

# Sviluppo di algoritmi per la determinazione di ipercammini minimi

Marco Gualtieri

Relatore: Prof. Fabio Schoen  
Correlatore: Ing. Mirko Maischberger

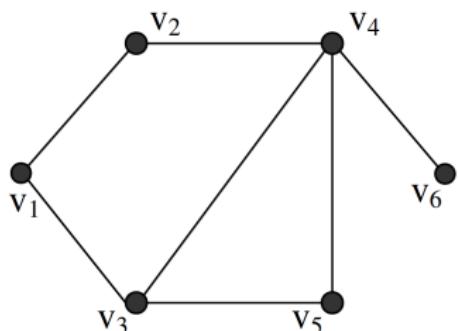
Luglio 2012

# Scopo della tesi

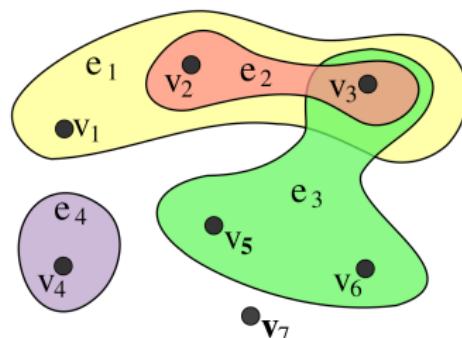
- Approfondimento e formalizzazione della teoria degli ipergrafi
- Modello del trasporto pubblico locale
- Studio del problema degli ipercammini minimi
- Implementazione degli algoritmi

# Ipergrafi non orientati

$$G = (V, E)$$



Grafo non orientato



Ipergrafo non orientato

$$E \subseteq \{ (v_i, v_j) \mid v_i, v_j \in V \}$$

$$E \subseteq \{ e_i \mid e_i \subseteq V \}$$

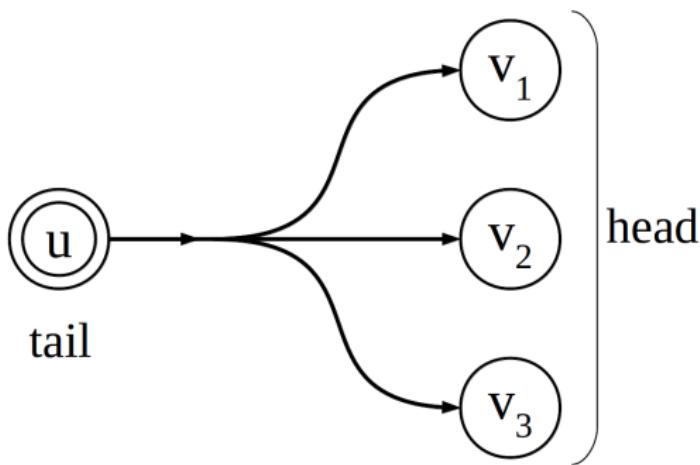
---

Grafo tradizionale: ipergrafo avente  $|e_i| = 2$ ,  $\forall e_i \in E$

---

# Iperarco orientato

$$e = [ t(e), h(e) ]$$

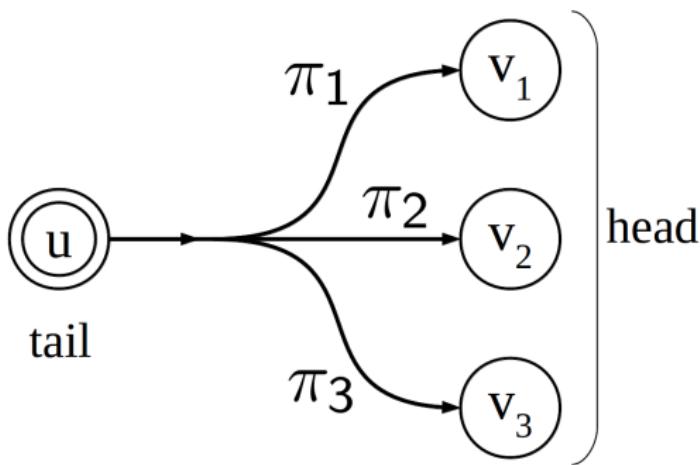


$$t(e) = u$$

$$h(e) = \{v_1, v_2, v_3\}$$

# Iperarco orientato

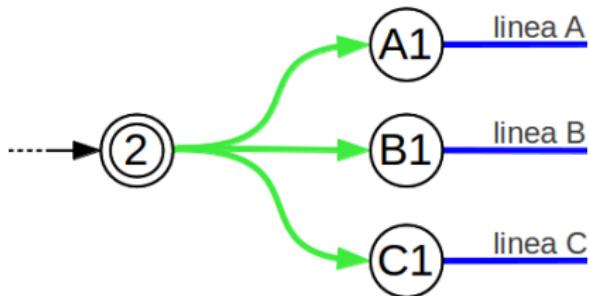
$$e = [t(e), h(e)]$$



$$\sum_i \pi_i = 1$$

# Nodi fermata

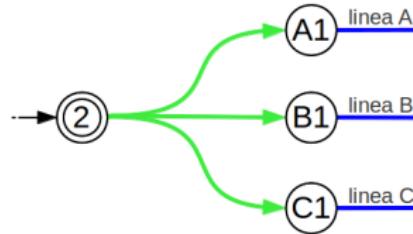
Ciascun utente sale sul primo mezzo *attrattivo* disponibile



**Evento aleatorio:** arrivo del primo veicolo.

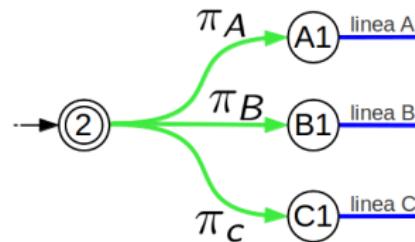
# Nodi fermata

- Frequenza oraria  $\varphi_i$



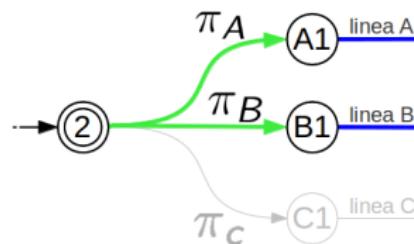
# Nodi fermata

- Frequenza oraria  $\varphi_i$
- Probabilità  $\pi_i$



# Nodi fermata

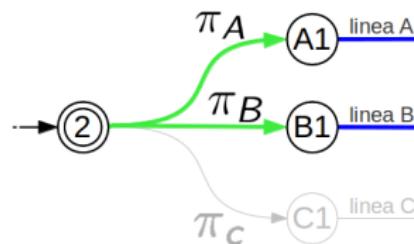
- Frequenza oraria  $\varphi_i$
- Probabilità  $\pi_i$
- Insieme attrattivo  $L$



$$L = \{A, B\}$$

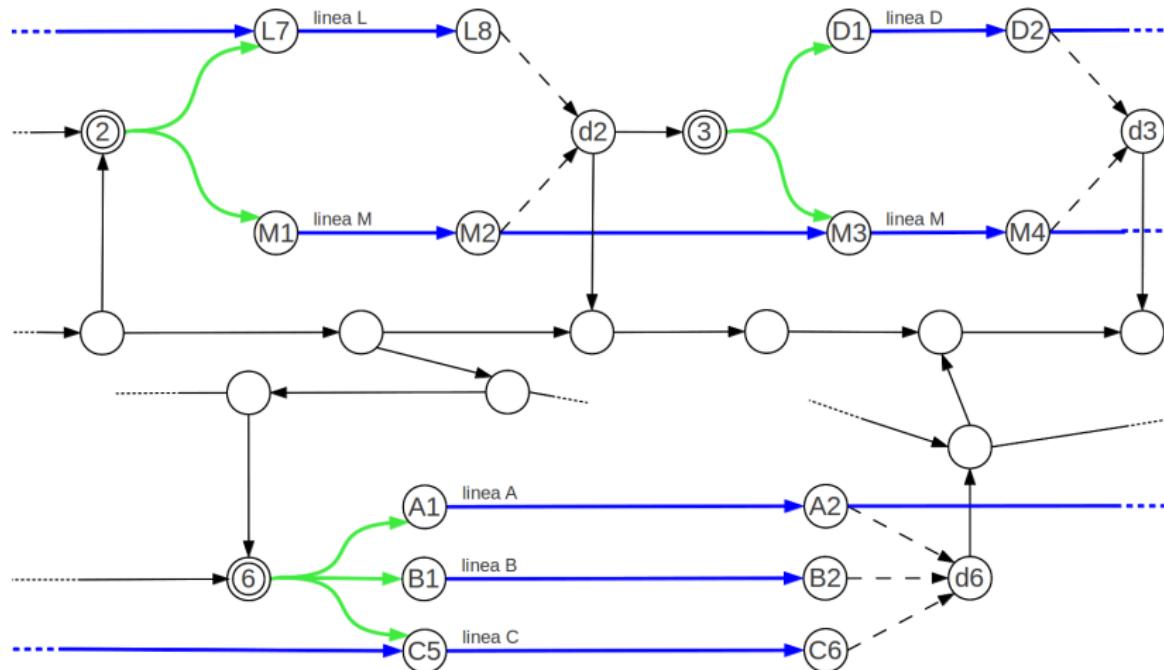
# Nodi fermata

- Frequenza oraria  $\varphi_i$
- Probabilità  $\pi_i$
- Insieme attrattivo  $L$
- Tempo medio di attesa  $w$

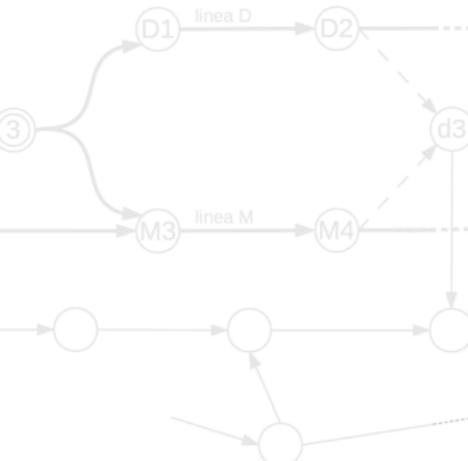
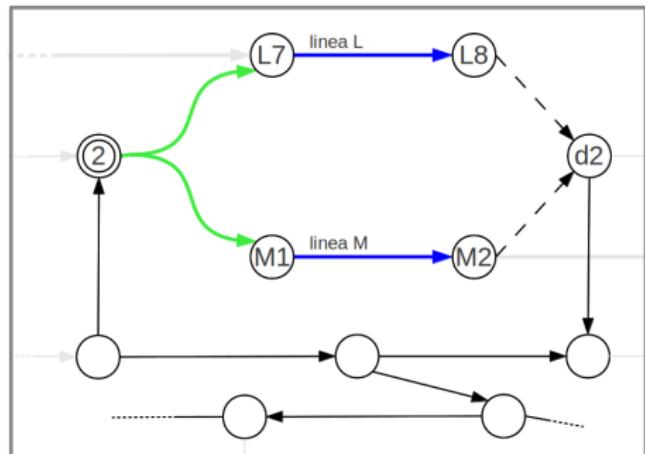


$$L = \{A, B\}$$

# Modello della rete



# Modello della rete

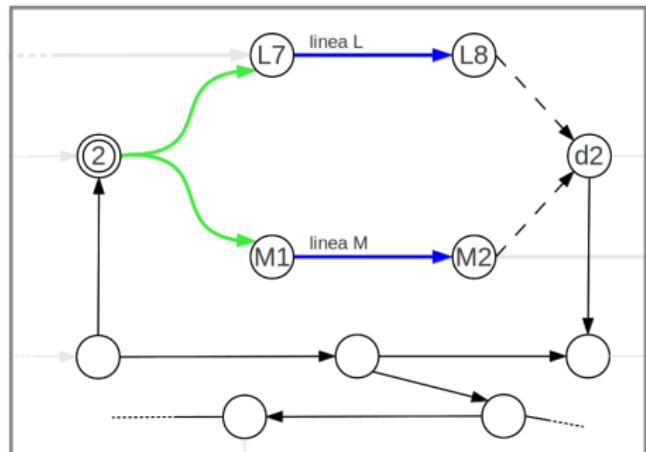


**Nodi fermata** (2)

**Iperarchi di salita**

A graphic element consisting of three green arrows pointing towards a central point, representing an incoming hyperedge or flow.

# Modello della rete



## Nodi “normali”

pedonale

L7

a bordo

d2

di discesa



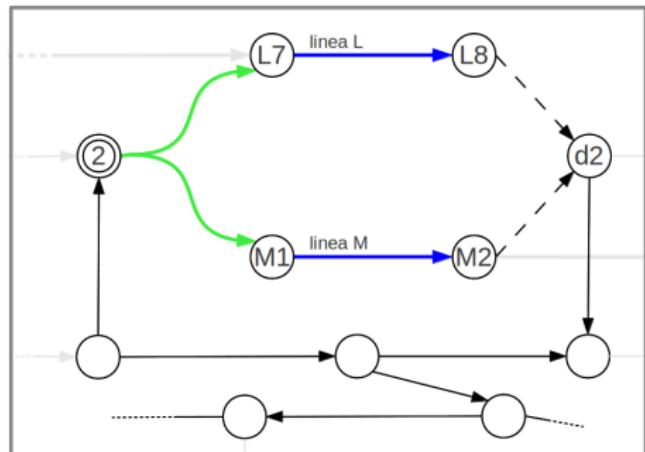
## Nodi fermata

②

## Iperarchi di salita



# Modello della rete



## Nodi “normali”

pedonale

L7

a bordo

d2

di discesa

## Archi di viaggio

pedonale

a bordo

di discesa

tempo di percorrenza

costo del biglietto /  
penalità

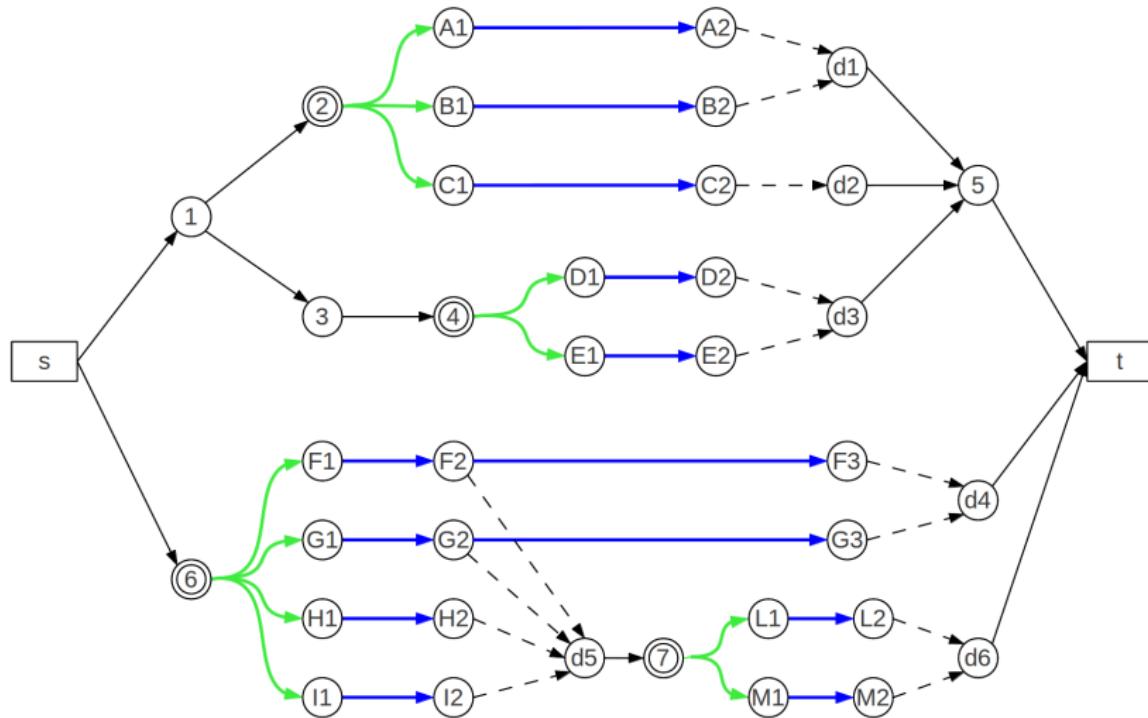
## Nodi fermata

(2)

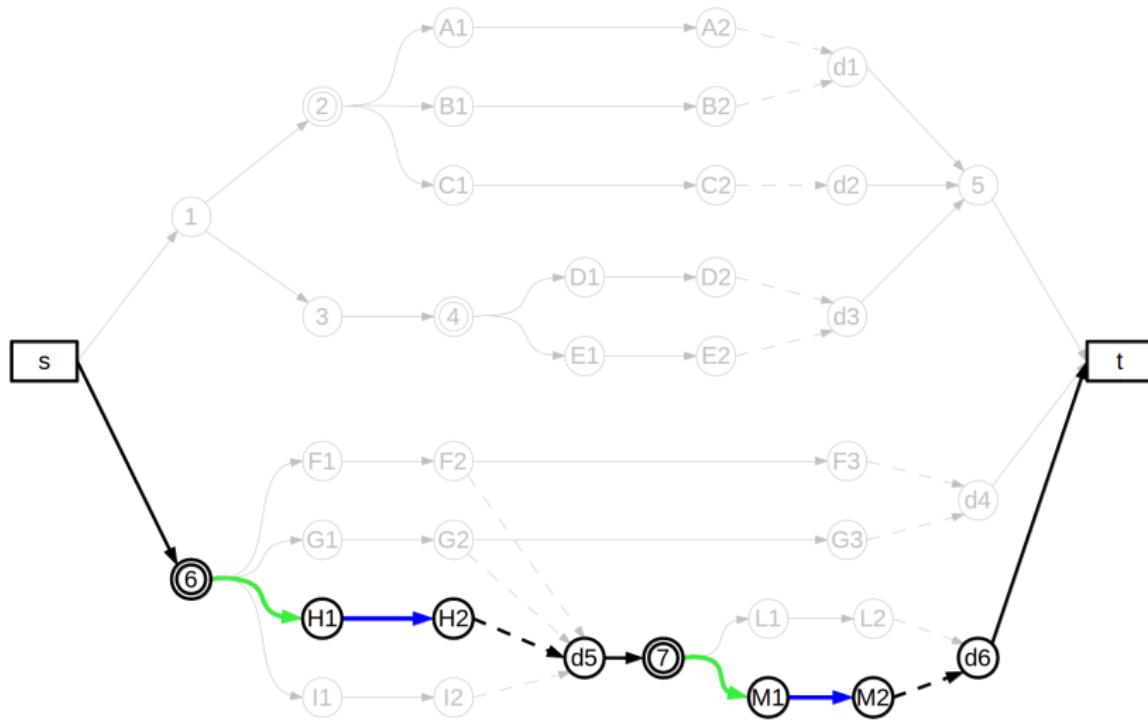
## Iperarchi di salita



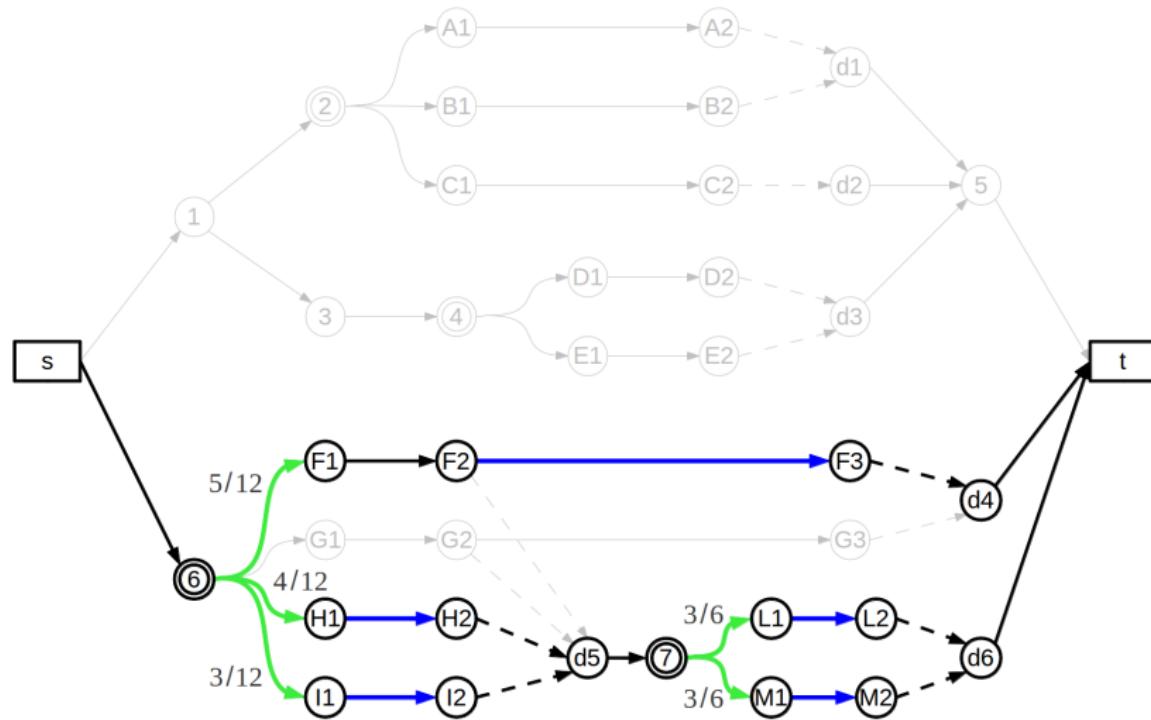
# Esempio di ipergrafo



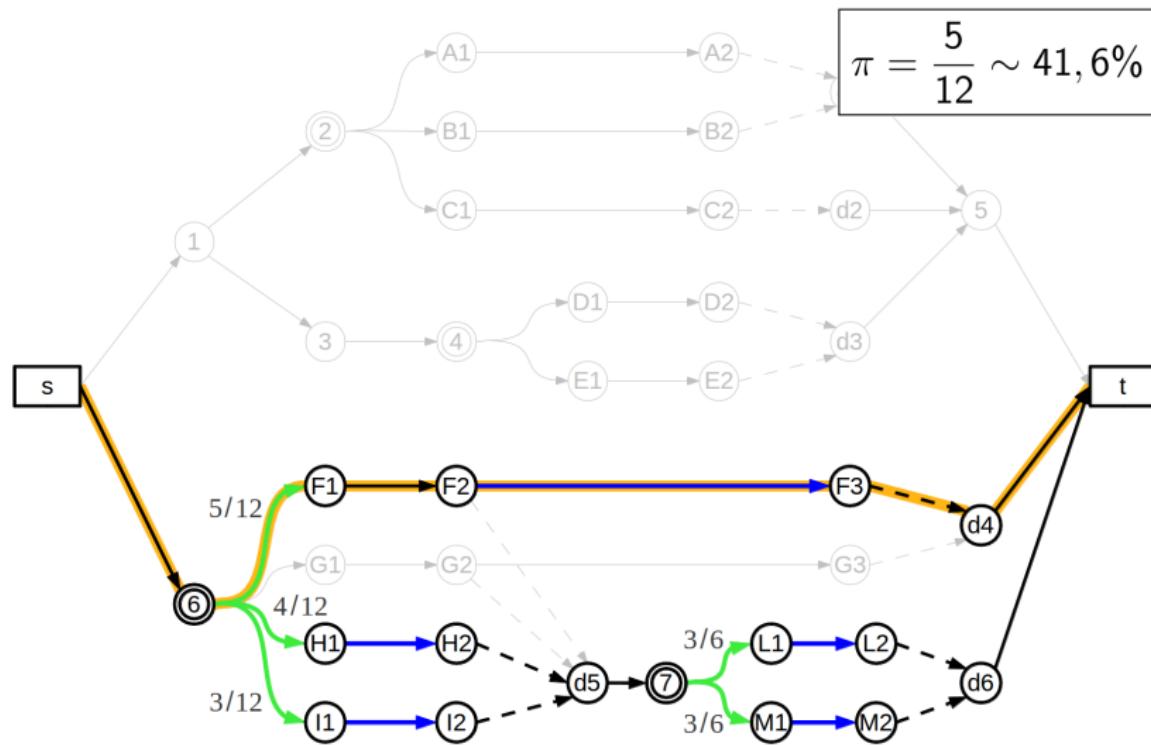
# Cammino



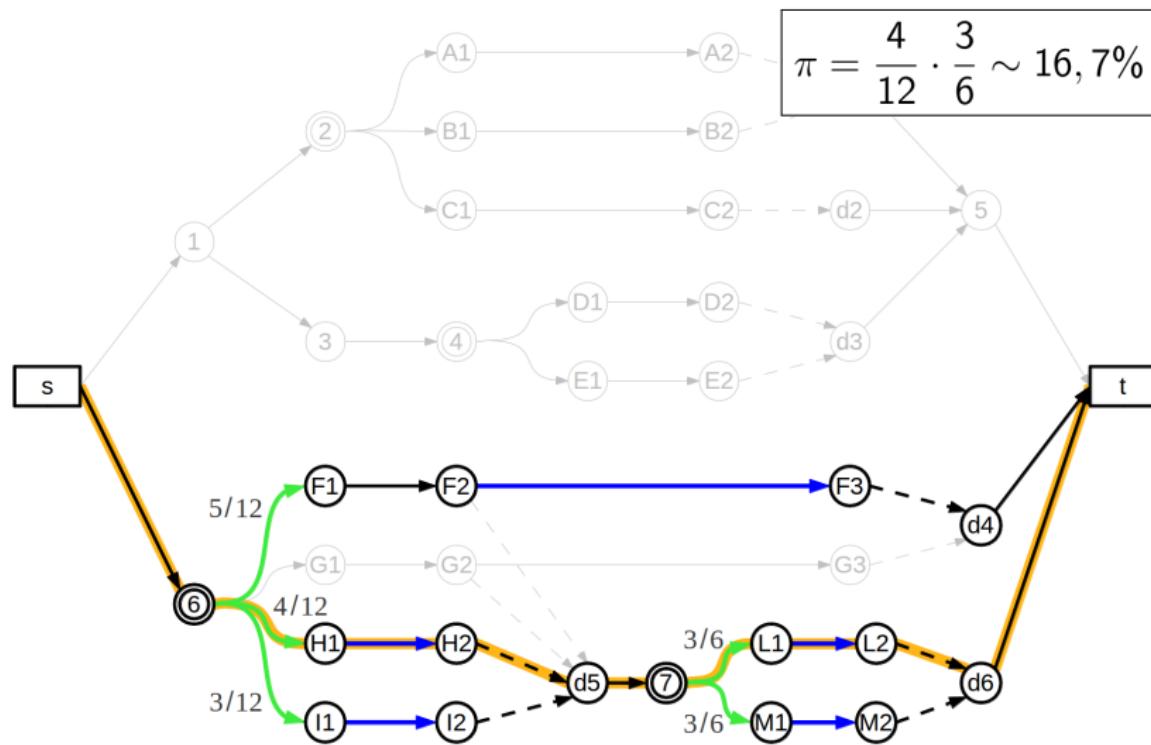
# Ipercammmino



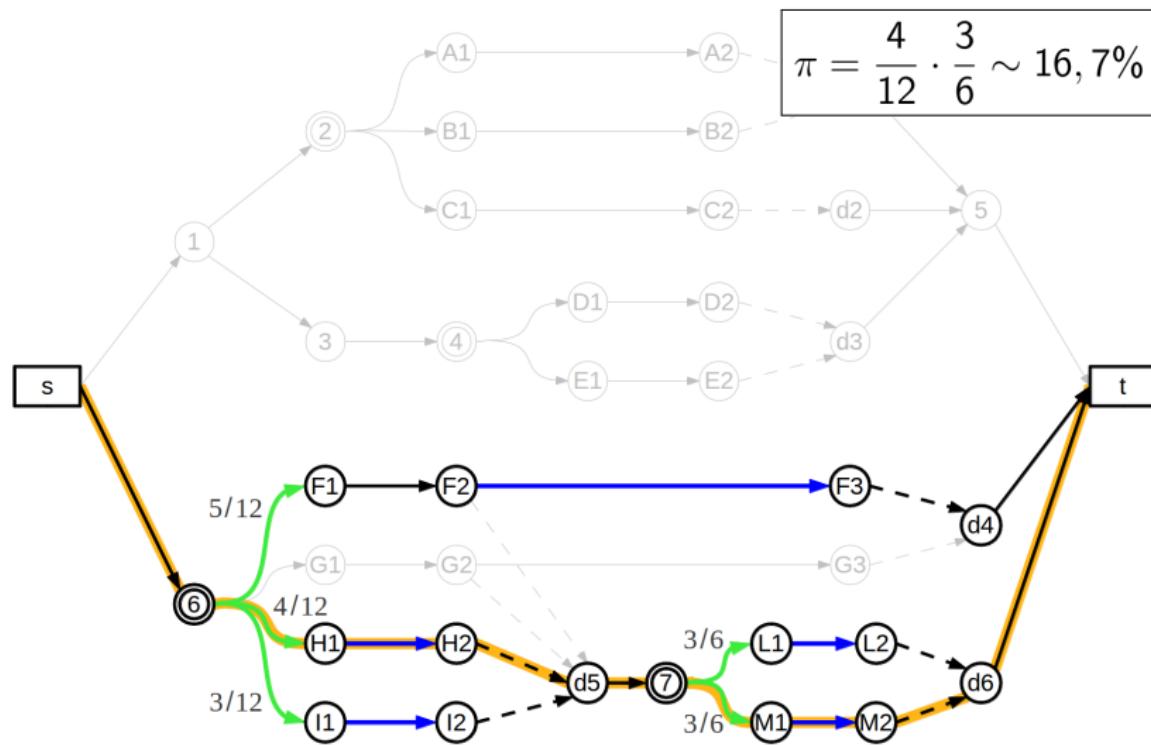
# Ipercammmino



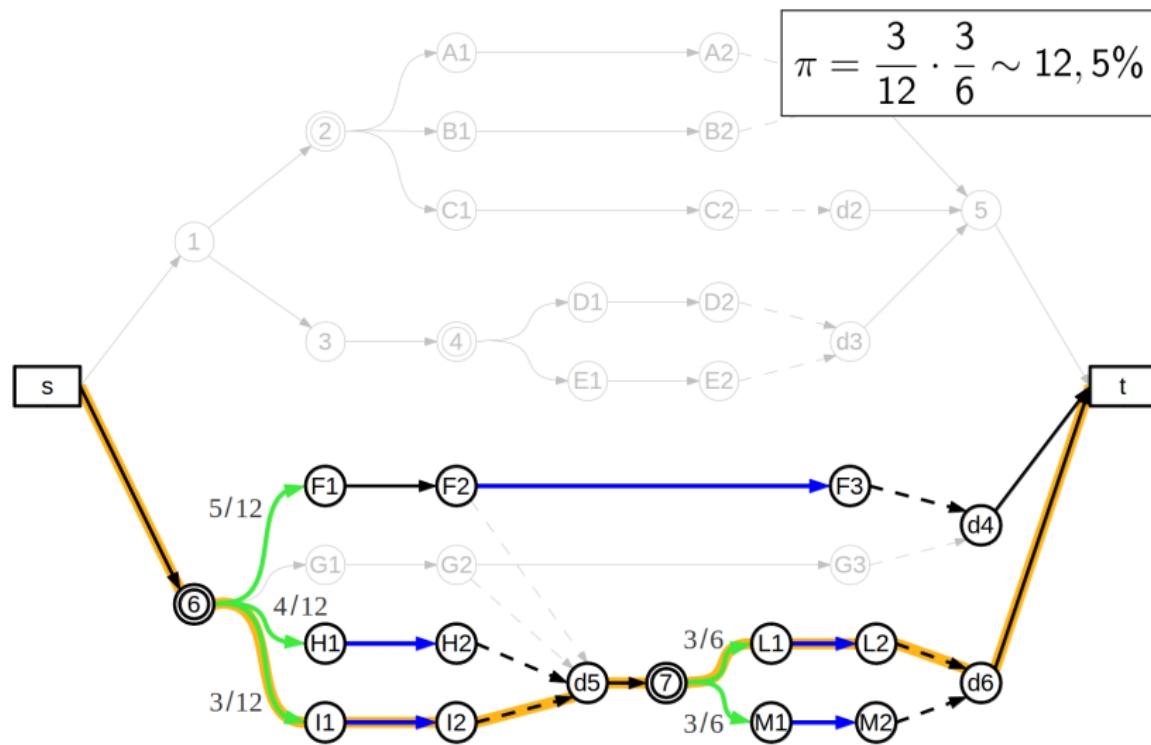
# Ipercammmino



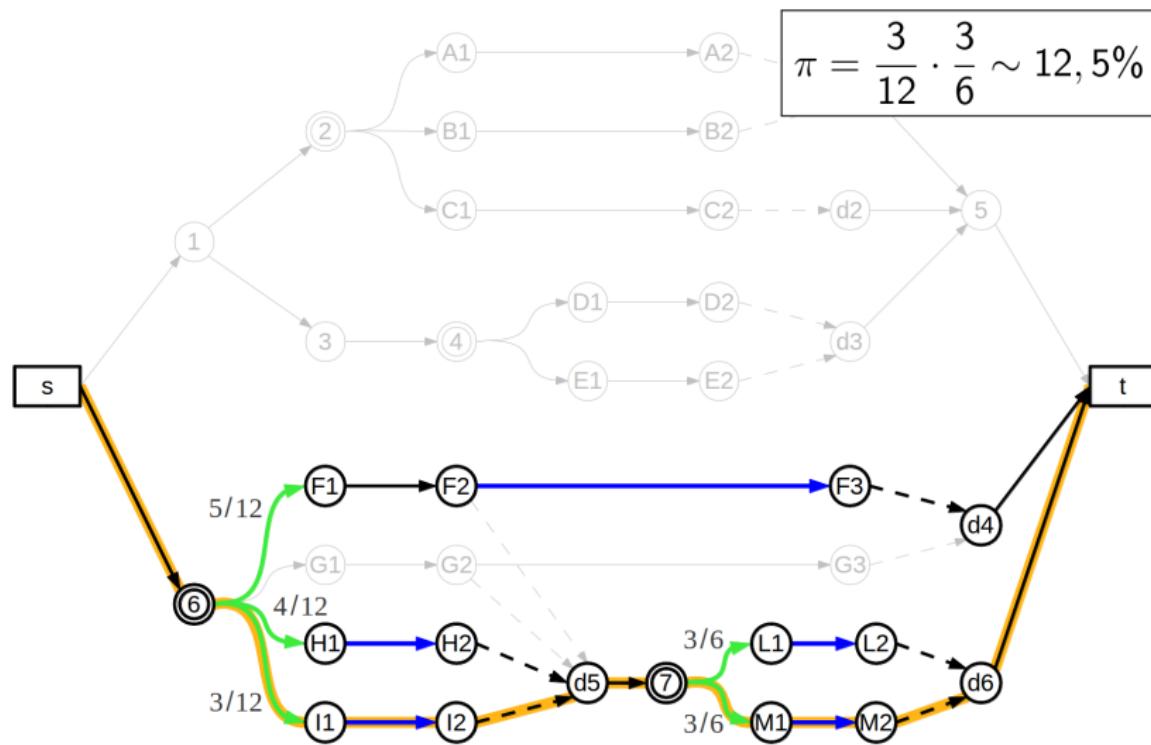
# Ipercammmino



# Ipercammmino



# Ipercammmino



# Costo di un ipercammino

Costo atteso per giungere da un nodo  $u$  alla destinazione:

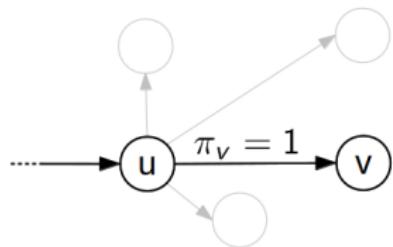
$$c_u = \sum_{(u,v)} \pi_v (c_{uv} + c_v)$$

# Costo di un ipercammino

Costo atteso per giungere da un nodo  $u$  alla destinazione:

$$c_u = \sum_{(u,v)} \pi_v (c_{uv} + c_v)$$

$$u \notin F$$



$$c_u = c_{uv} + c_v$$

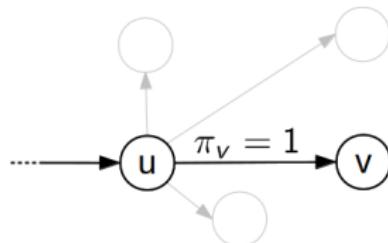
# Costo di un ipercammino

Costo atteso per giungere da un nodo  $u$  alla destinazione:

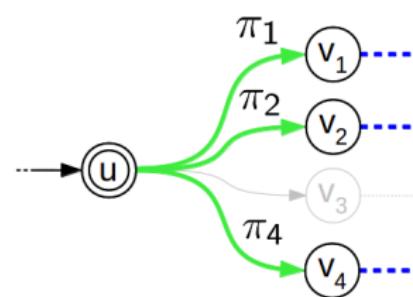
$$c_u = \sum_{(u,v)} \pi_v (c_{uv} + c_v)$$

$$u \in F$$

$$u \notin F$$



$$c_u = c_{uv} + c_v$$



$$c_u = \sum_{(u,v) \in L} \pi_v (w + c_v)$$

# Proprietà

## Ottimalità dei sotto-ipercammini

Per ogni generico nodo  $u$  è possibile costruire un ipercammino minimo  $p_u$  in modo che ciascun **sotto-ipercammino**  $p_v$  sia ipercammino minimo per il nodo  $v$ .

## Condizioni generalizzate di Bellman

L'ottimalità di un ipercammino è garantita se e solo se ogni suo nodo rispetta le condizioni generalizzate di Bellman.

$$c_u \leq \sum_{(u,v)} \pi_v (c_{uv} + c_v)$$

# Algoritmo Shortest Hyper Tree

- Determina l'**iperalbero minimo** di radice t
- Espande ricorsivamente i nodi dell'ipergrafo, partendo dal terminale e procedendo a **ritroso**
- Per ogni nodo esplorato viene controllata la condizione di Bellman, se non è rispettata il costo atteso viene aggiornato
- L'algoritmo termina quando non ci sono più nodi da espandere

# Algoritmo Shortest Hyper Tree - *priority*

L'insieme dei nodi da esplorare è memorizzato in una **coda con priorità**: ad ogni iterazione il nodo espanso è quello con il costo atteso corrente minore.

$$Q = \{v_3(7), v_6(10), v_8(12)\}$$



## Proprietà di monotonia

Ciascun nodo viene estratto una e una sola volta dalla coda.

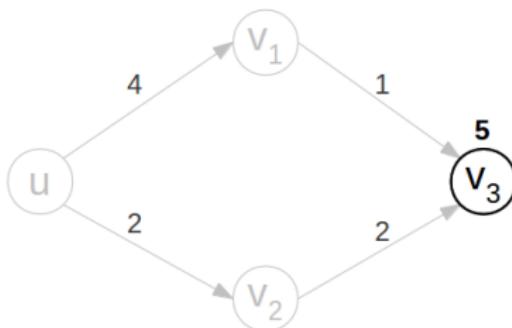
# Aggiornamento dei costi - Nodi normali

if  $c_u > c_v + c_{uv}$  {

$$c_u = c_v + c_{uv}$$

$$\text{successor}_u = \{v\}$$

}



$$Q = \{v_3\}$$

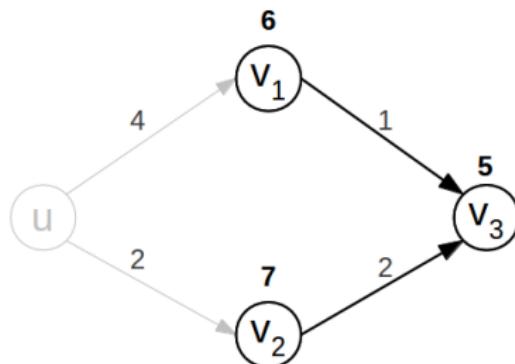
# Aggiornamento dei costi - Nodi normali

if  $c_u > c_v + c_{uv}$  {

$$c_u = c_v + c_{uv}$$

$$\text{successor}_u = \{v\}$$

}



$$Q = \{v_1, v_2\}$$

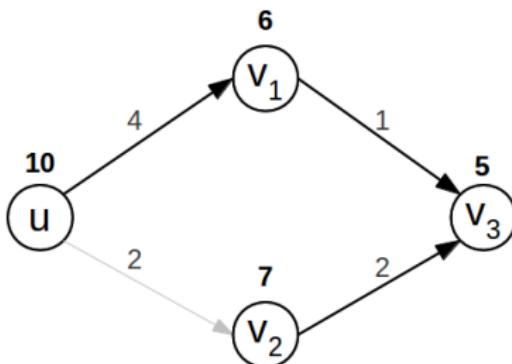
# Aggiornamento dei costi - Nodi normali

if  $c_u > c_v + c_{uv}$  {

$$c_u = c_v + c_{uv}$$

$$\text{successor}_u = \{v\}$$

}



$$Q = \{v_2, u\}$$

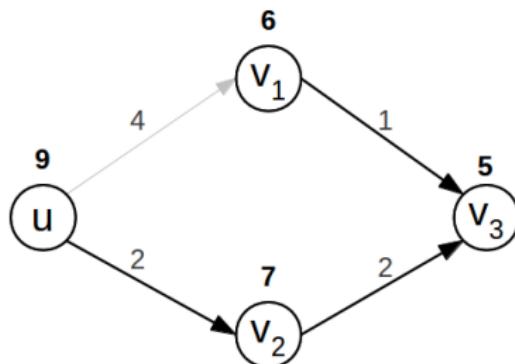
# Aggiornamento dei costi - Nodi normali

if  $c_u > c_v + c_{uv}$  {

$$c_u = c_v + c_{uv}$$

$$\text{successor}_u = \{v\}$$

}



$$Q = \{u\}$$

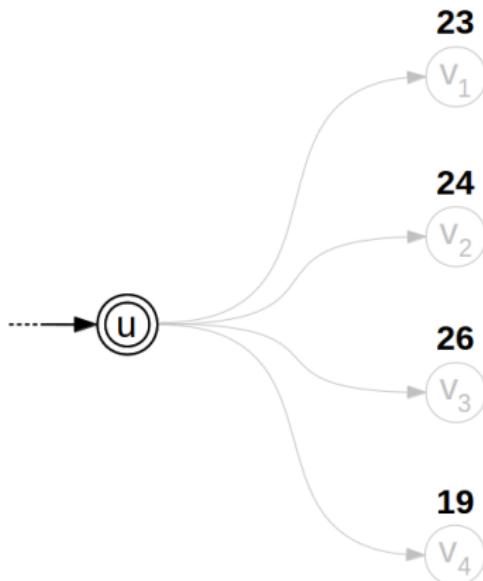
# Aggiornamento dei costi - Nodi fermata

if  $c_u > c_v$  {

$L \leftarrow \{v\}$

$$c_u = \sum_{(u,v) \in L} \pi_v (w + c_v)$$

}



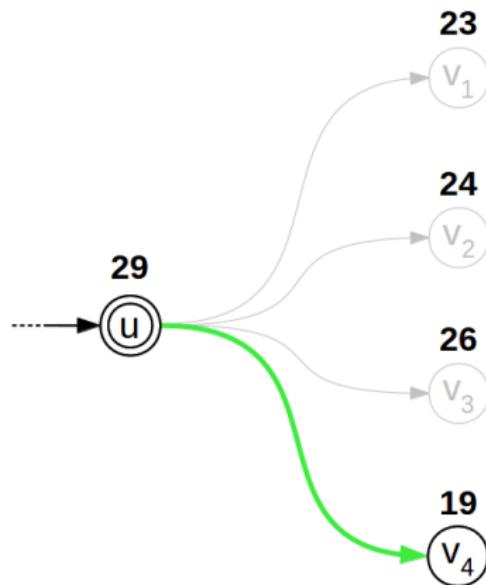
# Aggiornamento dei costi - Nodi fermata

if  $c_u > c_v$  {

$L \leftarrow \{v\}$

$$c_u = \sum_{(u,v) \in L} \pi_v (w + c_v)$$

}



$$L = \{v_4\}$$

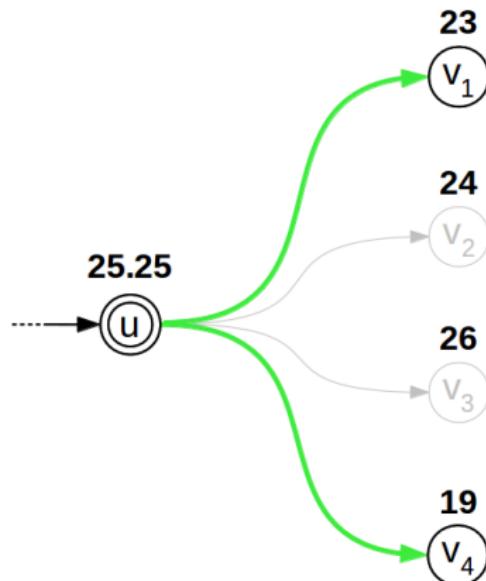
# Aggiornamento dei costi - Nodi fermata

if  $c_u > c_v$  {

$L \leftarrow \{v\}$

$$c_u = \sum_{(u,v) \in L} \pi_v (w + c_v)$$

}



$$L = \{v_4, v_1\}$$

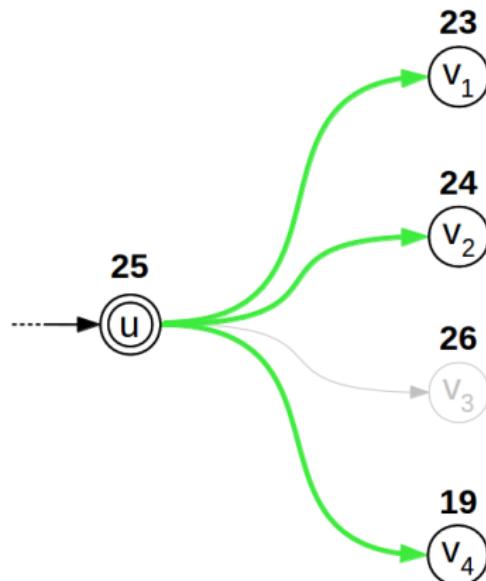
# Aggiornamento dei costi - Nodi fermata

if  $c_u > c_v$  {

$L \leftarrow \{v\}$

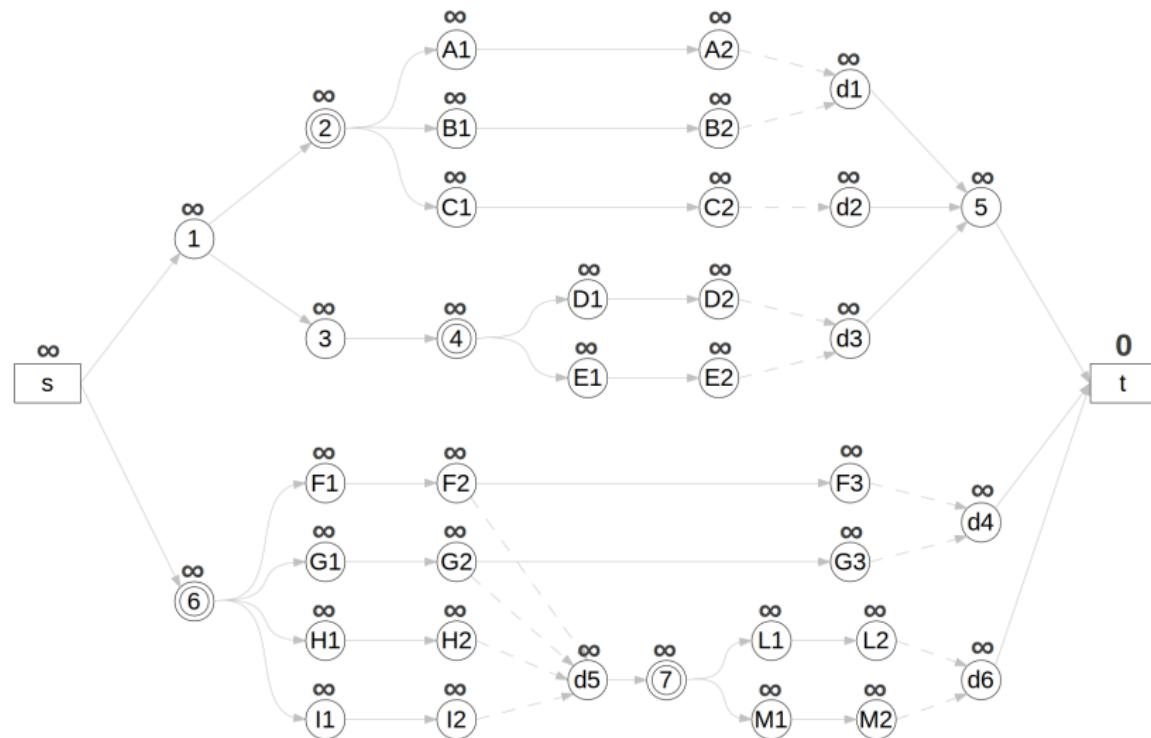
$$c_u = \sum_{(u,v) \in L} \pi_v (w + c_v)$$

}

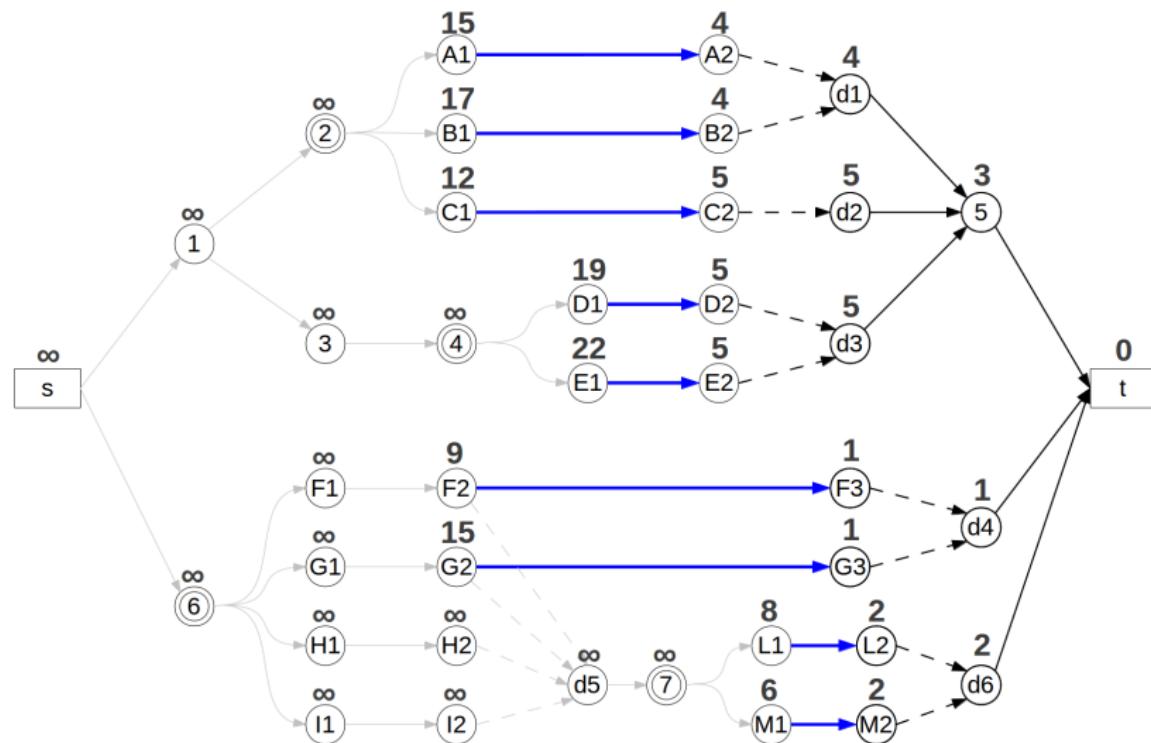


$$L = \{v_4, v_1, v_2\}$$

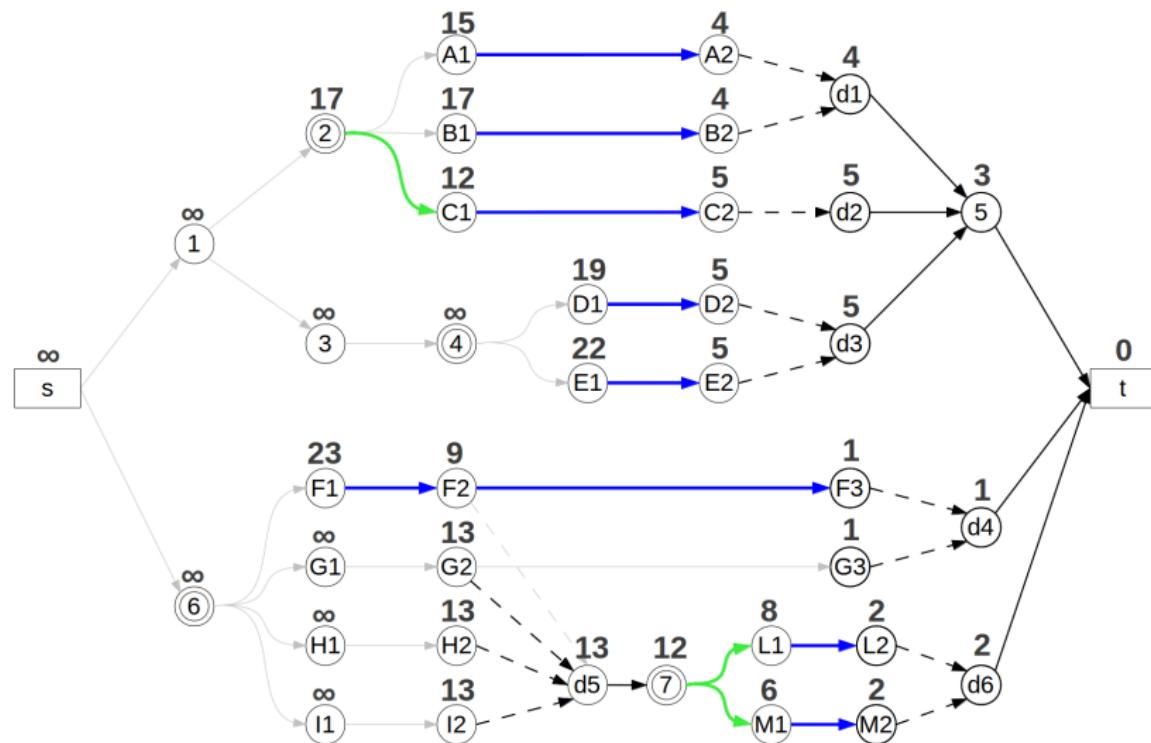
# Algoritmo Shortest Hyper Tree



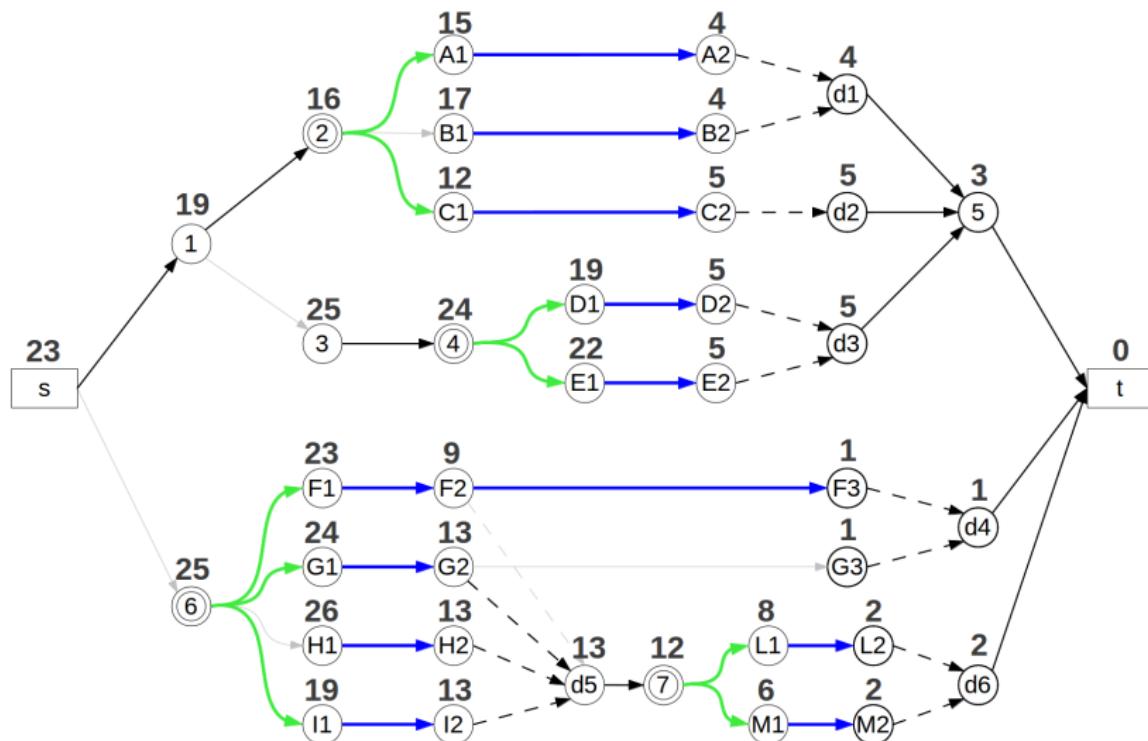
# Algoritmo Shortest Hyper Tree



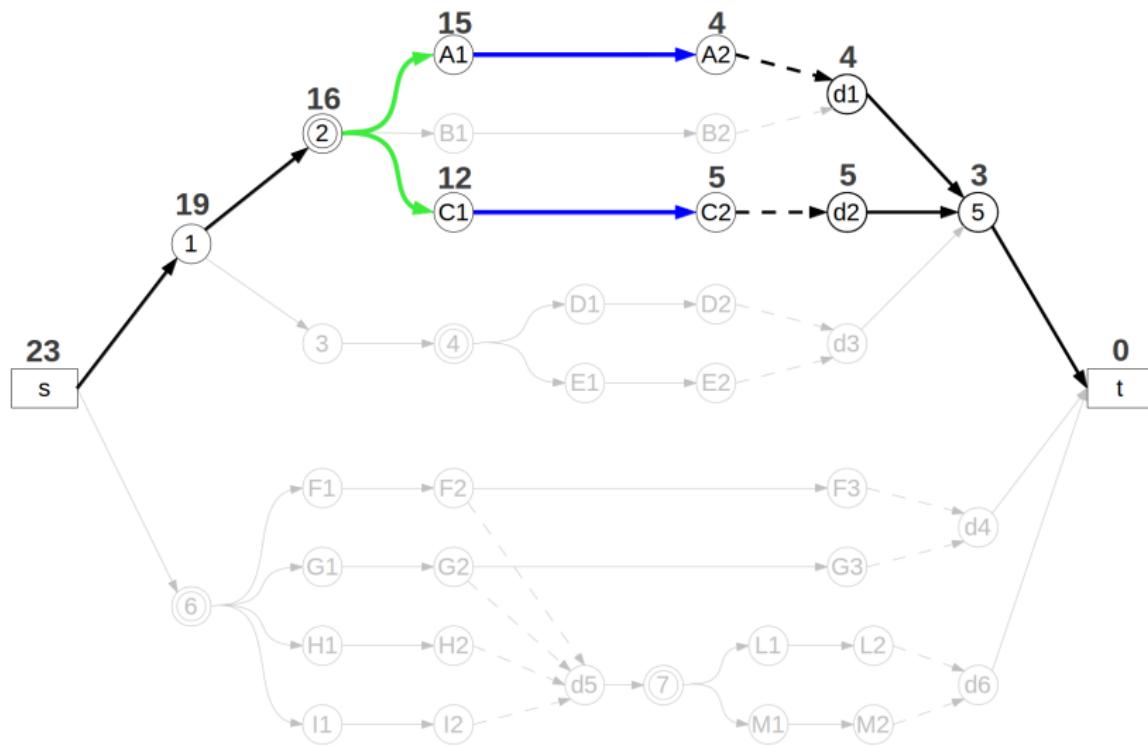
# Algoritmo Shortest Hyper Tree



# Algoritmo Shortest Hyper Tree



# Algoritmo Shortest Hyper Tree



# Implementazione

- Linguaggio C++
- Librerie **Boost**: basate su programmazione generica (*template*) per consentire flessibilità e riutilizzo del codice.

## Boost Graph Library

- Interfacce generiche
- Interoperabilità di algoritmi e strutture dati
- Facilità di estensione
- Multi-parametrizzazione di nodi e archi

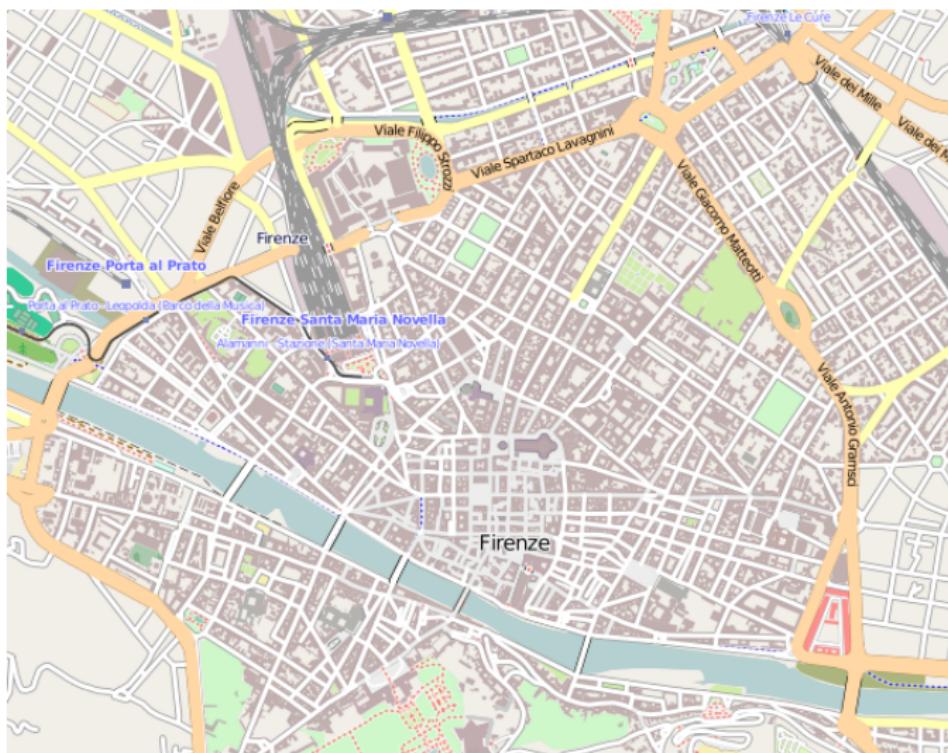
# Implementazione

L'ipergrafo è modellato con la classe **boost::adjacency\_list**

- Insieme dei nodi
- Insieme degli archi uscenti

```
struct vertex_info {  
    bool stop_vertex;  
    string name, lat, lon;  
};  
  
struct edge_info {  
    double weight;  
    string name;  
};
```

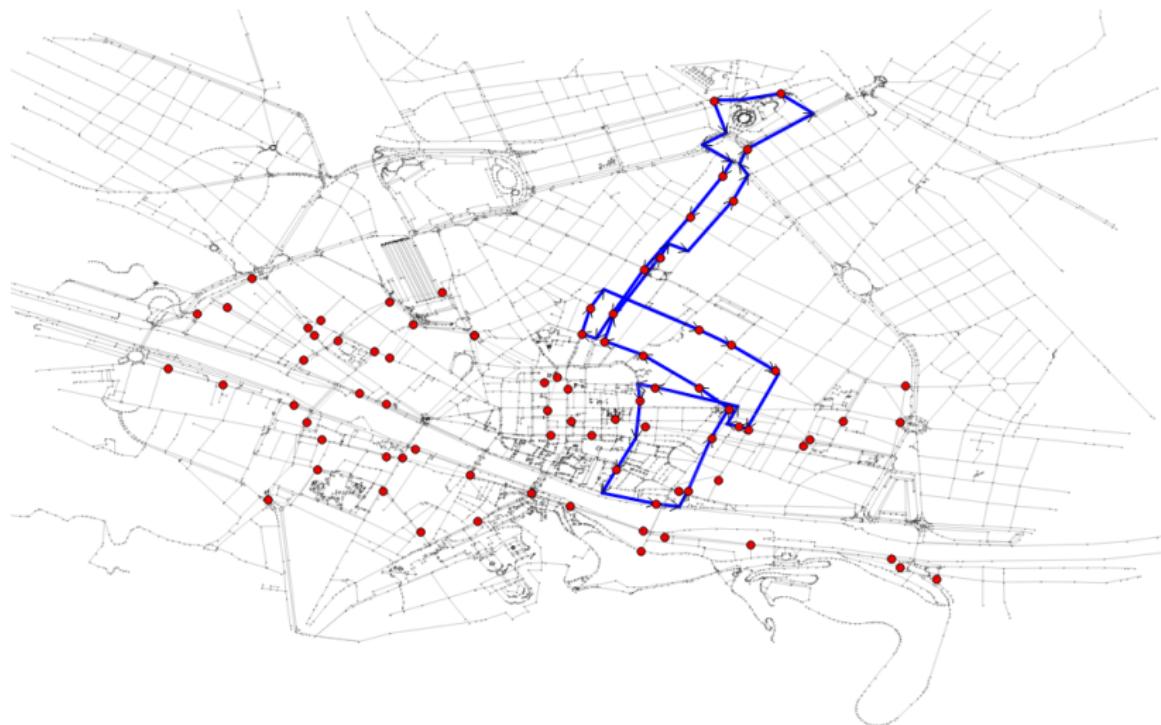
# La rete del centro di Firenze



# La rete del centro di Firenze



# La rete del centro di Firenze



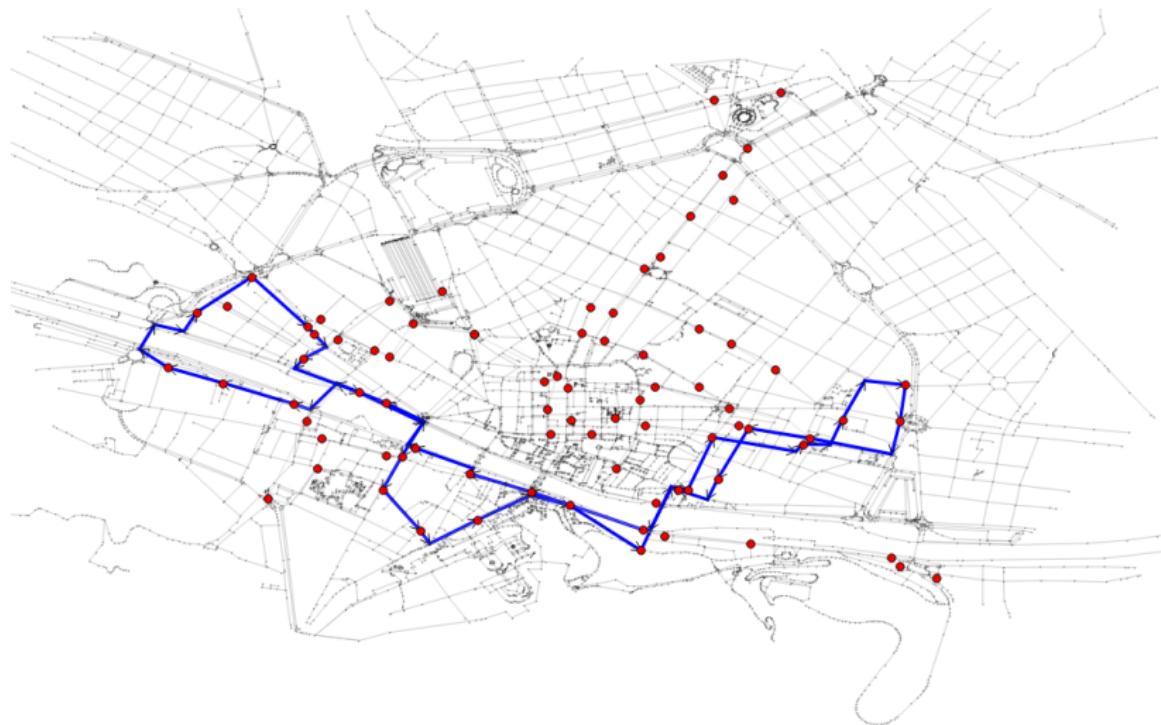
*Linea C1*

# La rete del centro di Firenze



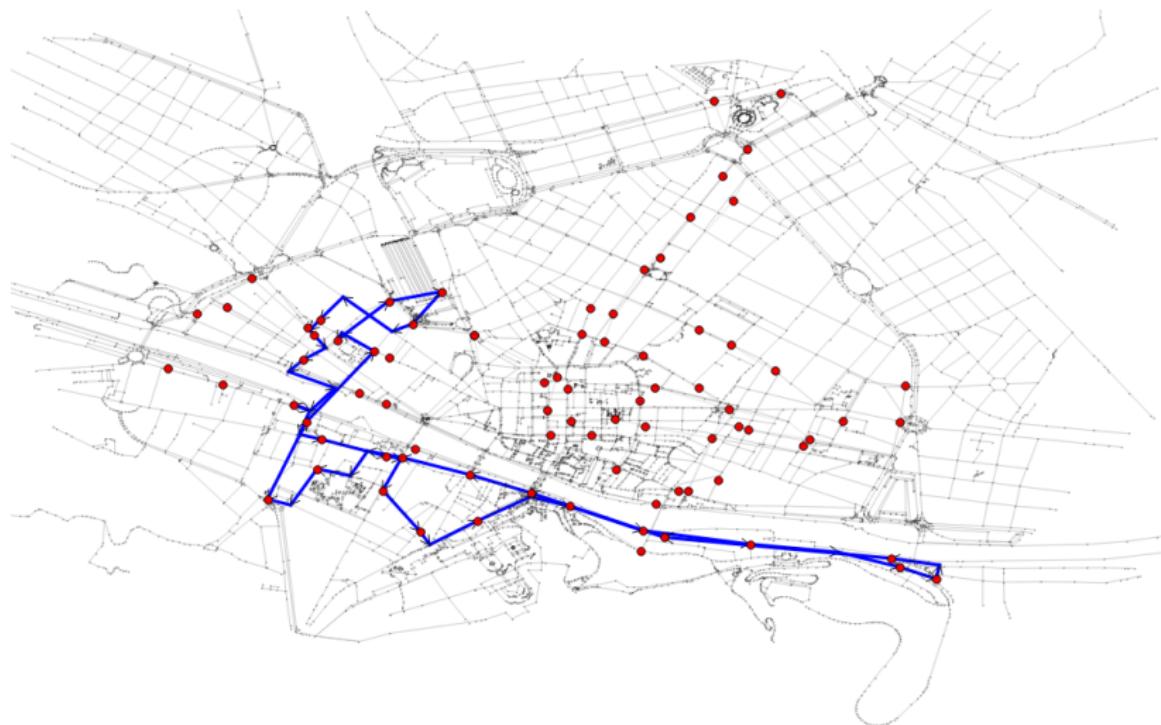
*Linea C2*

# La rete del centro di Firenze



*Linea C3*

# La rete del centro di Firenze



*Linea D*

# La rete del centro di Firenze

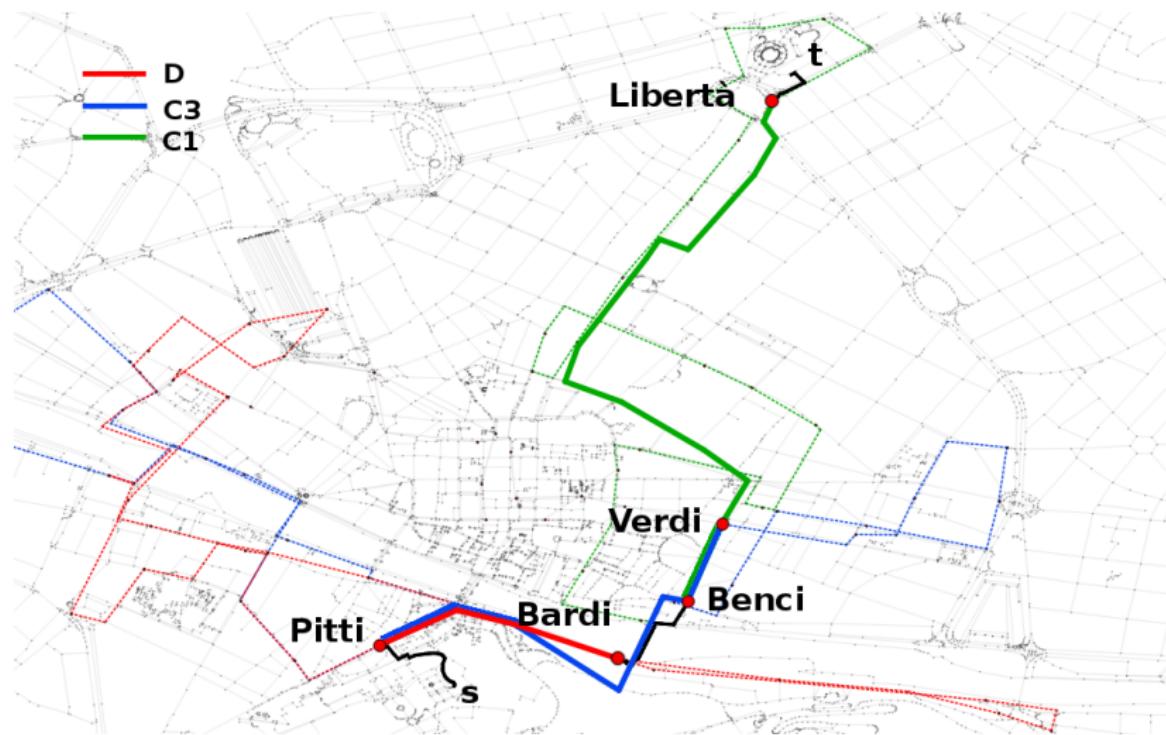
I tempi di percorrenza associati agli archi di viaggio sono stati ricavati assumendo le velocità medie seguenti:

$$\begin{cases} 1.2 \frac{m}{s} (\sim 4.3 \frac{km}{h}), & \text{per gli archi pedonali} \\ 7 \frac{m}{s} (\sim 25 \frac{km}{h}), & \text{per gli archi a bordo} \end{cases}$$

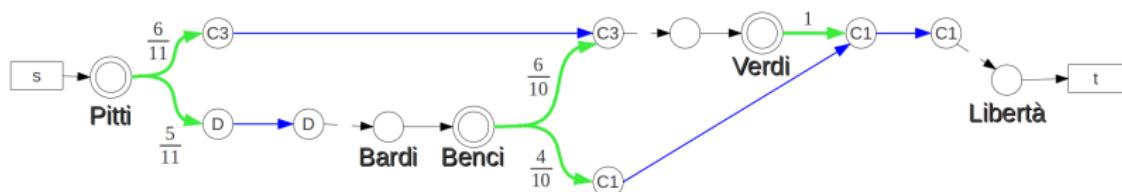
Le frequenze orarie dei bus sono:

$$\{\varphi_{C1}, \varphi_{C2}, \varphi_{C3}, \varphi_D\} = \{4, 8, 6, 5\}$$

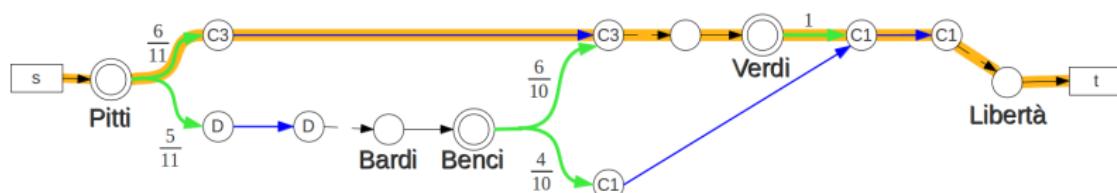
# Un esempio di ipercammino minimo



# Un esempio di ipercammino minimo



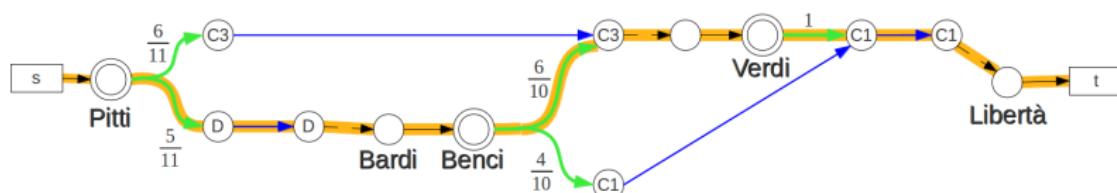
# Un esempio di ipercammino minimo



$$w = \begin{cases} Pitti : & 5 \text{ min} \\ Benci : & - \\ Verdi : & 7 \text{ min } 30 \text{ sec} \end{cases}$$

$$c = 29 \text{ min}$$

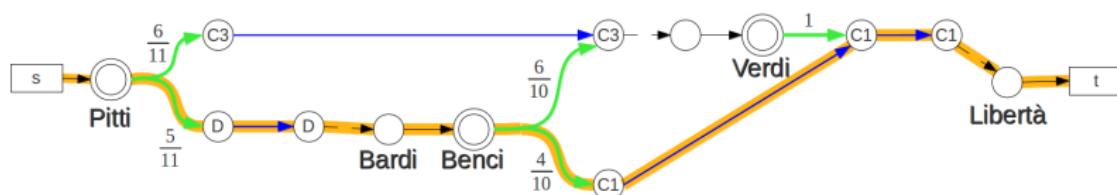
# Un esempio di ipercammino minimo



$$w = \begin{cases} Pitti : & 6 \text{ min} \\ Benci : & 5 \text{ min} \\ Verdi : & 7 \text{ min } 30 \text{ sec} \end{cases}$$

$$c = 27 \text{ min } 30 \text{ sec}$$

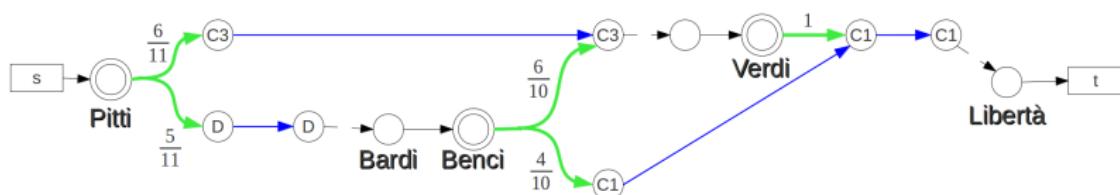
# Un esempio di ipercammino minimo



$$w = \begin{cases} Pitti : & 6 \text{ min} \\ Benci : & 7 \text{ min } 30 \text{ sec} \\ Verdi : & - \end{cases}$$

$$c = 30 \text{ min}$$

# Un esempio di ipercammino minimo



$$w = \begin{cases} Pitti : & 2 \text{ min } 45 \text{ sec} \\ Benci : & 3 \text{ min} \\ Verdi : & 7 \text{ min } 30 \text{ sec} \end{cases}$$

$$c = 24 \text{ min } 30 \text{ sec}$$

# Conclusioni

## Obiettivi raggiunti

- Modellazione del trasporto pubblico tramite ipergrafi
- Formalizzazione del problema degli ipercammini minimi
- Studio e implementazione di algoritmi

## Sviluppi futuri

- Punto di partenza teorico per analisi successive
- Passo base per problemi più complessi (flussi di equilibrio)