Università di Bergamo

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

Reti di Telecomunicazione

Prof. Fabio Martignon

Università di Bergamo

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

4 – Routing

Reti di Telecomunicazione

Instradamento

- Parametri e requisiti
- Tecniche di instradamento deterministiche
- Tecniche di instradamento dinamiche
- Instradamento multicast e broadcast
- Routing Ottimo
- Instabilità

Instradamento

- **L'instradamento è la funzione dello Strato Rete che permette l'utilizzazione contemporanea delle risorse di rete da parte di diverse relazioni di traffico.**
- **L'instradamento richiede la funzione di commutazione nei nodi della rete.**

- **Si richiede efficienza nell'utilizzo di tale risorse, per poter smaltire il traffico offerto alla rete …**
- **… e l'ottimizzazione nell'uso dei canali di trasmissione quando la topologia della rete ammette più cammini fra ogni coppia di nodi.**

Instradamento

- **La politica di instradamento specifica il flusso medio, x_p , in ogni cammino fra i nodi *i* e *j* in modo tale che**

$$\sum_{\substack{\text{su tutti i}\\ \text{cammini fra (i, j)}}} x_p = \gamma_{ij} \longleftarrow \text{Traffico offerto tra i e j}$$

- **rispettando i vincoli di:**
  - **Fattibilità:**

    somma dei flussi in un lato ≤ capacità del lato
  - **Solenoidalità**

    somma dei flussi entranti in un nodo (arco) = somma dei flussi uscenti dal nodo (arco)

Instradamento

- **Il traffico massimo smaltito da una rete definisce la *capacità di rete*, C**

$$C = \max_{\text{instradamenti}} S(I, A)$$

ove S = traffico smaltito, funzione di

I = Instradamento

A = matrice delle relazioni di traffico offerto alla rete

- **In generale C dipende da**
  - **Topologia della rete**
  - **Capacità dei canali**
  - **Matrice delle relazioni di traffico (ovvero: A)**

Instradamento

- **Se:**
  - **gli *m* canali di rete sono tutti uguali con velocità μ pck/s e ritardo di trasferimento pari a τ secondi**
  - **L è numero medio di trasmissioni di un pacchetto in rete = lunghezza media del cammino misurata in tratte.**
- **allora si ha:**
  - ✓ **Lτ = tempo medio di permanenza in rete**
  - ✓ **$m\ \mu\ \tau$ = numero medio di pacchetti in rete (a pieno regime)**

  $\gamma \cdot L\ \tau = m\ \mu\ \tau$ (per il teorema di Little)

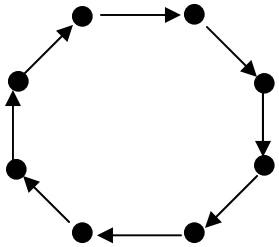
  essendo $m\ \mu\ \tau$ il numero massimo di pacchetti in rete γ = C

$$C = \frac{m\mu}{L} \text{ pacc/s}$$

- **Commenti**
  - **vale il segno di uguale se tutti i canali sono pieni, se no** $C \le \frac{m\mu}{L}$
  - **indipendente da τ**

Esempi

- **Rete ad anello con canali unidirezionali (Token Ring)**



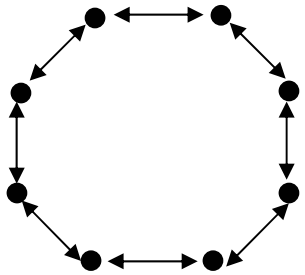
C = 2 μ

M = *m* = 8

L = 4

$$L = \frac{\sum_{i=1}^{M-1} i}{M-1} = \frac{(M-1)M}{2(M-1)} = \frac{M}{2}$$

- **Rete ad anello con canali bidirezionali (capacità ripartita nelle due direzioni)**



C = 4 μ

$$L = \frac{M}{4}$$

(per M pari)

Tecniche di Instradamento

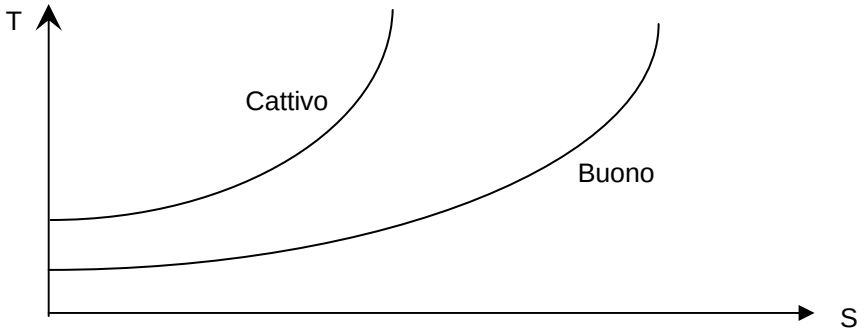
- **Definizione**
  - **Insieme di regole secondo le quali il traffico esterno entrante viene instradato verso la propria destinazione.**
- **Datagram: la tecnica di instradamento agisce sui singoli pacchetti (ogni pck ha storia a sé)**
- **Virtual call: la tecnica di instradamento agisce sui pacchetti di richiesta di chiamata. La strada è fissata per ogni circuito virtuale (instradamento di sessione).**
- **Le tecniche di instradamento coinvolgono varie procedure anche complesse che possono lavorare indipendentemente ma che devono supportarsi a vicenda e scambiarsi informazioni.**

Tecniche di Instradamento

- **Complessità dovuta a:**
  - **Coordinamento fra tutti i nodi della rete (a livello Data link solo 2 nodi)**
  - **Funzionamento anche in presenza di guasti nei collegamenti e nodi. Reinstradamento del traffico e aggiornamento di tabelle.**
  - **Dinamicità nelle scelte dovuto a cambiamenti nel traffico. Per ottenere migliori prestazioni.**
- **Nota: non tutti gli instradamenti sono dinamici. Possono anche essere statici.**

Tecniche di Instradamento e Prestazioni

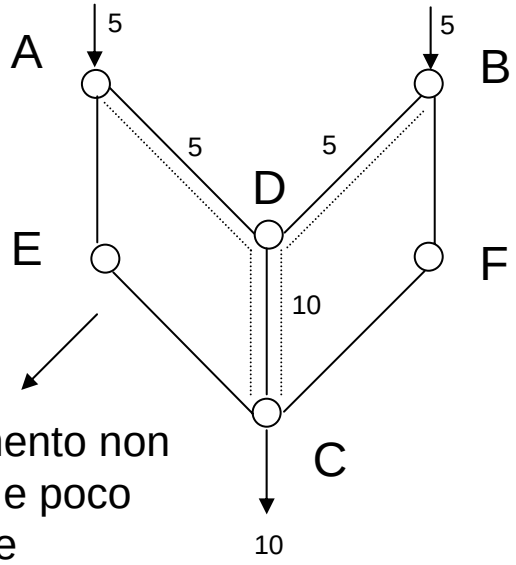
- **Prestazioni che dipendono dalle tecniche di instradamento, ovvero criteri di valutazione delle tecniche di instradamento**
  - **Throughput S**
  - **Ritardo medio T**
  - **Affidabilità A**



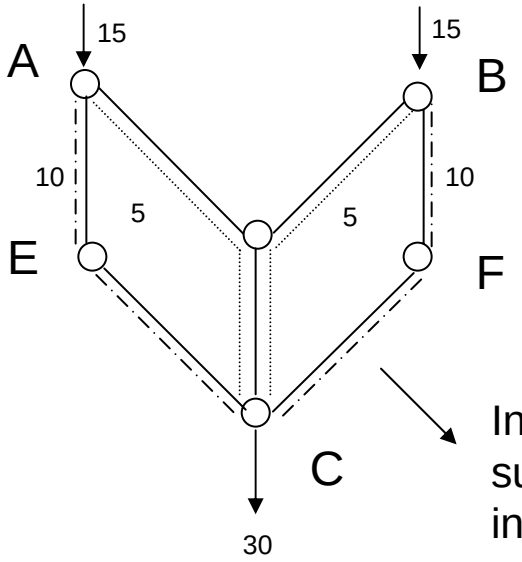
Nota: tali curve possono incrociarsi.
Si individuano allora delle aree di convenienza delle diverse tecniche di instradamento, a seconda del traffico smaltito S.

Esempio

- **Capacità di ogni lato uguale a 10 unità di banda (udb)**



Instradamento non suddiviso e poco intelligente



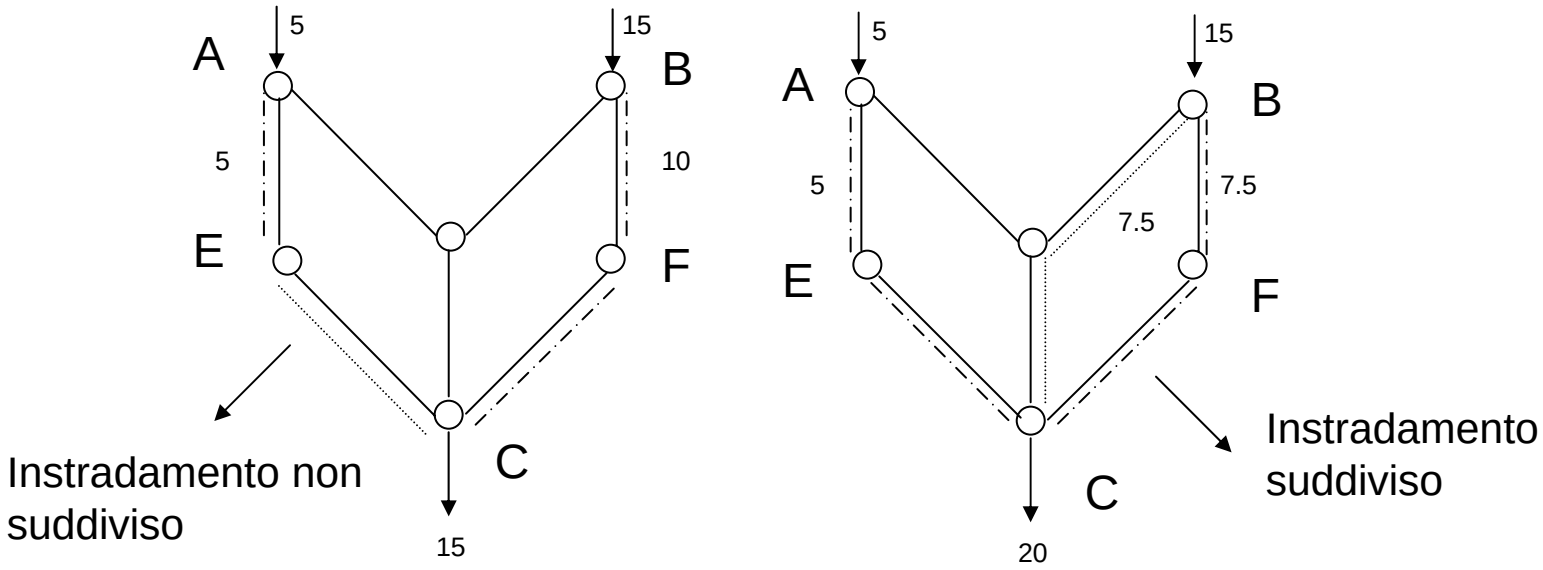
Instradamento suddiviso e intelligente

Più di 10 udb non posso mandare
Sul lato DC ho ritardi elevatissimi

- **Il throughput massimo ottenibile varia da 10 a 30 udb, a seconda dell'instradamento scelto.**

Esempio (2)

- **Capacità di ogni lato uguale a 10 unità di banda**



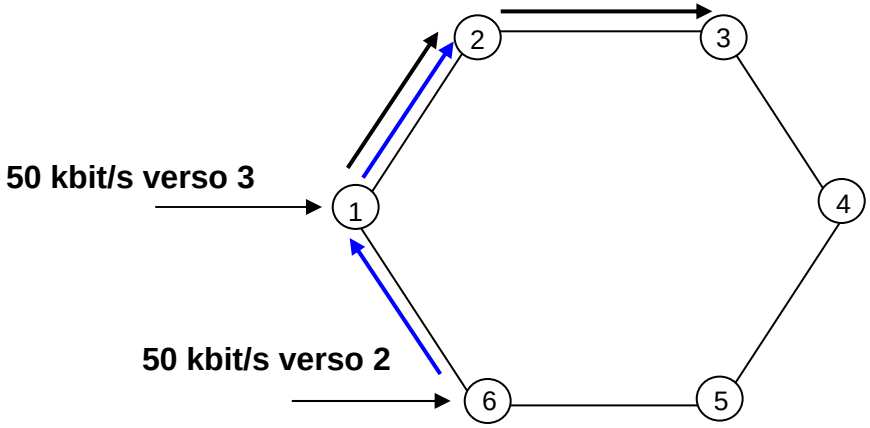
- Se non posso splittare il traffico, allora devo rifiutare 5 udb al nodo B (come minimo) e instradare lungo A-E-C e B-F-C (accetto massimo 15 udb totali)
- Se il traffico in B può essere splittato, ma in modo uguale, allora uso: A-E-C (5 udb) e poi B-F-C (7.5 udb) e B-D-C (7.5 udb). Totale traffico accettato: 20 udb.

Requisiti delle Tecniche di Instradamento

- **Requisiti:**
  - **Semplicità: di descrizione e programmazione (diversi calcolatori, tempo reale)**
  - **Affidabilità: possa reagire quando nodi e lati si guastano**
  - **Stabilità: a valori di traffico fissi deve fornire una soluzione stabile ed accurata, che non dipende da come è evoluto il sistema**
  - **Adattamento: alle variazioni della topologia della rete e del traffico. Velocità di adattamento**
  - **Ottimità globale: ottimizzare il traffico totale smaltito dalla rete**
  - **Fairness (eguaglianza) con cui tratto gli utenti della rete**

Requisiti Tecniche Instradamento

Ottimità globale

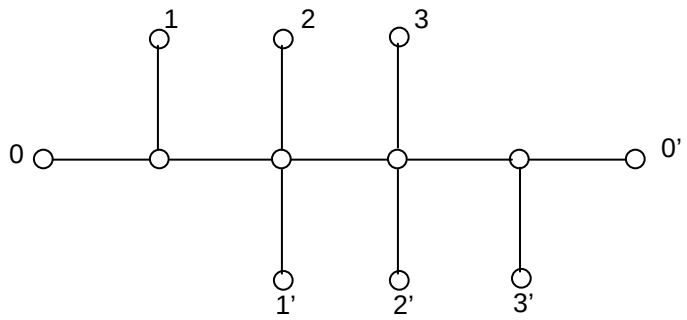


Tutti i link sono full-duplex ed hanno capacità pari a 50 kbit/s

- **In generale, tenere basso il numero di tratte (link) utilizzate massimizza la capacità del sistema, soprattutto ad alti carichi (Si ricordi l'espressione di $C = \frac{m\mu}{L}$)**
- **Tuttavia in questo caso tale criterio non andrebbe bene: il link (1,2) si vedrebbe offerto un traffico di 100 kbit/s, superiore alla sua capacità**

Requisiti Tecniche Instradamento

Eguaglianza (Fairness)



Ogni link ha capacità unitaria (1 unità di banda)
Tutte le coppie di nodi offrono alla rete 1 unità di banda

- **Instradando tutto il traffico proveniente dai nodi 1, 2 e 3, saturo i 3 link centrali, ed impedisco al nodo 0 di trasmettere. Tuttavia ho accettato 3 u.d.b. nella rete**
- **Se invece instrado 0.5 u.d.b. dei traffici 1,2 e 3, posso instradare anche 0.5 u.d.b. del traffico proveniente da 0. In totale ho accettato solo 2 u.d.b, ma ho trattato equamente i 4 utenti della rete**

Tecniche di Instradamento: Classificazione

- **Possono essere classificate in:**
  - **Tecniche Deterministiche: assegnate "una volta per tutte" in fase di progettazione/configurazione della rete**
    - ✓ **Diffusione o Flooding (Inondazione)**
    - ✓ **Instradamento Fissato**
    - ✓ **Instradamento Suddiviso**
  - **Tecniche Dinamiche**

Diffusione (o Flooding)

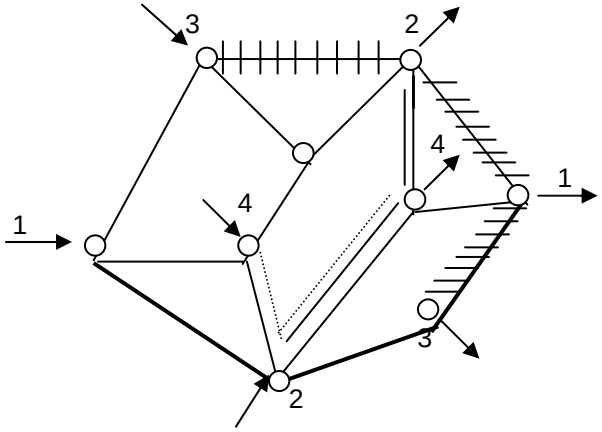
- **1)DIFFUSIONE:**
  Ogni nodo ritrasmette ogni messaggio ricevuto su tutti i lati in uscita escludendo quello da cui è stato ricevuto.

- **Proprietà**
  - **Un messaggio viene propagato a tutti i nodi della rete.**
  - **Per implementarlo in modo corretto, va memorizzata l'identità (ID) dei messaggi già inoltrati per evitare continuo ricircolo (Numerazione ciclica)**
- **Svantaggi**
  - **Elevato traffico interno**
  - **T* grande**
  - **Realizzazione abbastanza complessa (dovuto alla memorizzazione degli ID)**
- **Vantaggi**
  - **Grande affidabilità: è garantito il pck giunga a destinazione, ammesso esista almeno un cammino tra sorgente e destinazione. Questa tecnica le prova tutte, le strade.**
- **Applicazioni**
  - **Rete con piccolo coefficiente di utilizzo e vincoli ristretti sulla affidabilità (esempio reti militari, pensate per funzionare anche in ambiente ostile)**
- **Varianti**
  - **Diffusione selettiva: ad es i messaggi che arrivano da un nodo a Nord li mando a Sud.**
  - **Spanning Tree: albero di diffusione**

Instradamento Fissato

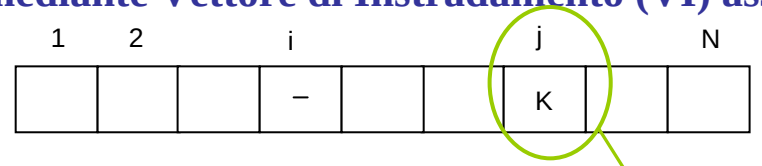
- **2)INSTRADAMENTO FISSATO** (Tecnica duale del Flooding)

  L'instradamento del pacchetto dipende solo dal nodo in cui si trova e dal nodo di destinazione



Esempio: 4 coppie sorgente-destinazione
Come faccio a specificare tale instradamento? Non memorizzo N^2 cammini! Ad ogni nodo assegno VI, che specifica il next hop. Ogni nodo conosce solo il next-hop.

- **Realizzazione mediante Vettore di Instradamento (VI) assegnato in ogni nodo**



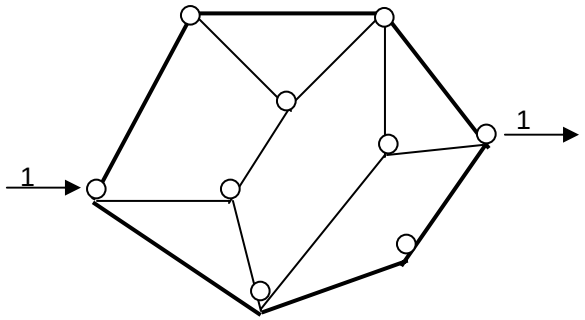
Nodo i

- **VI è determinato in ogni nodo dall'assegnamento ottimo dei flussi.**
- **Svantaggi:**
  - ✓ **Scarsa affidabilità (ho 1! cammino per ogni coppia di nodi)**
    - ✓ **Se *k* si guasta, qui, *j* è irraggiungibile per il nodo *i***
- **Vantaggi:**
  - ✓ **Semplicità di realizzazione**
  - ✓ **T* piccolo**
  - ✓ **I pacchetti non percorrono circuiti**

I pck con destinazione j vengono inviati al next-hop k

Instradamento Suddiviso

- **3) INSTRADAMENTO SUDDIVISO**
  E' pur sempre prefissato ma ad ogni nodo sono specificati più cammini per ogni destinazione (se esistono alternative, ovviamente).

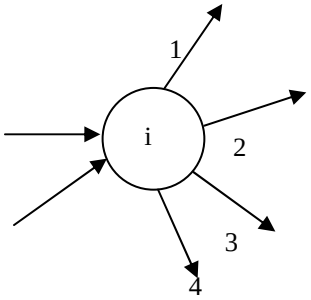


Esempio: 4 coppie sorgente-destinazione

- **Realizzazioni mediante la Matrice di Instradamento (MI) assegnata in ogni nodo (ha tante righe quanti sono i nodi della rete).**
- **$MI_i(j,\ell)$, percentuale di traffico entrante nel nodo *i* con destinazione nodo *j* instradato attraverso l'arco ℓ**

- **Le probabilità di scelta possono essere fissate in modo da ottenere in ogni lato il traffico ottimo (instradamento ottimo).**

Instradamento Suddiviso



Il nodo *i* ha 4 uscite
Il traffico per *j* lo instrado con p=0.3 in 1, p=0.2 in 2
…

Sono nel nodo *i*

	1	2	3	4
1 ⋮				
j	0.30	0.20	0.40	0.10
	…	…	…	…

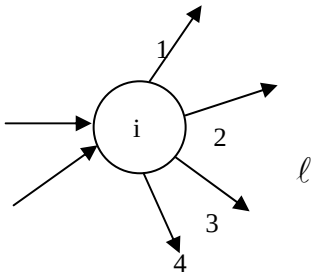
- **Svantaggi:**
  - **Scarsa affidabilità (maggiore che nell'instradamento fissato)**
  - **Alcuni flussi (TCP) mal tollerano riordino e segmenti out-of-order giunti a destinazione**
  - **Complessità maggiore (memorizzo una matrice)**
- **Vantaggi:**
  - **Miglior bilanciamento del traffico**
  - **T* piccolo**
  - **I pacchetti non percorrono circuiti**
- **Ipotesi (valide anche per la tecnica precedente):**
  - **Stazionarietà valori medi del traffico**
  - **Invarianza della topologia**

Se si rompe il lato 1, perdo ad es. il 30% del traffico verso *j*

Tecniche Dinamiche

- **Il Traffico cambia in modo statistico. La topologia anche.**
- **Si ipotizza che tutti i nodi conoscano (o abbiano stimato) una Tabella dei Ritardi**
- **Sulla base di tale misura/stima, adattano l'instradamento alle condizioni del traffico e della topologia.**
- **La Tabella dei Ritardi (T), indica, per ogni nodo *i*, qual è il tempo medio necessario a raggiungere ogni altro nodo passando per un determinato link uscente**

Al nodo i



	1	2	3	4
1	.2	.1	.5	.4
	:	:	:	:
j	.1	.3	.7	.5
	:	:	:	:

Ad es. in secondi

- **$T_i(j,\ell)$ = stima al nodo *i* del tempo (medio) necessario a raggiungere il nodo *j* utilizzando l'arco ℓ.**
- **Nota (stimata) la Tabella, ricavo un VI (stavolta dinamico)**

Tecniche Dinamiche

- **Come viene costruita la tabella dei ritardi ? Utilizzando:**
  1) **Tecniche locali (isolate): ogni nodo è autonomo ed esegue da solo le misure/stime**
  2) **Tecniche distribuite**
  3) **Tecniche centralizzate: esiste un nodo specializzato nella rete che esegue le misure/stime, e le comunica poi ai vari nodi**

Tecniche Locali

- **1)Coda in uscita più breve (Hot Potato)**

  T_i = vettore lunghezze code in uscita. Tiene basso il ritardo localmente ma i pacchetti viaggiano casualmente

- **Per migliorane le prestazione è utilizzabile unitamente all'instradamento prefissato:**
  - **-Si utilizza il lato scelto purché la coda non superi una soglia assegnata**
  - **-Si utilizza il lato con coda minore se non è troppo improbabile**

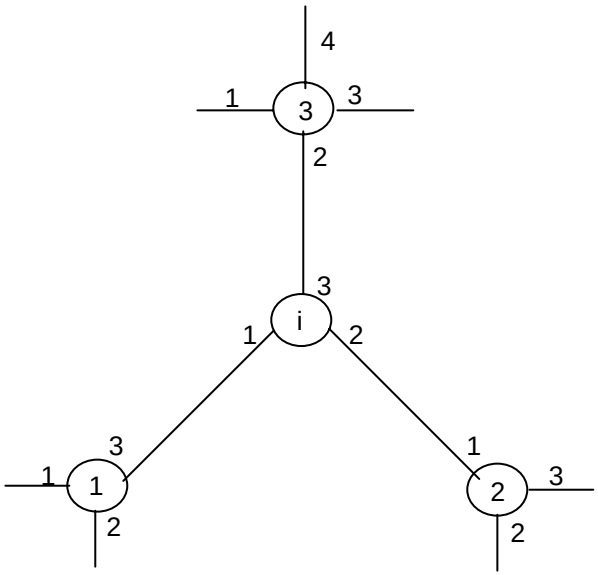
- **Stima locale del ritardo (Backward learning)**

  Aggiornamento di T_i all'arrivo di ogni pacchetto utilizzando l'informazione:
  - **TIN(j,ℓ) = tempo impiegato dal pacchetto originato in *j* per raggiungere *i* (numero di tratte percorse). Oppure mi baso su timestamp.**
- **Aggiornamento della stima corrispondente all'arco ℓ da cui il pacchetto è stato ricevuto (considero sia la vecchia stima che già ho che TIN, la nuova)**
  - $T_i\,(j,\ell)_n = K_1\,T_i\,(j,\ell)_v + K_2\,TIN(j,\ell)$

- **Svantaggi: Instabilità – instradamento su cammini non ottimi – Inefficienza**
- **Vantaggi: Basso costo di realizzazione (tutto è realizzato localmente, non ho bisogno di scambi di messaggi ad hoc)**

Tecniche Distribuite

- Per aggiornare T_i i nodi si scambiano informazioni
- Ogni nodo trasmette ai vicini il vettore VT_i (il vettore dei ritardi):
  $VT_i(j) = \min_{\ell} (T_i(j, \ell))$
- VT_i ha dimensione n (# nodi) e per ogni destinazione riporta il ritardo minimo che il nodo *i* osserva per quella destinazione (minimo rispetto alle alternative per raggiungere la destinazione).



Tecniche Distribuite

- **Quando il nodo *k* adiacente al nodo *i* riceve VT_i dal nodo *i*, aggiorna T_K come segue:**
  - $T_K(j, \ell) = VT_i(j) + Q_K(\ell) + T_T + T_P(k, \ell)$
  - **Ove:**
    - ℓ = **canale connettente k ad *i***
    - $Q_K(\ell)$ = **tempo medio di coda in** ℓ
    - T_T = **tempo di trasmissione in** ℓ
    - $T_P(k, \ell)$ = **tempo di propagazione in** ℓ

Tecniche Distribuite

TRASMISSIONE DI VT

- **Periodica: si trasmette VT in istanti prefissati (facile implementazione)**
- **Aperiodica: si trasmette VT quando si verificano cambiamenti nella rete. Permette di riconoscere i nodi sconnessi dalla rete.**

Esempi di tecniche distribuite

- Metriche e cifre di merito più utilizzate:
  - numero di nodi/hop da attraversare
  - tempo di transito nei nodi
  - larghezza di banda dei link
  - massima lunghezza pacchetti
  - costi dei link
  - Reliability/availability
- Algoritmi di instradamento
  - Distance-Vector (Bellman-Ford)
  - Link State/Shortest Path First (Dijkstra)

Instradamento Distance-Vector

- I nodi di rete cooperano alla determinazione dei cammini minimi realizzando l'algoritmo di Bellman-Ford distribuito.
- I nodi mantengono nella tabella di instradamento la stima di minor costo per ogni destinazione.
- Instradamento tratta per tratta (hop-by-hop).
- Per accelerare il riconoscimento di nodi sconnessi:
  - limite alla stima del ritardo massimo
  - time-out alla durata delle stime
  - non si invia stima al nodo vicino se esso è quello che ha comunicato la stima minore (split horizon).

Instradamento Link-State

- Costruisce in ogni nodo di rete una base dati contenente informazioni su:
  - topologia di rete
  - costi su cui effettuare la scelta dell'instradamento
- La distribuzione di tali informazioni avviene mediante un algoritmo di flooding.
- Maggiore informazione da scambiare, ma, avendo convergenza più veloce, la frequenza di aggiornamento può essere ridotta.
- Permette instradamento sia hop-by-hop sia da sorgente (source-routing).

Instradamento Link-State

- Le informazioni trasmesse da ogni nodo (Link State Advertisement) contengono:
  - identità del nodo
  - lista dei collegamenti incidenti
  - costi per l'uso dei collegamenti
  - identità dei segmenti di rete
  - indicazione delle applicazioni di rete ospitate nel nodo.
- Osservazioni
  - DV minor complessità rispetto a LS
  - LS velocità di convergenza superiore
  - Tempi di convergenza:
    - ✓ DV : n x K — n = numero dei nodi; K= numero di sottoreti
    - ✓ LS : $n \log n$ — non dipende dal numero di sottoreti

Richiami sui Grafi

- **Digrafo $G(N,A)$**
  - **N nodi**
  - **$A=\{(i,j),\ i \in N,\ j \in N\}$ archi (coppia ordinata di nodi)**
- **percorso: $(n_1, n_2, \ldots, n_l)$ insieme di nodi con $(n_i, n_{i+1}) \in A$**
- **cammino: percorso senza nodi ripetuti**
- **ciclo: percorso con $n_1 = n_l$**
- **digrafo connesso: per ogni coppia *i* e *j* esiste almeno un cammino da *i* a *j***
- **digrafo pesato: d_{ij} peso associato all'arco $(i,j) \in A$**
- **lunghezza di un cammino $(n_1, n_2, \ldots, n_l)$:**

$$d_{n1,\,n2} + d_{n2,n3} + \ldots + d_{n(l-1),\,nl}$$

Problema del cammino minimo

Dato un digrafo *G(N,A)* e due nodi *i* e *j*, trovare il cammino di lunghezza minima tra tutti quelli che consentono di andare da *i* a *j*

- **il problema è di complessità polinomiale**

Proprietà:

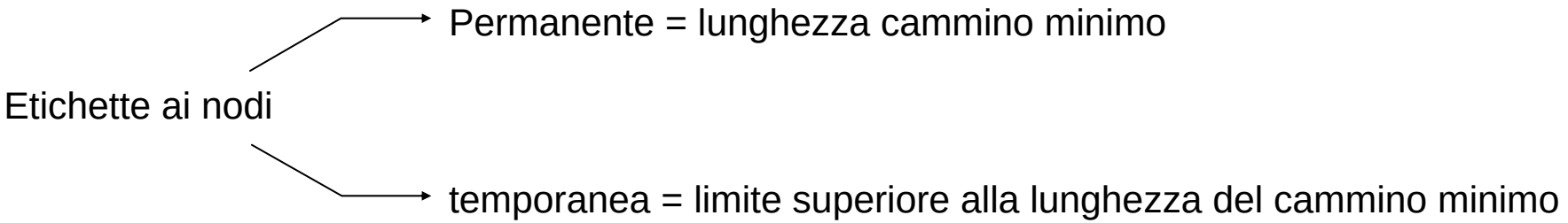
se il nodo k è attraversato dal cammino minimo da i a j, il sotto-cammino fino a k è anch'esso minimo

Algoritmi per Cammini Minimi

- **Da un nodo a tutti gli altri**
  - **Archi non negativi:**
    - ✓ **Dijkstra (1959)** $0(n^2)$
  - **Archi negativi, Cicli non negativi**
    - ✓ **Ford (1956)**
    - ✓ **Moore (1957)**
    - ✓ **Bellman (1958)** $0(n^3)$
    - ✓ **Yen (1970), minor occupazione di memoria**
- **Fra tutte le coppie di nodi**
  - ✓ **Ripetere n volte gli algoritmi precedenti** $0(n^4)$
  - ✓ **Floyd (1962)**
  - ✓ **Warshall (1962)** $0(n^3)$

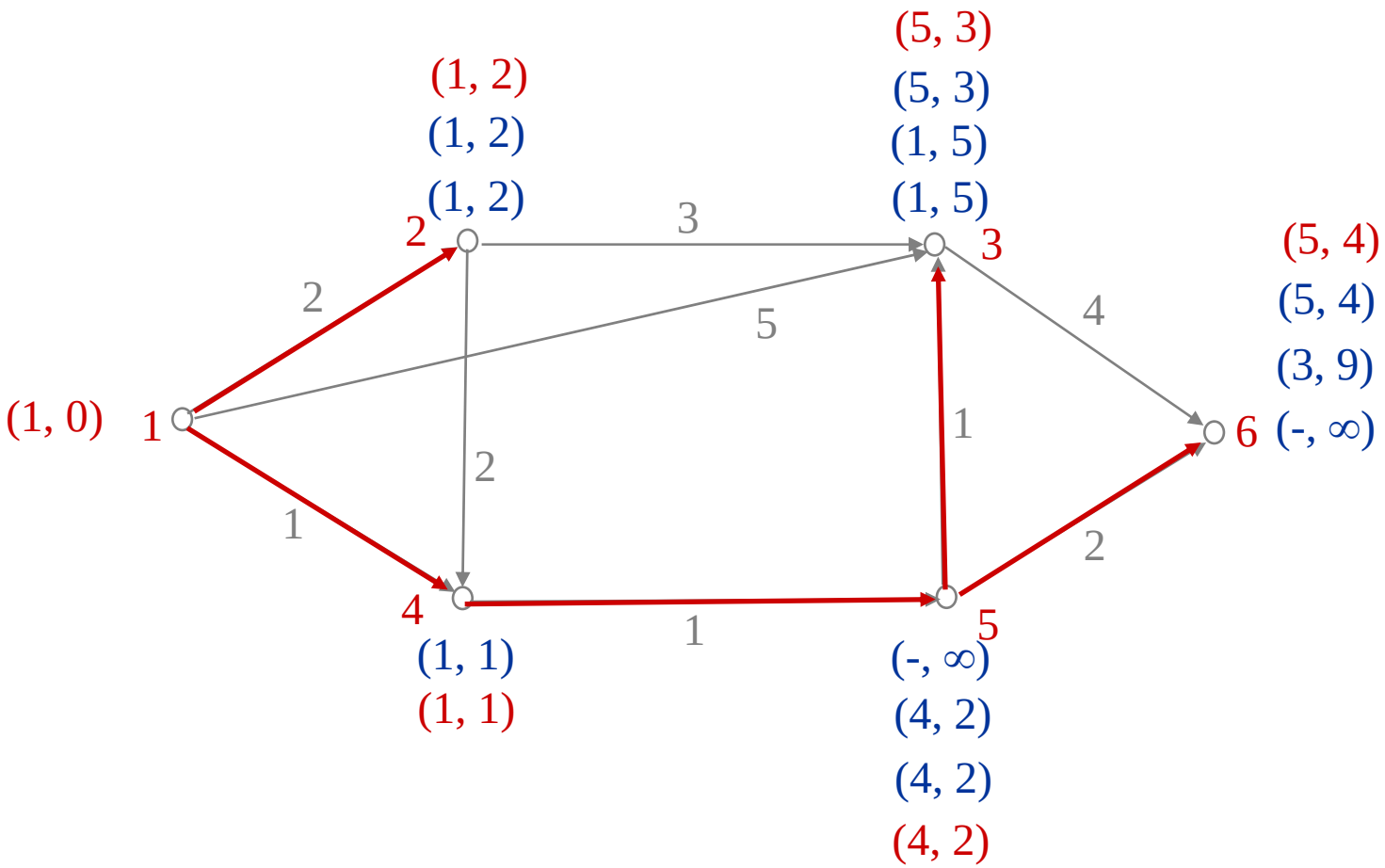
Algoritmo di Dijkstra

- **Ipotesi: Lunghezza archi positiva**
- **Metodo di Dijkstra:**



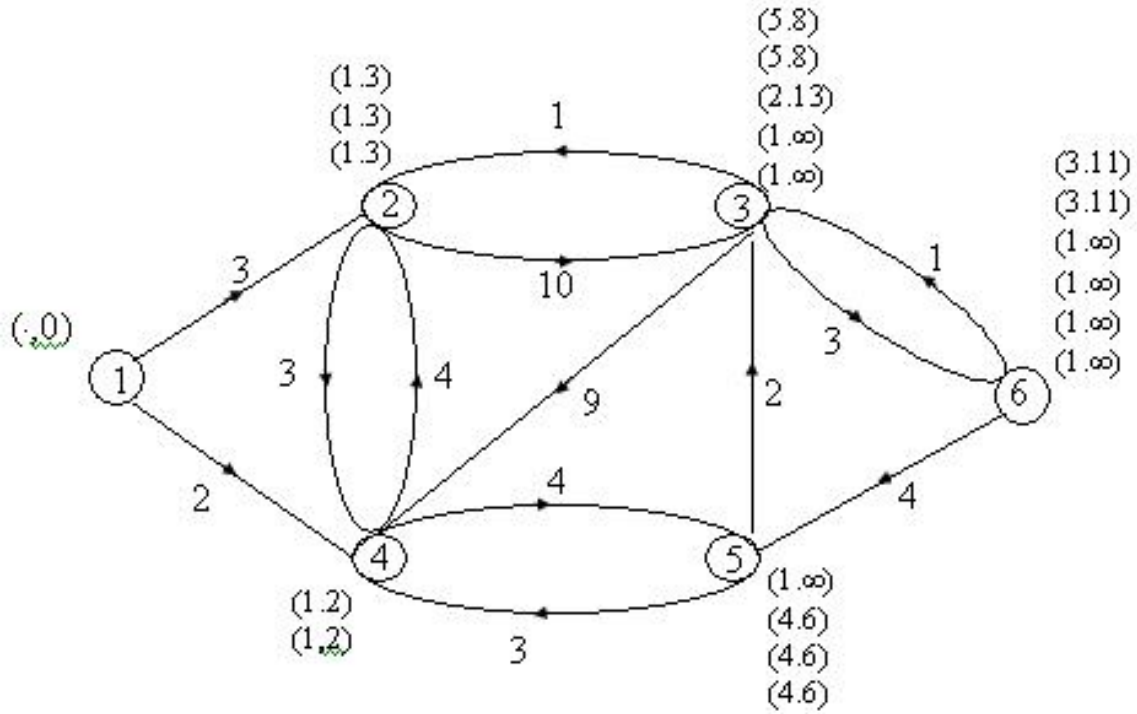
1. **Inizializzazione:**
   1. **Etichetta permanente nodo 1 = 0 : $u_1 = 0$**
   2. **Etichetta temporanea al nodo $i = l_{1i}$ $\forall i$ $u_i = l_{1i}$**
   3. **Si assume $l_{1i} = \infty$ se l'arco tra i e j non esiste**
2. **Trovare nodo k con etichetta temporanea minima**
   Dichiarare etichetta di k permanente
   Aggiornare etichette temporanee
   $u_j = \min (u_j , u_k + a_{kj})$ $\forall i$ con etichetta temporanea
3. **Se tutte le etichette sono permanenti STOP**

Esempio: Dijkstra



Esempio 2: Dijkstra

- **L'algoritmo etichetta in modo permanente i nodi in ordine crescente di distanza dal nodo 1**



Algoritmo di Dijkstra

algorithm *Dijkstra*
begin
$S := \phi$; $\overline{S} = N$;
d(i) : ∞ for each node i $\in N$;
d(s) : = 0 and pred(s) : = 0
while |S| $< n$ **do**
begin
let $i \in \overline{S}$ be a node for which d(i) = min{d(j) : $j \in \overline{S}$}
$S := S \cup \{i\}$
$\overline{S} := \overline{S}$ - {i}
for each (i, j) $\in$ A(i) **do**
if d(j) > d(i) + c_{ij} **then** d(j) : = d(i) + c_{ij} and pred(j) : = i
end;
end;

Algoritmo di Dijkstra: Commenti

- Fornisce i cammini minimi dal nodo 1 a tutti gli altri nodi
- Se interessa solo il cammino minimo fra nodo 1 e nodo N, l'algoritmo si ferma quando si etichetta in modo permanente il nodo N
- Algoritmo di Dijkstra inverso
  - ✓ Se interessano i cammini minimi da tutti i nodi al nodo N: si etichettano i nodi a ritroso
- Algoritmo di Dijkstra bidirezionale.
  - ✓ Se interessa il cammino fra 1 e N: si applicano contemporaneamente Dijkstra diretto e inverso. L'algoritmo si ferma quando lo stesso nodo è stato etichettato da entrambi gli algoritmi.
  - ✓ Risulta più veloce in quanto mediamente si etichettano meno nodi.
- La complessità dell'algoritmo di Dijkstra è $0(n^2)$.
  - ✓ L'algoritmo esegue n volte la selezione del nodo da etichettare permanentemente. Ad ogni selezione deve scandire tutti i nodi etichettati temporaneamente quindi: $n + (n-1) + (n-2) + ...+ 1 = 0(n^2)$

Algoritmo di Bellman-Ford

- **Ipotesi:**
  - **Lunghezza archi sia positivi che negativi**
  - **non esiste alcun ciclo di lunghezza negativa**
- **Scopo:**
  - **trovare i cammini minimi tra un nodo (sorgente) e tutti gli altri nodi, oppure**
  - **trovare i cammini minimi da tutti i nodi ad un nodo (destinazione)**

Algoritmo di Bellman-Ford

- **Variabili aggiornate nelle iterazioni:**
  - **$D_i^{(h)}$ lunghezza del cammino minimo tra il nodo *1* (sorgente) e il nodo *i* composto da un numero di archi $\leq h$**
- **Valori iniziali:**

$$D_1^{(h)} = 0 \qquad \forall h$$

$$D_i^{(0)} = \infty \qquad \forall i \neq 1$$

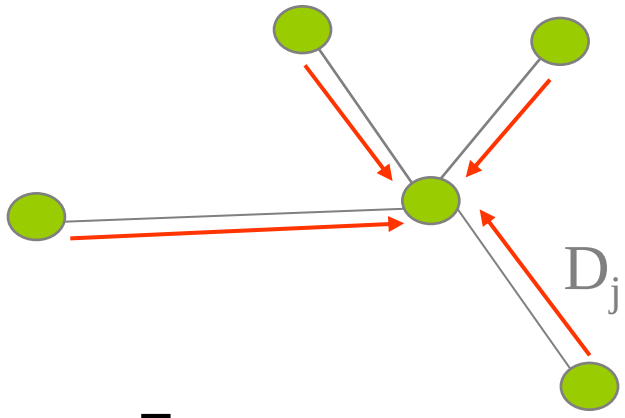
- **Iterazioni:**

$$D_i^{(h+1)} = \min\left[D_i^{(h)}, \min_j \left(D_j^{(h)} + d_{ji} \right) \right]$$

- **l'algoritmo termina in N-1 passi**

Algoritmo di Bellman-Ford in forma distribuita

- Si dimostra che l'algoritmo converge in un numero finito di passi anche nel caso in cui viene implementato in modo distribuito
- Periodicamente i nodi inviano l'ultima stima del cammino minimo ai vicini e aggiornano la propria stima secondo il criterio delle iterazioni

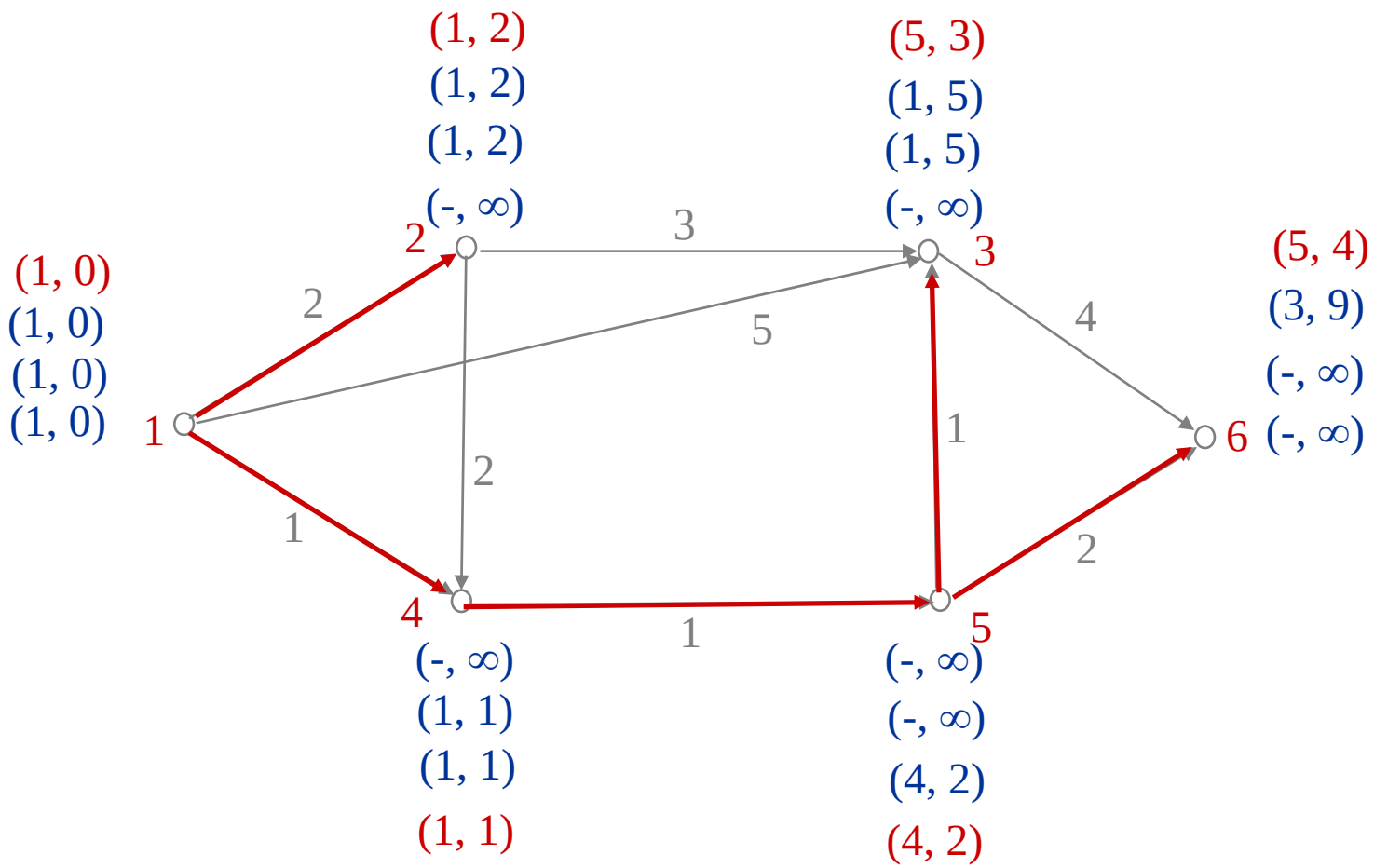


$$D_i := \min\left[D_i , \min_j \left(D_j + d_{ji} \right) \right]$$

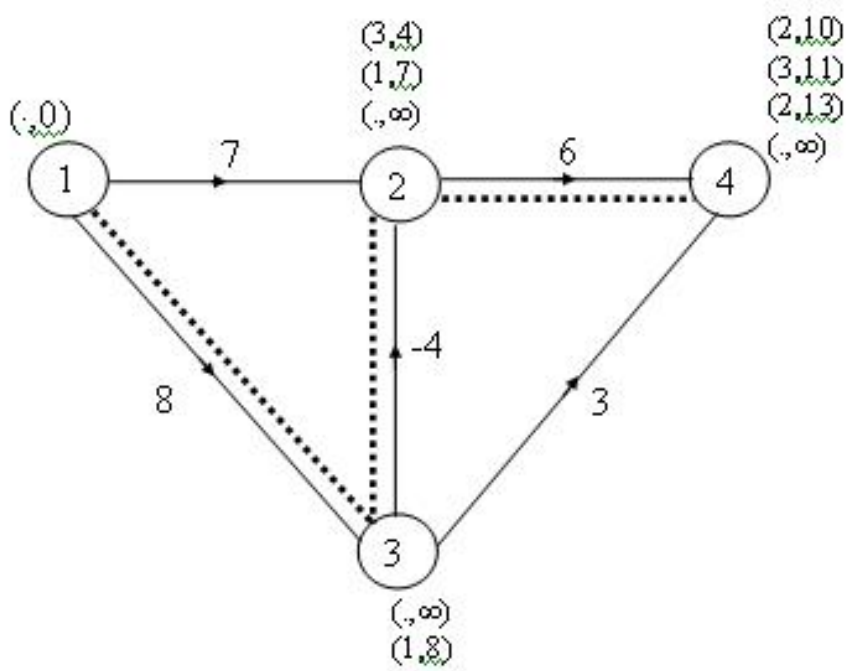
Algoritmo di Bellman-Ford in pratica

- **Per poter applicare praticamente l'algoritmo si può procedere in questo modo:**
- **Si usano delle etichette per i nodi *(n, L)* dove *n* indica il primo nodo sul cammino minimo ed *L* la sua lunghezza**
- **Le etichette vengono aggiornate guardando le etichette dei vicini (l'ordine non conta grazie alla proprietà dell'algoritmo distribuito)**
- **Quando le etichette non cambiano più si ricostruisce l'albero dei cammini minimi ripercorrendo le etichette**

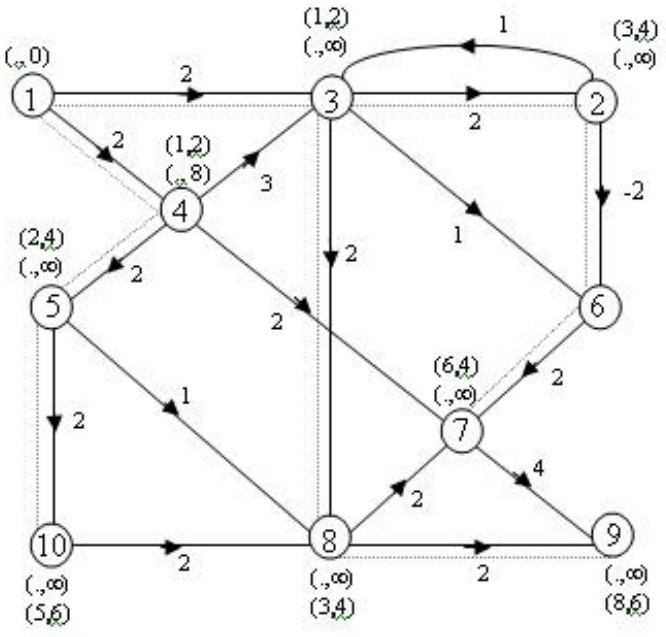
Esempio: Bellman-Ford



Esempio 2: Bellman-Ford



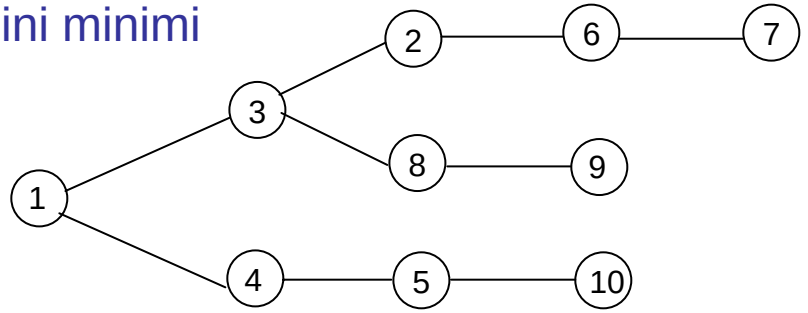
Esempio 2: Bellman-Ford



L'algoritmo trova contemporaneamente i cammini minimi dalla sorgente *s* a tutti i nodi.

Possono esistere più cammini minimi fra due nodi

Si viene a costruire l'albero dei cammini minimi



Algoritmi: complessità

- L'algoritmo di Bellman-Ford ha una complessità:
  - N-1 iterazioni
  - N-1 nodi per iterazione
  - N-1 confronti per nodo
    - ✓ Complessità: $O(N^3)$
- L'algoritmo di Dijkstra ha una complessità:
  - N-1 iterazioni
  - in media N operazioni per iterazioni
    - ✓ Complessità: $O(N^2)$
- L'algoritmo di Dijkstra è in generale più conveniente

Tecniche Centralizzate

- **Calcola vettori instradamento per tutti i nodi, che eseguono un instradamento fisso sulla base dei vettori VT ricevuti ed aggiornati da un calcolatore centralizzato.**

- **Centralizzate vs. Distribuite: - Vantaggi**

  Centralizzate
  - **Calcolo più semplice**
  - **I nodi non hanno overhead di calcolo**
  - **L'algoritmo può essere più sofisticato**
  - **Possono essere evitati i "loops"**

  Distribuite
  - **Maggiore affidabilità**
  - **Applicabile a grandi reti**

Tecniche Centralizzate

- **Centralizzate vs. Distribuite: - *Svantaggi***

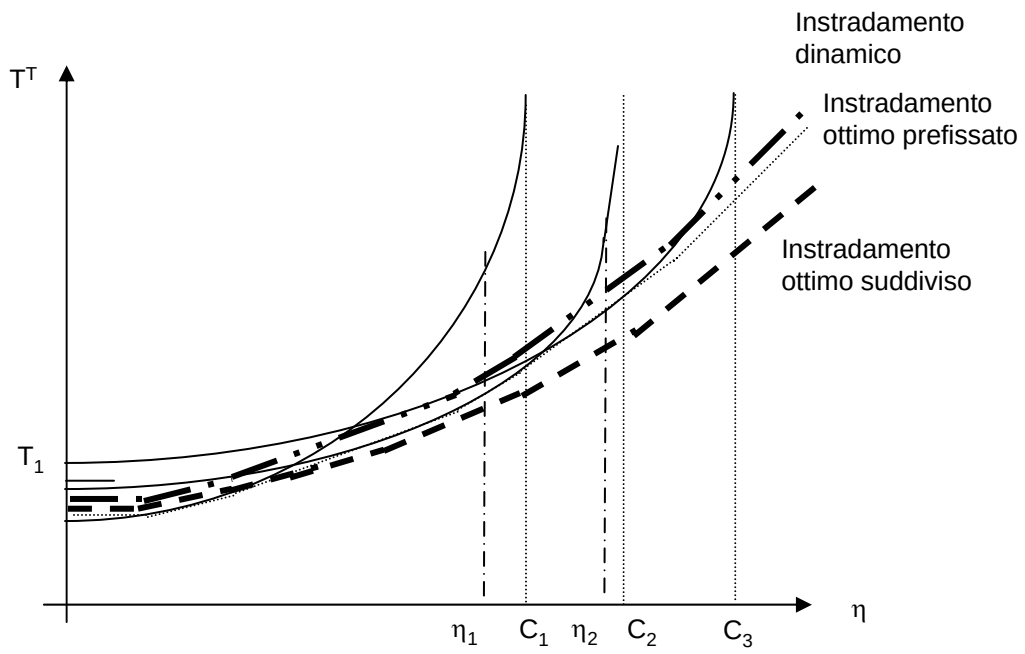
  Centralizzate
  - **Non applicabile a grandi reti**
  - **Quantità di calcolo nel calc. centralizzato molto pesante (N^3 invece di N^2)**
  - **Calcolo su informazione vecchia**
  - **Ritardo dell'applicazione nuove decisioni**
  - **Possono esistere "loops" dovuto ai ritardi di propagazione**
  - **Scarsa affidabilità:**
  - **Duplicazione calc. centralizzato**
  - **Pericolo di isolamento congestione traffico di instradamento**
  - **Non si conosce il calc. centralizzato in funzione**
  - **La rottura di un lato o di un nodo può essere critica**

  Distribuite
  - **Reagisce lentamente alle cattive notizie**
  - **Maggior tempo di adattamento**
  - **Tutti i nodi devono calcolare in modo corretto.**
  - **(L'errore di uno si propaga)**

Criteri di Valutazione Algoritmi Dinamici

- Velocità di risposta
  - Tempo di convergenza ad un nuovo stato a causa di cambiamenti deve essere minore dell'intervallo fra le variazioni
- Overhead di informazione
  - Numero di pacchetti di controllo
  - Dimensione pacchetti di controllo
- Complessità di calcolo
  - Tempo di elaborazione sui nodi
- Occupazione di memoria
  - Memorizzazione e aggiornamento tabelle
- Assenza di cicli
  - In transitorio e a regime

Criteri di Valutazione Algoritmi Dinamici



Instradamento Multicast/Broadcast

- **Un pacchetto deve essere inviato a tutti i nodi della rete (Broadcast) oppure ad un gruppo di essi (Multicast).**

INSTRADAMENTO SEPARATO

- **Il nodo sorgente trasmette n volte lo stesso pacchetto con indirizzi diversi. Non richiede delle tecniche di instradamento con destinazione unica.**
- **Spreco di banda.**

INSTRADAMENTO A DIFFUSIONE (FLOODING)

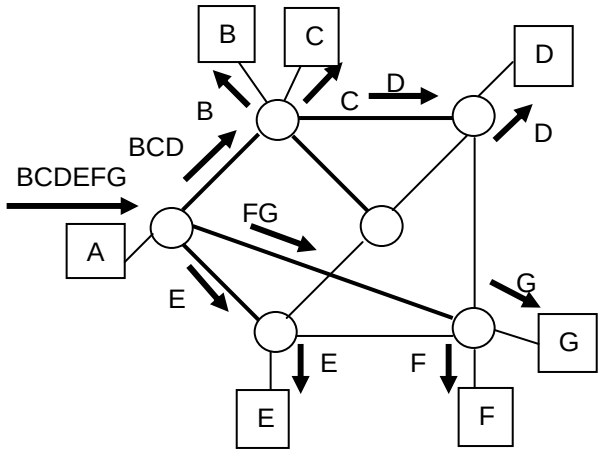
- **Spreco di banda. Robusto.**

Instradamento Multicast/Broadcast

INSTRADAMENTO CON MULTIDESTINAZIONE

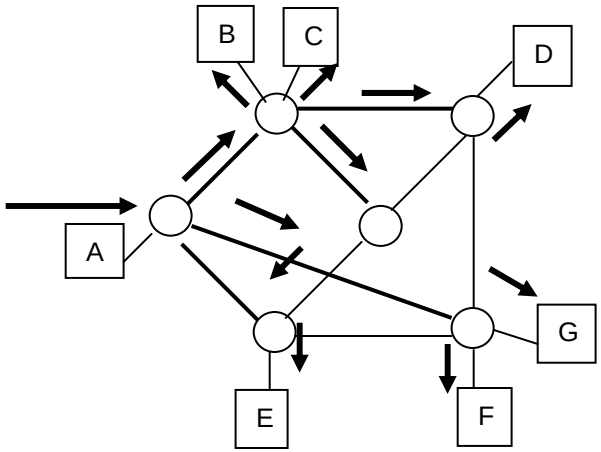
- **Il pacchetto contiene indicazione delle destinazioni.**
- **Ogni nodo ritrasmette il pacchetto su tutte le linee in uscita necessarie per raggiungere le destinazioni indicate.**
- **Il pacchetto trasmesso in una linea contiene solo le destinazioni raggiungibili attraverso di essa.**
- **Il numero delle destinazioni si riduce ad ogni ritrasmissione**
- **Una sola copia viene trasmessa per più destinazioni che seguono la stessa strada.**
- **Efficiente uso della banda**
- **Richiede modifica al formato del pacchetto, elaborazione nei nodi ma usa la stessa informazione dell'instradamento singolo.**

Esempio: Instradamento Broadcast



Instradamento sullo Spanning Tree

- **Determinazione dello spanning tree memorizzando nei nodi i lati in uscita che appartengono a ST. Ogni nodo alla ricezione di un pacchetto broadcast ne invia copia sui lati in uscita appartenenti a ST e alle stazioni locali.**
- **Un pacchetto broadcast è identificato da un indirizzo, (es. tutti 1).**
- **Efficienza di banda del multidestinazione. Non richiede modifiche al formato del pacchetto.**
- **Necessita informazione addizionale nei nodi (memorizzazione di ST).**



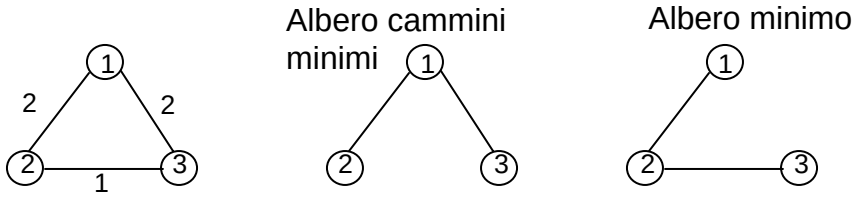
Determinazione Albero Minimo

- Ipotizziamo grafi non diretti con lunghezza archi anche negative.
- Se il grafo ha archi diretti si parla del sottografo con cammini diretti uscenti da un nodo verso tutti gli altri (routed arborescence). Problema estremamente complicato.
- Lunghezza sottografo G*i* $\ell(Gi) = \sum_{[i,j] \in Gi} \ell[i, j]$

- Trovare sottografo albero che contenga tutti i nodi e di lunghezza minima.

Determinazione Albero Minimo

- **Ovviamente, in generale, l'albero minimo è *diverso* dall'albero dei cammini minimi.**



- **Se $\ell[i, j] \geq 0$ l'albero minimo è il sottografo più corto contenente tutti i nodi.**

Algoritmi per la Determinazione Albero Minimo

- **Diverse realizzazioni della procedura "greedy".**
- **Ad ogni passo si aggiunge un arco di costo minimo scelto dalla lista degli archi possibili (archi che non introducono cicli).**

- **Kruskal**

  genera sotto alberi in modo irregolare partendo dai lati più corti
- **Prim**

  partendo da un'unica radice fa crescere il sottoalbero fino a diventare spanning tree
- **Sollin**

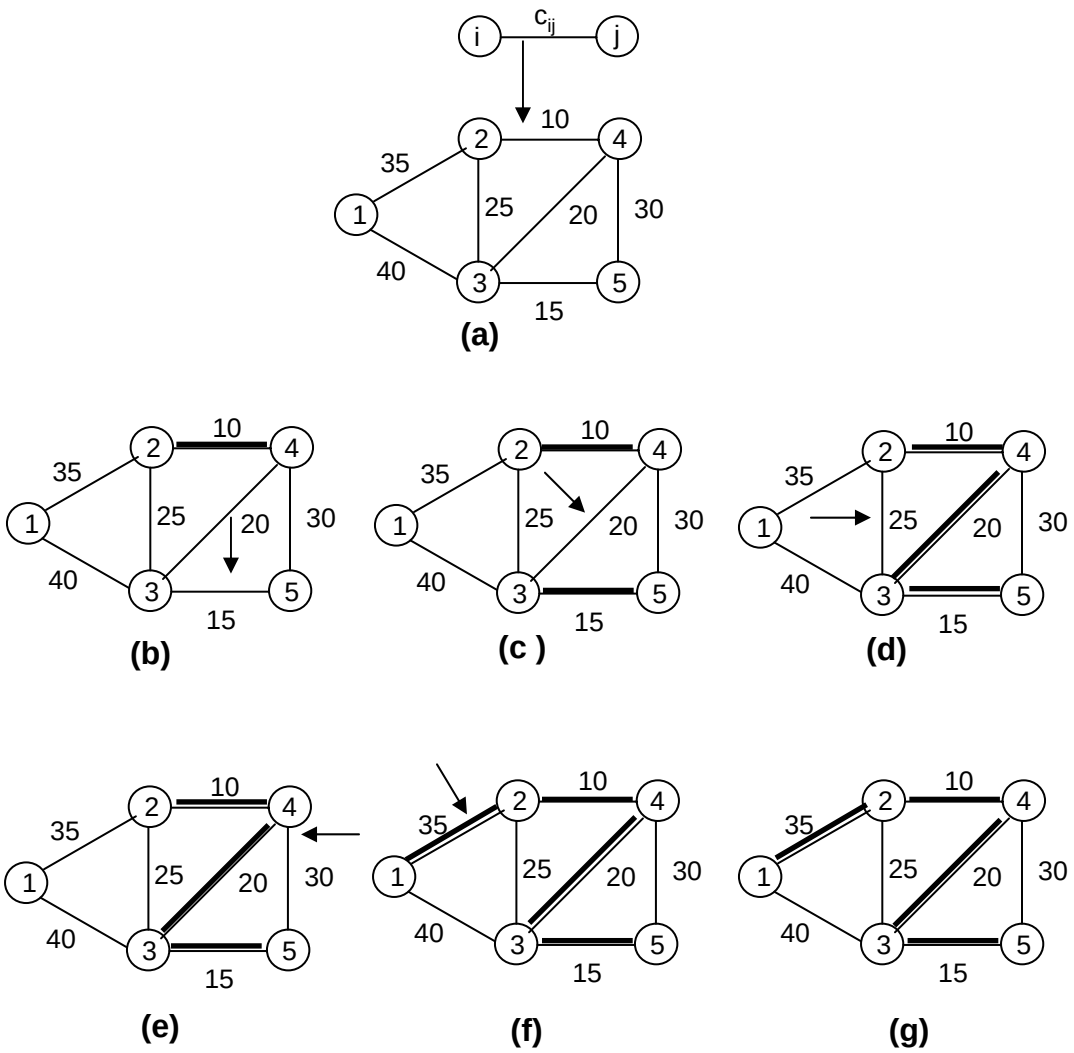
  genera più sottoalberi come Kruskal ma ad ogni passo connette un nodo ad ogni sotto albero, analogamente a Prim.

- **Albero di costo massimo: stessa procedura dopo aver moltiplicato per -1 tutte le lunghezze dei lati**

Algoritmo di Kruskal

1. Ordinare archi in ordine non decrescente di costo.
2. Definire un insieme, LIST, di archi che appartengono all'albero minimo.
3. Inizialmente LIST è vuoto.
4. Esaminare in modo ordinato gli archi ed inserire in LIST quelli la cui aggiunta non genera cicli con gli archi già in LIST.

Algoritmo di Kruskal



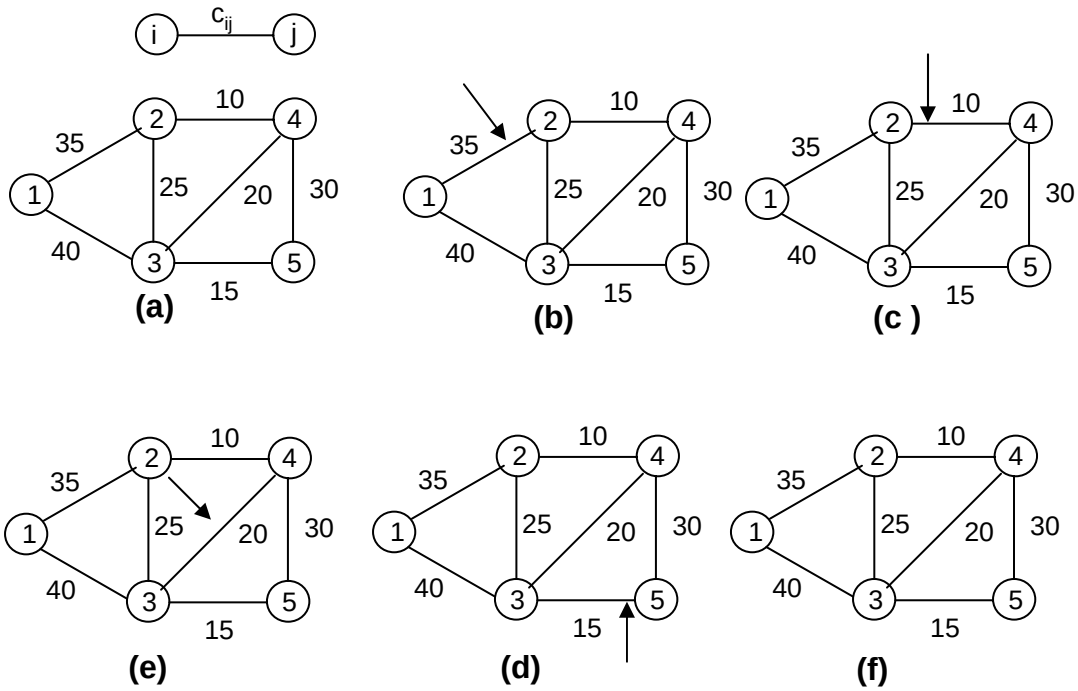
Algoritmo di Prim

1. **I nodi della rete sono divisi in due insiemi *S* e $\overline{S}$**
2. **Inizialmente *S* = {1}**
3. **Esaminare il taglio [*S* e $\overline{S}$] e selezionare l'arco più corto = (i,j).**
4. **Aggiornare gli insiemi**

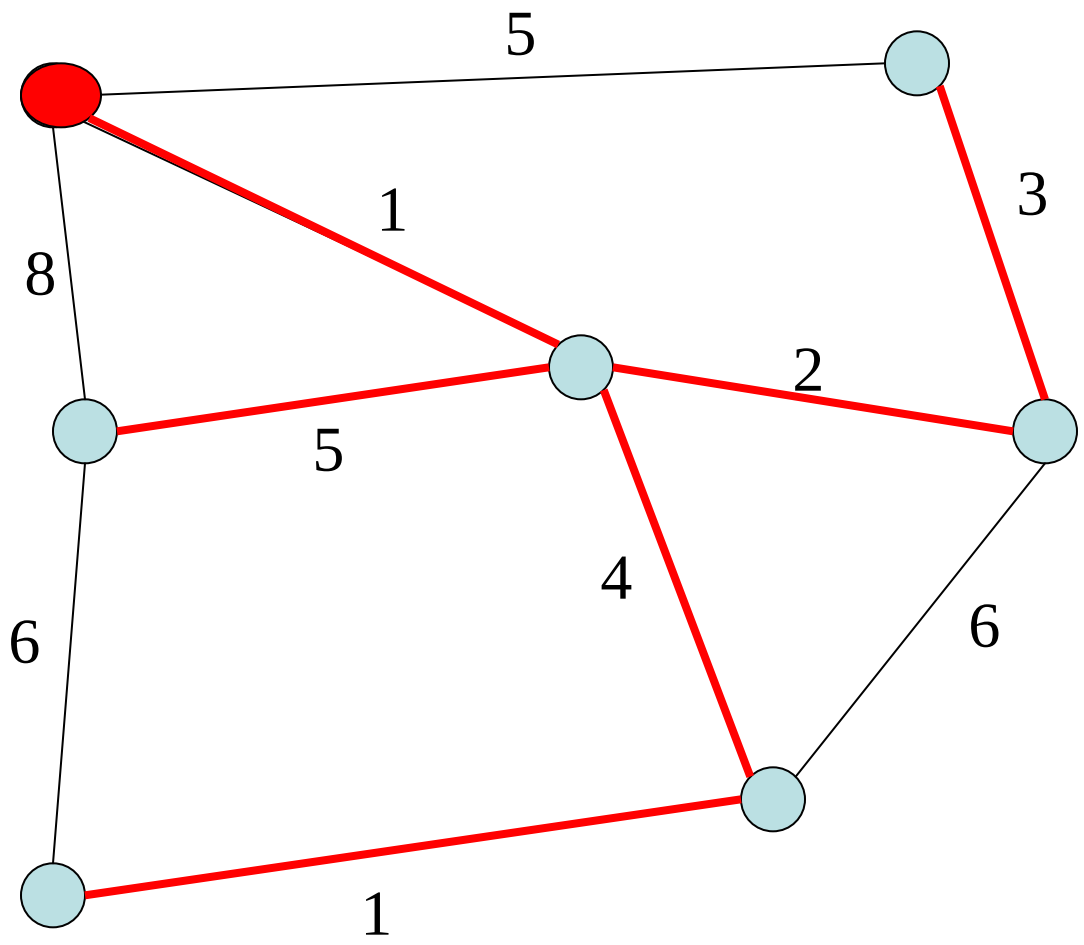
$$S = S + j \qquad \overline{S} = \overline{S} - j$$

Se S contiene tutti i nodi gli archi selezionati costituiscono l'albero minimo.

Algoritmo di Prim



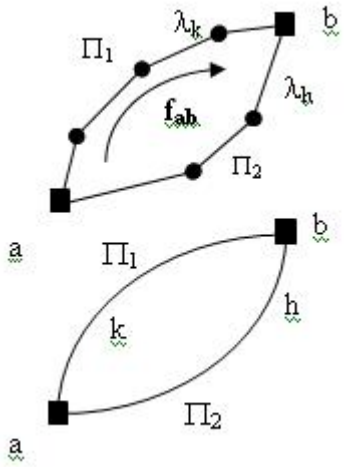
Esempio 2:Algoritmo di Prim



Routing Ottimo

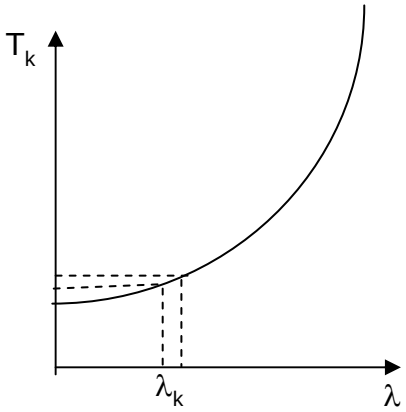
- Viene anche definito “assegnamento ottimo dei flussi”
- Soluzioni proposte in letteratura:
  - Metodi di programmazione convessa (inefficienti)
  - Metodo della Deviazione di Flusso

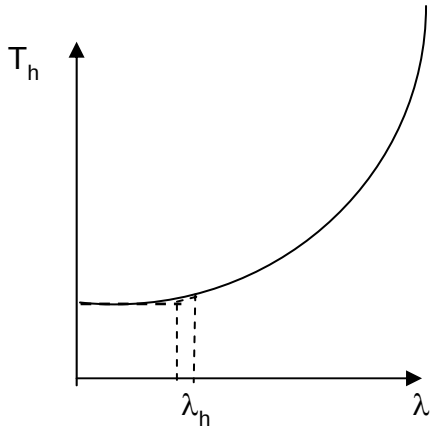
Metodo della Deviazione di Flusso
Flow Deviation Method



$$\lambda'_k = \lambda_k + f_{ab} \qquad k \in \Pi_1$$

$$\lambda'_h = \lambda_h - f_{ab} \qquad h \in \Pi_2$$





Quando la pendenza dei 2 cammini diventa la stessa, non ho più convenienza a spostare traffico da un path all'altro.

Algoritmo di Deviazione di Flusso

- Impostare la lunghezza del lato: $\ell_i = \frac{\partial T_i}{\partial \lambda_i}$
- Calcolare i cammini più corti fra ogni coppia (i,j)
- Instradare f_{ij} lungo il cammino più corto

- Note:
  - Per T_i può venir usata l'approssimazione per cui ogni lato è considerato come una coda M|M|1
  - Quindi: $\ell_i = \frac{\partial}{\partial \lambda_i}\left(\frac{1}{\mu_i - \lambda_i}\right) = \frac{1}{(\mu_i - \lambda_i)^2}$
  - È necessario utilizzare deviazioni piccole di traffico
  - Altrimenti posso verificarsi problemi di instabilità