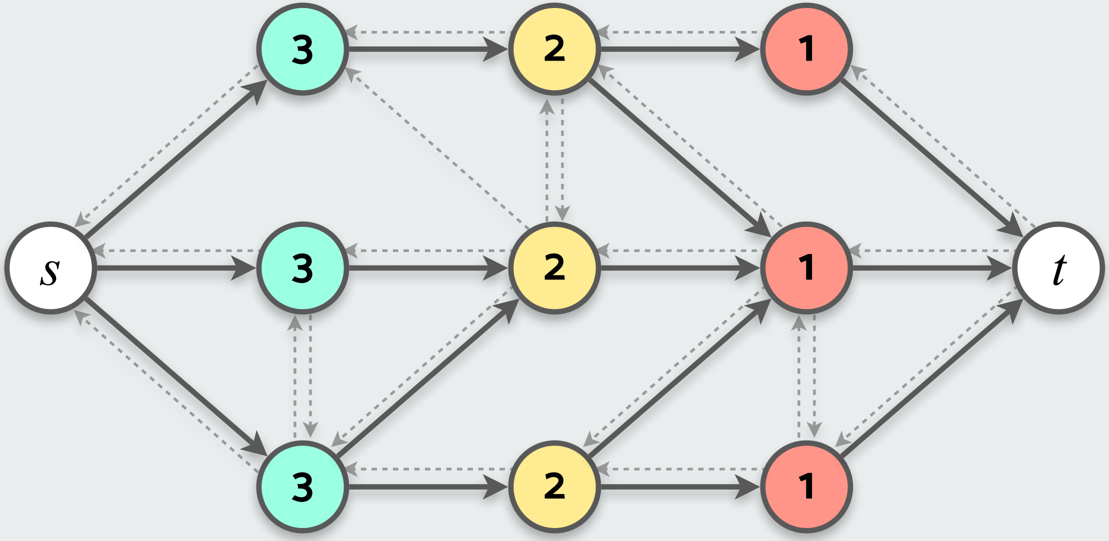


Algoritmo di Dinitz



Il grafo *admissible* “A”

Algoritmo di Dinitz



Il grafo *admissible* “A”

$$\forall (i, j) \in E(A)$$

$$d(i) = d(j) + l((i, j))$$

Flusso bloccante in $O(nm)$

Flusso massimo in $O(n^2m)$

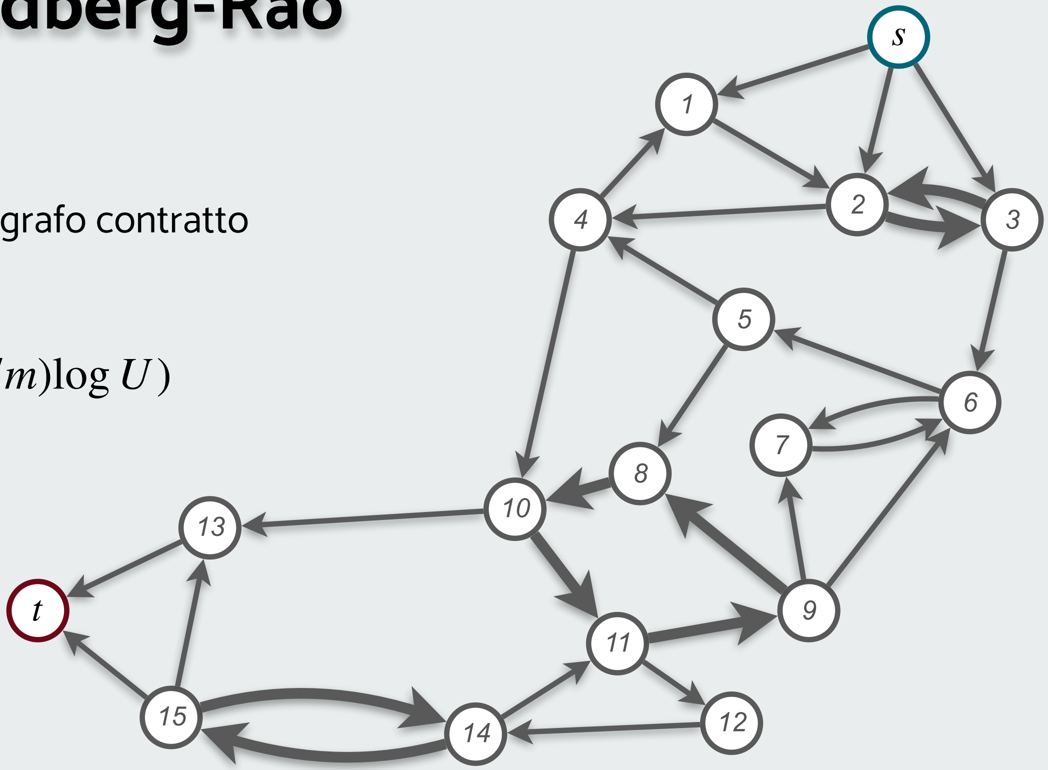
Algoritmo di Goldberg-Rao

Idea

Usare l'algoritmo di Dinitz su un grafo contratto

Costo computazionale

$$O(\min\{m^{1/2}, n^{2/3}\} \cdot m \log(n^2/m) \log U)$$



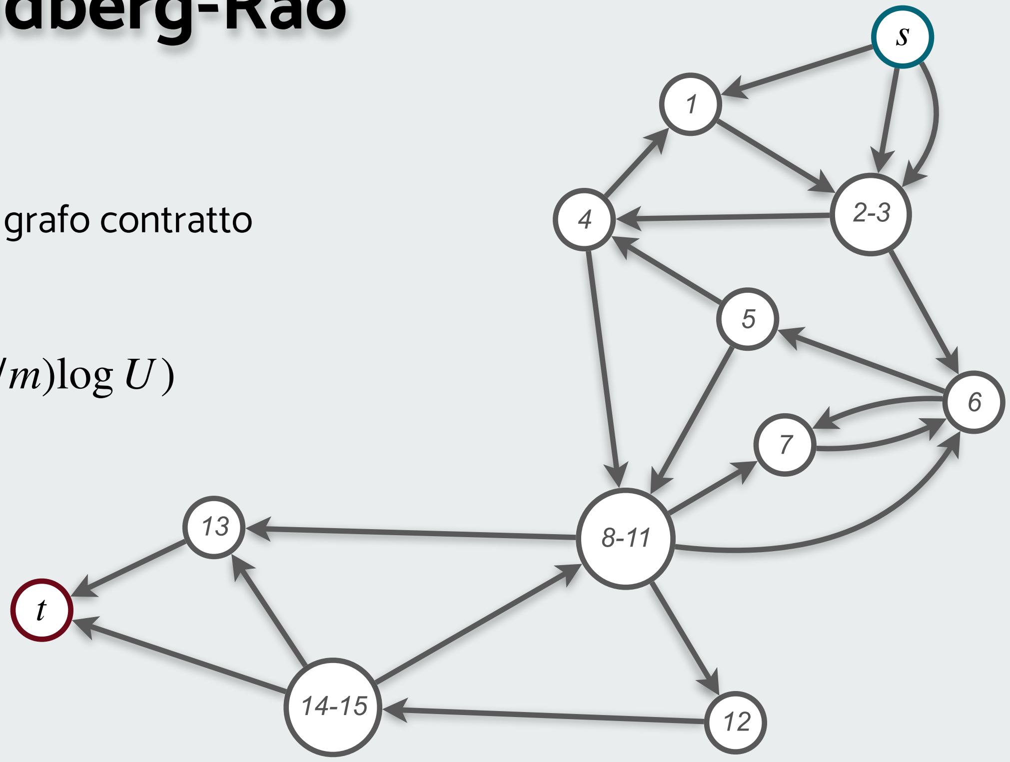
Algoritmo di Goldberg-Rao

Idea

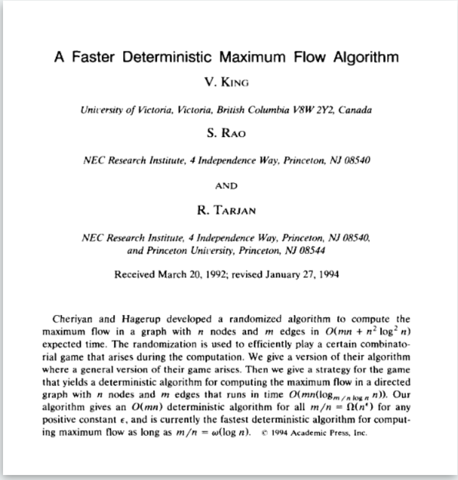
Usare l'algoritmo di Dinitz su un grafo contratto

Costo computazionale

$$O(\min\{m^{1/2}, n^{2/3}\} \cdot m \log(n^2/m) \log U)$$



Flusso massimo in O(nm): per grafi densi


A Faster Deterministic Maximum Flow Algorithm

V. King

University of Victoria, Victoria, British Columbia V8W 2Y2, Canada

S. Rao

NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

AND

R. Tarjan

NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, and Princeton University, Princeton, NJ 08544

Received March 20, 1992; revised January 27, 1994

Cheriyan and Hagerup developed a randomized algorithm to compute the maximum flow in a graph with n nodes and m edges in $O(mn + n^2 \log^2 n)$ expected time. The randomization is used to efficiently play a certain combinatorial game that arises during the computation. We give a version of their algorithm where a general version of their game arises. Then we give a strategy for the game that yields a deterministic algorithm for computing the maximum flow in a directed graph with n nodes and m edges that runs in time $O(mn(\log_{m/n \log n} n))$. Our algorithm gives an $O(mn)$ deterministic algorithm for all $m/n = \Omega(n^\epsilon)$ for any positive constant ϵ, and is currently the fastest deterministic algorithm for computing maximum flow as long as $m/n = \omega(\log n)$.

Autori

V. King, S. Rao, R. Tarjan; 1994

Complessità

$$O(nm \log_{\frac{m}{n \log n}} n),$$

$$O(nm) \text{ se } m/n = \Omega(n^{\varepsilon})$$

Flusso massimo in O(nm): per grafi sparsi

Max flows in $O(nm)$ time, or better

James B. Orlin*

Revised: July 25, 2012

Abstract

In this paper, we present improved polynomial time algorithms for the max flow problem defined on a network with n nodes and m arcs. We show how to solve the max flow problem in $O(nm)$ time, improving upon the best previous algorithm due to King, Rao, and Tarjan, who solved the max flow problem in $O(nm \log_{m/(n \log n)} n)$ time. In the case that $m = O(n)$, we improve the running time to $O(n^2/\log n)$.

We further improve the running time in the case that $U^* = U_{\max}/U_{\min}$ is not too large, where $U_{\max}$ denotes the largest finite capacity and $U_{\min}$ denotes the smallest non-zero capacity. If $\log(U^*) = O(n^{1/3} \log^{-3} n)$, we show how to solve the max flow problem in $O(nm/\log n)$ steps. In the case that $\log(U^*) = O(\log^k n)$ for some fixed positive integer k, we show how to solve the max flow problem in $\widetilde{O}(n^{8/3})$ time. This latter algorithm relies on a subroutine for fast matrix multiplication.

Autore

J. B. Orlin, 2013

Complessità

$$O(nm + m^{31/16} \log^2 n),$$

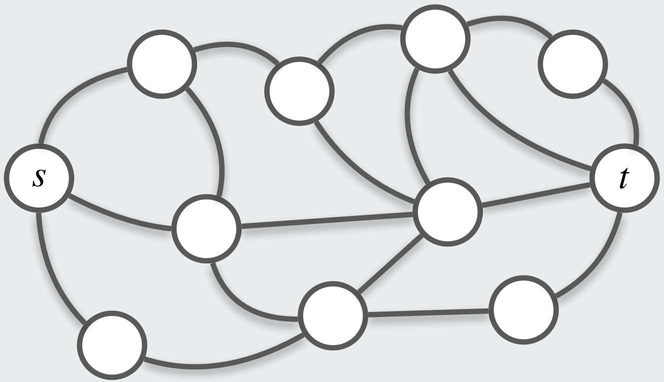
$$O(nm) \text{ se } m = O(n^{1+\varepsilon})$$

L'algoritmo di Orlin: Intuizione

- Complessità del Goldberg-Rao: $O(m^{1/2} m \log \frac{n^2}{m} \log U)$
- $\log U < m^{7/16} \implies \tilde{O}(m^{3/2} \cdot m^{7/16}) = \tilde{O}(m^{31/16})$
- Se il grafo è abbastanza sparso:

$m = O(n^{16/15 - \varepsilon}) \implies$

$\tilde{O}(m \cdot m^{15/16}) = \tilde{O}(m \cdot n^{1-\varepsilon(15/16)}) = O(nm)$

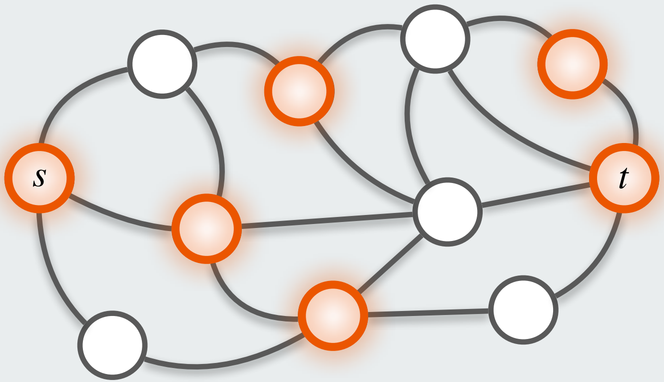


L'algoritmo di Orlin: Intuizione

- Complessità del Goldberg-Rao: $O(m^{1/2} m \log \frac{n^2}{m} \log U)$
- $\log U < m^{7/16} \implies \tilde{O}(m^{3/2} \cdot m^{7/16}) = \tilde{O}(m^{31/16})$
- Se il grafo è abbastanza sparso:

$$m = O(n^{16/15-\varepsilon}) \implies$$

$$\tilde{O}(m \cdot m^{15/16}) = \tilde{O}(m \cdot n^{1-\varepsilon(15/16)}) = O(nm)$$

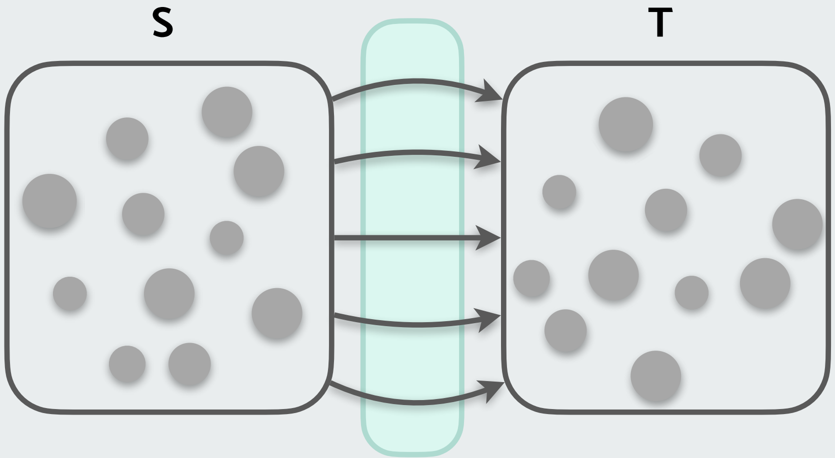


- $C = O\left(\frac{m}{\log U}\right)$
- $\log U \leq m^{7/16} \iff C \geq m^{9/16}$

$O(m)$ nodi Δ-critici in tutte le iterazioni

Il parametro Δ

Scegliamo un bound $\Delta = val(f(S,T))$



Il **taglio canonico** minore

può essere trovato in $O(m)$

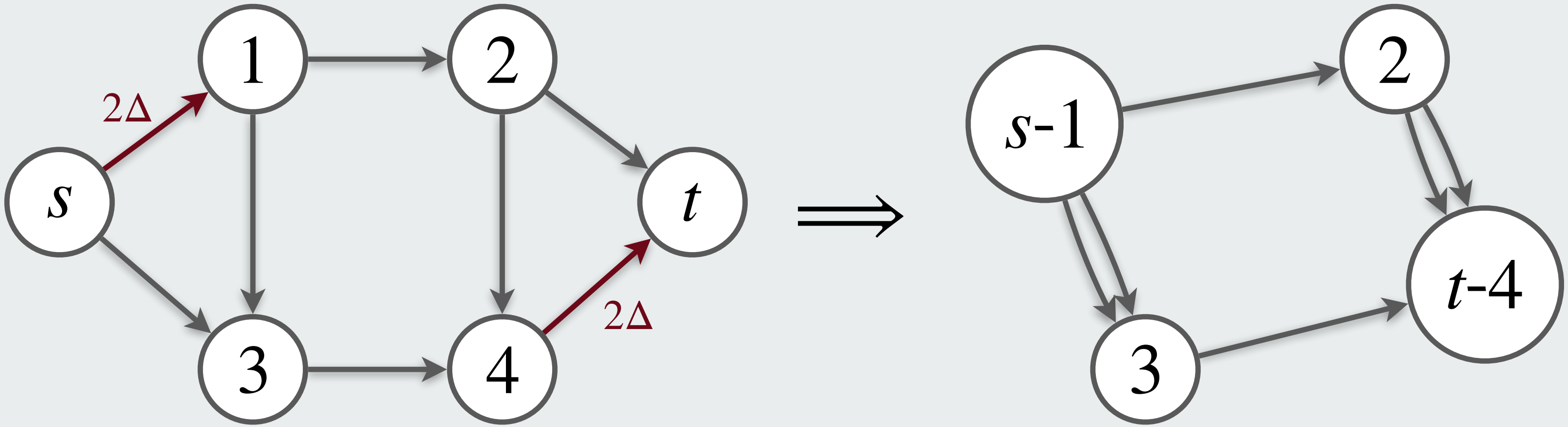
Il flusso massimo in ogni

fase di incremento è $\leq \Delta$

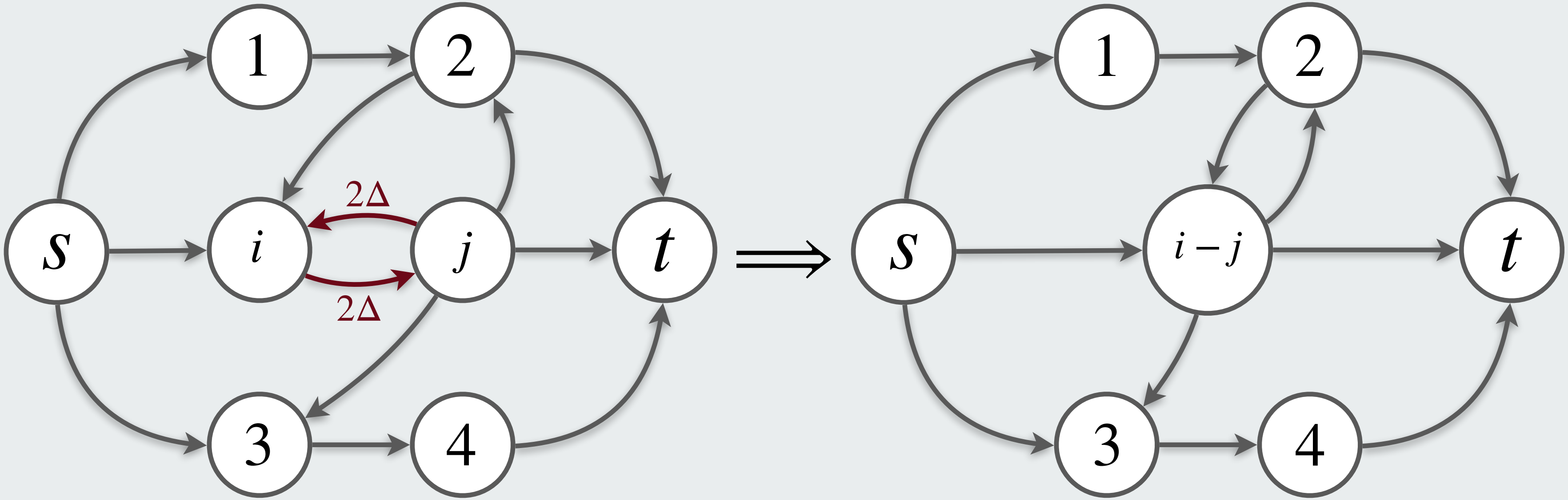
Tutti gli archi di capacità $\geq 2\Delta$

vengono definiti **abbondanti**

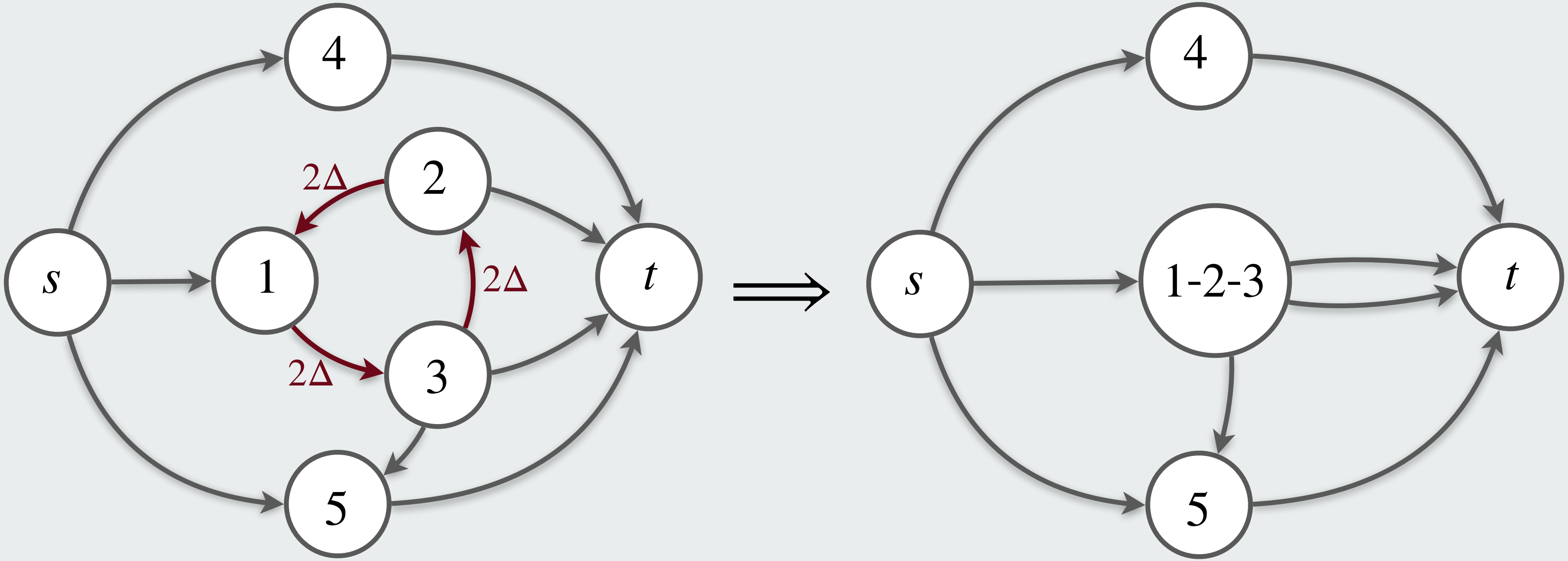
Il network contratto



Il network contratto

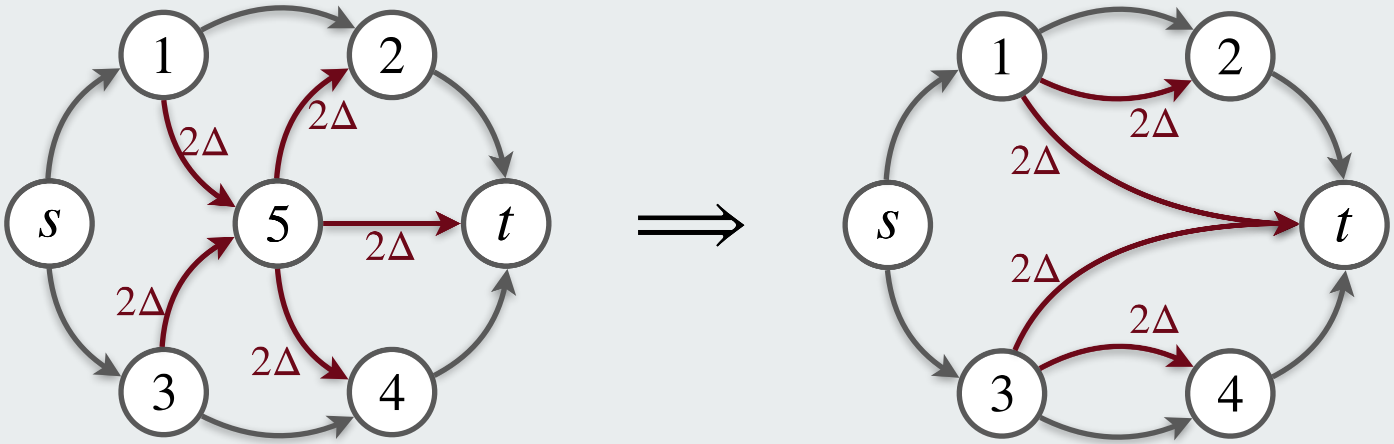


Il network contratto



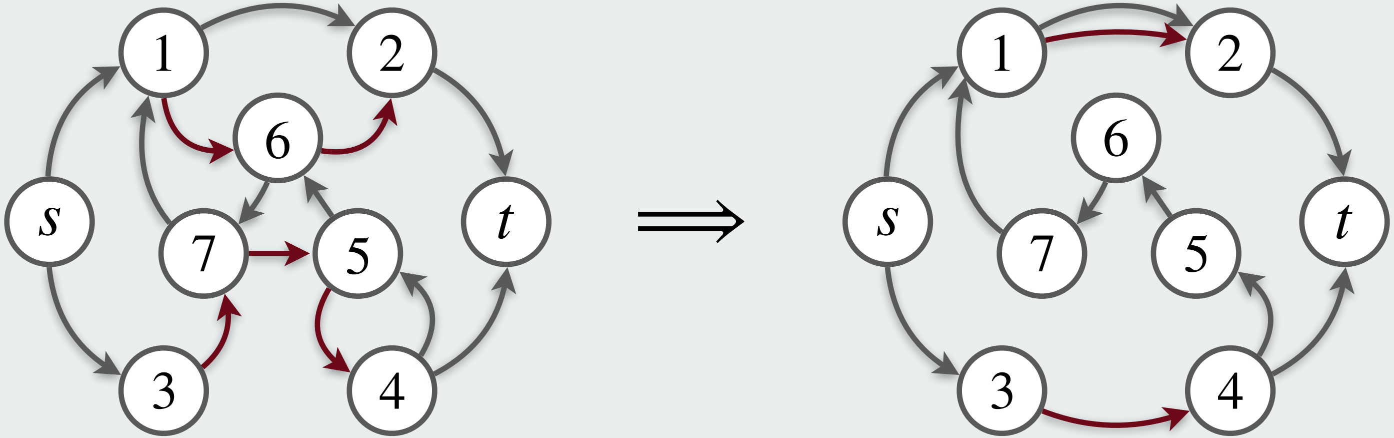
Il grafo compatto

Strongly Compact



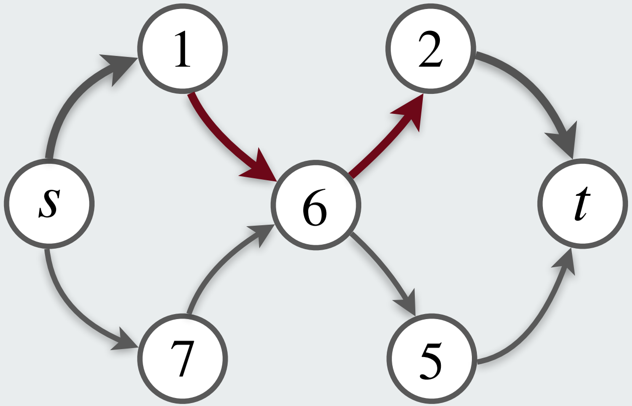
Il grafo compatto

Trasferimento di capacità



Il grafo compatto

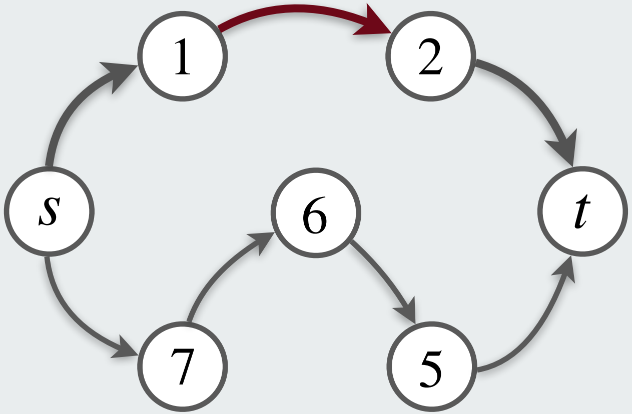
Approssimazione



- Bassa capacità se $u_{ij} + u_{ji} < \Delta/(64m^3)$
- Capacità persa durante la compattazione: $\leq \Delta/16m$
- Incremento previsto: $f^* = \Delta/8m$-optimal flow $f_{max} - \Delta/8m \leq f^* \leq f_{max}$

Il grafo compatto

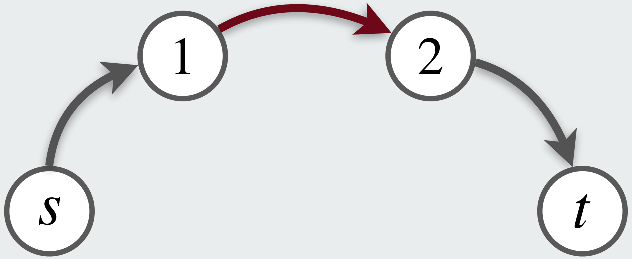
Approssimazione



- Bassa capacità se $u_{ij} + u_{ji} < \Delta/(64m^3)$
- Capacità persa durante la compattazione: $\leq \Delta/16m$
- Incremento previsto: $f^* = \Delta/8m$-optimal flow
  $f_{max} - \Delta/8m \leq f^* \leq f_{max}$

Il grafo compatto

Approssimazione



- Bassa capacità se $u_{ij} + u_{ji} < \Delta/(64m^3)$
- Capacità persa durante la compattazione: $\leq \Delta/16m$
- Incremento previsto: $f^* = \Delta/8m$-optimal flow
  $f_{max} - \Delta/8m \leq f^* \leq f_{max}$

L’algoritmo

L’algoritmo

1. $\Delta = r(S, T)$
2. $C = |N^C|$

L'algoritmo

1. $\Delta = r(S, T)$
2. $C = |N^C|$
3. Se $C \geq m^{9/16}$
    1. Goldberg-Rao $\rightarrow \Delta / 8m$-optimal flow

L'algoritmo

1. $\Delta = r(S,T)$
2. $C = |N^C|$
3. Se $C \geq m^{9/16}$
    1. Goldberg-Rao $\rightarrow \Delta/8m$-optimal flow
4. Se $m^{1/3} \leq C \leq m^{9/16}$
    1. Creazione del grafo Δ-compatto
    2. Goldberg-Rao $\rightarrow \Delta/8m$-optimal flow
    3. Induzione flusso sul network originale

L'algoritmo

1. $\Delta = r(S, T)$
2. $C = |N^C|$
3. Se $C \geq m^{9/16}$
    1. Goldberg-Rao $\rightarrow \Delta/8m$-optimal flow
4. Se $m^{1/3} \leq C \leq m^{9/16}$
    1. Creazione del grafo Δ-compatto
    2. Goldberg-Rao $\rightarrow \Delta/8m$-optimal flow
    3. Induzione flusso sul network originale
5. Se $C \leq m^{1/3}$
    1. Scegliere Γ bilanciato

L'algoritmo

1. $\Delta = r(S, T)$
2. $C = |N^C|$
3. Se $C \geq m^{9/16}$
    1. Goldberg-Rao $\rightarrow$ $\Delta/8m$-optimal flow
4. Se $m^{1/3} \leq C \leq m^{9/16}$
    1. Creazione del grafo Δ-compatto
    2. Goldberg-Rao $\rightarrow$ $\Delta/8m$-optimal flow
    3. Induzione flusso sul network originale
5. Se $C \leq m^{1/3}$
    1. Scegliere Γ bilanciato
    2. Creazione del grafo Γ-compatto
    3. Goldberg-Rao $\rightarrow$ optimal flow
    4. Induzione flusso sul network originale

L’algoritmo

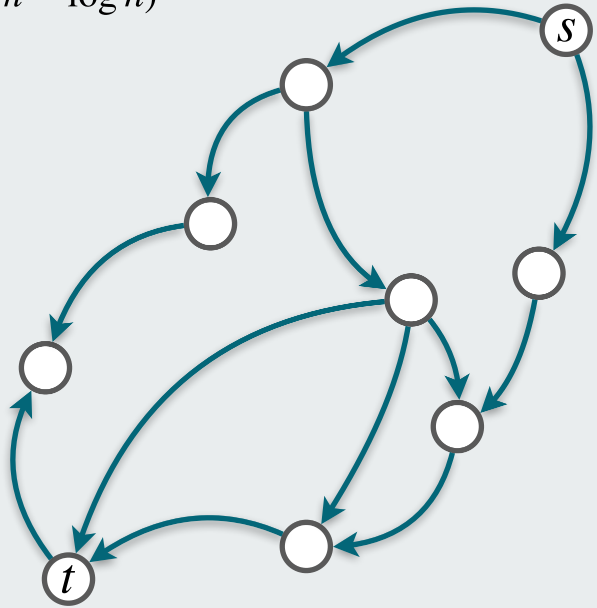
1. $\Delta = r(S, T)$
2. $C = |N^C|$
3. Se $C \geq m^{9/16}$
    1. Goldberg-Rao $\rightarrow$ $\Delta/8m$-optimal flow
4. Se $m^{1/3} \leq C \leq m^{9/16}$
    1. Creazione del grafo Δ-compatto
    2. Goldberg-Rao $\rightarrow$ $\Delta/8m$-optimal flow
    3. Induzione flusso sul network originale
5. Se $C \leq m^{1/3}$
    1. Scegliere Γ bilanciato
    2. Creazione del grafo Γ-compatto
    3. Goldberg-Rao $\rightarrow$ optimal flow
    4. Induzione flusso sul network originale

Costo della procedura per tutte le iterazioni:

$O(m^{31/16} \log^2 n)$

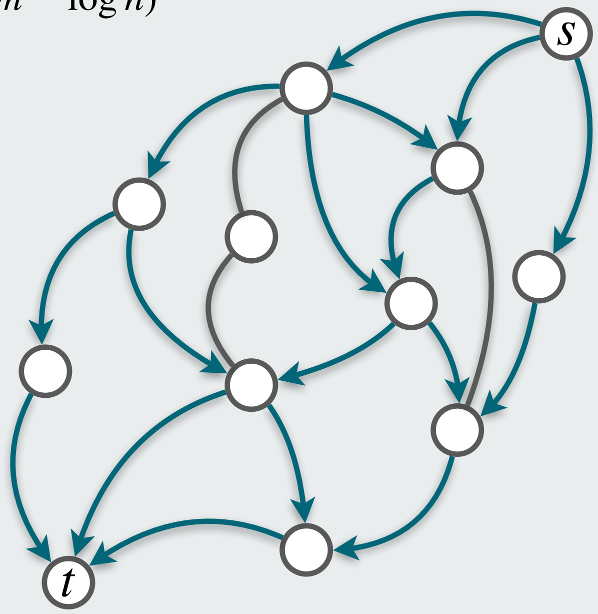
Bottlenecks e complessità finale

- **Adattare** il flusso dal grafo contratto a quello originale: $O(nm + m^{5/3} \log n)$
- Mantenere la **chiusura transitiva** degli archi abbondanti: $O(nm)$
- **Complessità** finale: $O(nm + m^{31/16} \log^2 n)$
- Se $m = O(n^{1.06})$ allora il **costo finale** è di $O(nm)$



Bottlenecks e complessità finale

- **Adattare** il flusso dal grafo contratto a quello originale: $O(nm + m^{5/3} \log n)$
- Mantenere la **chiusura transitiva** degli archi abbondanti: $O(nm)$
- **Complessità** finale: $O(nm + m^{31/16} \log^2 n)$
- Se $m = O(n^{1.06})$ allora il **costo finale** è di $O(nm)$



Ulteriori miglioramenti

- Soluzione di Orlin per $m = O(n)$ raggiunge costo $O(n^2/\log n)$
- Orlin e Xiao-Yue Gong nel 2019, per $m = O(n \log n)$ dominano il King et al. di $\log \log n$

Ford-Fulkerson

$O(mf_{max})$

Edmonds-Karp

$O(nm^2)$

Dinitz

$O(n^2m)$

Goldberg-Rao

$$O\left(m^{3/2}\log\frac{n^2}{m}\log U_{max}\right)$$

Orlin

$O(nm + m^{31/16}\log^2 n)$

Grazie per l’attenzione!