



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Spanner e oracoli per il problema del k-waypoint routing in grafi temporali

Introduzione

Introduzione informale al problema

Introduzione

❖ Campo di studio:

- *Grafi temporali*: variante dei grafi in cui ad ogni arco è associato un istante di tempo (timestamp) in cui questo può venir attraversato.
- In questa tesi si analizzano i cammini di tipo **k-waypoint routing**:
 - Dati due nodi s e t , un cammino *k-waypoint routing* con waypoint x_1, \dots, x_k è un *cammino temporale* da s a t che passa per tutti i nodi x_1, \dots, x_k , in qualsiasi ordine.

❖ Obiettivo di ricerca:

- Progettare dei buoni *k-waypoint routing spanner* e *oracoli*

Definizioni

Definizione degli strumenti utilizzati all'interno della tesi

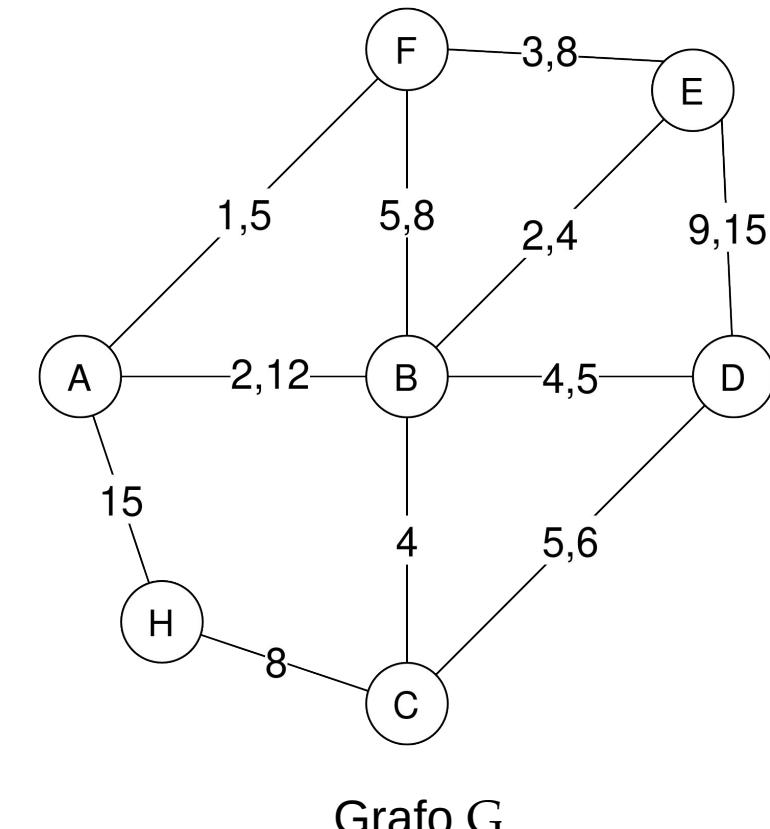
Definizioni

❖ **Grafo temporale:** $G = (V, E, \lambda)$

- $|V| = n$
- $|E| = m$
- $\lambda: E \rightarrow 2^{\mathbb{N}}$

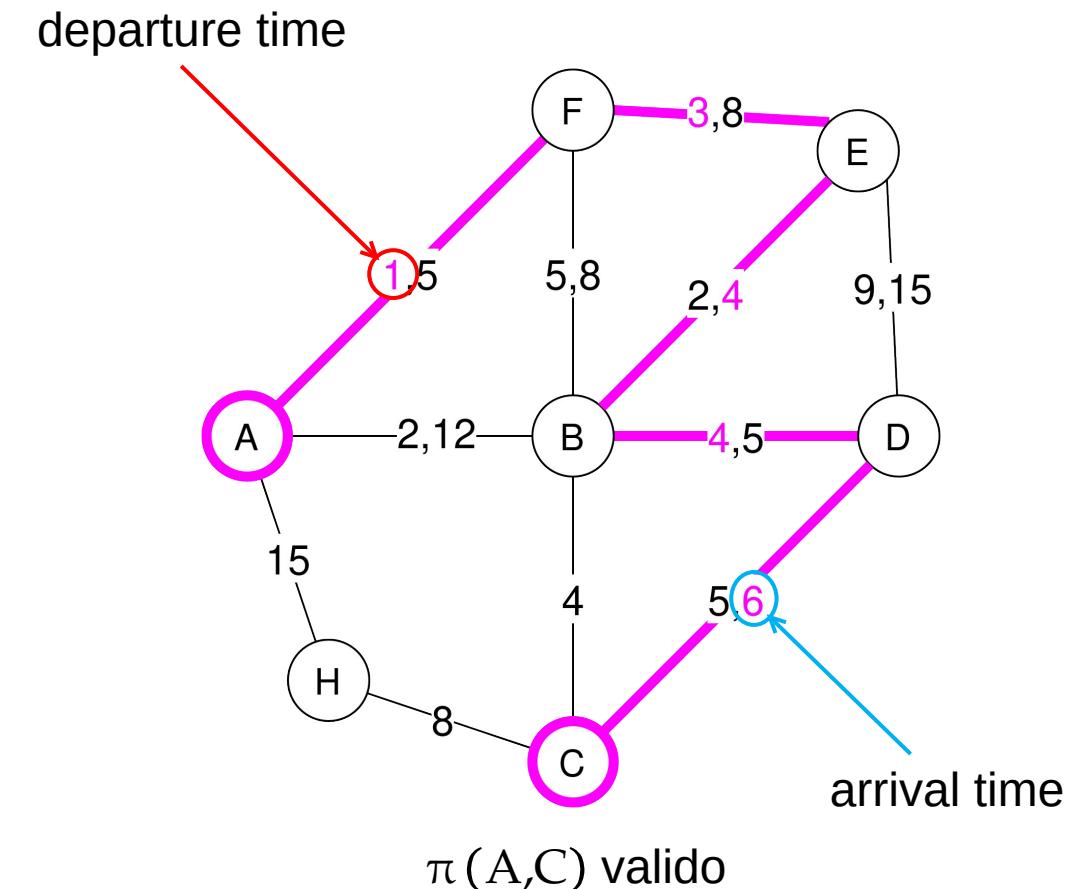
❖ **Stream temporale:**

- Insieme degli archi temporali di G in ordine crescente
 - $(A, F, 1)$
 - $(A, B, 2)$
 - $(B, E, 2)$
 - $(F, E, 3)$
 - $(B, E, 4)$
- $|\text{Stream}| = M$



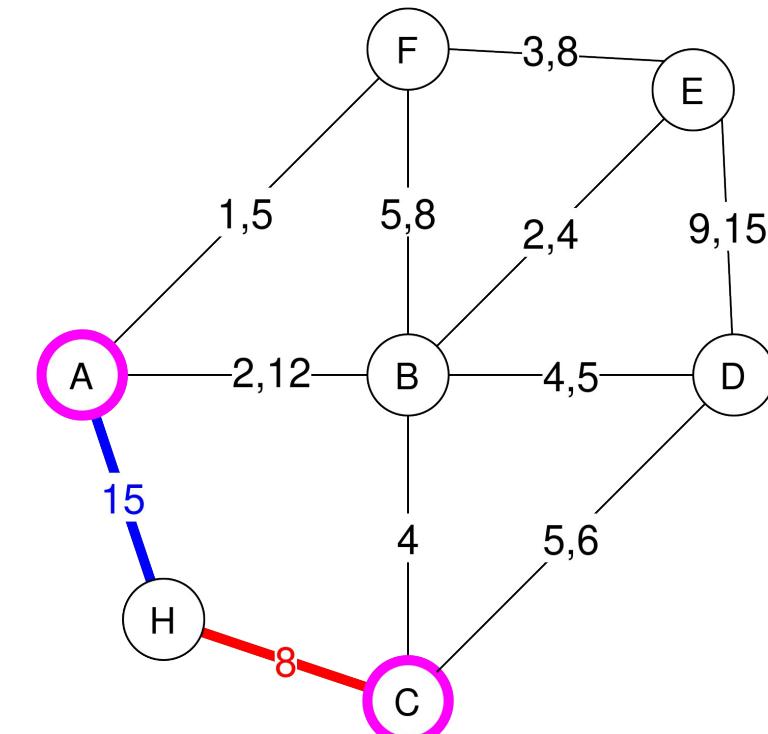
Definizioni

- ❖ **Cammino Temporale valido:** $\pi(x,y)$ con $x,y \in V$
- Affinché $\pi(x,y)$ sia valido deve attraversare archi con ordine di timestamp non decrescente



Definizioni

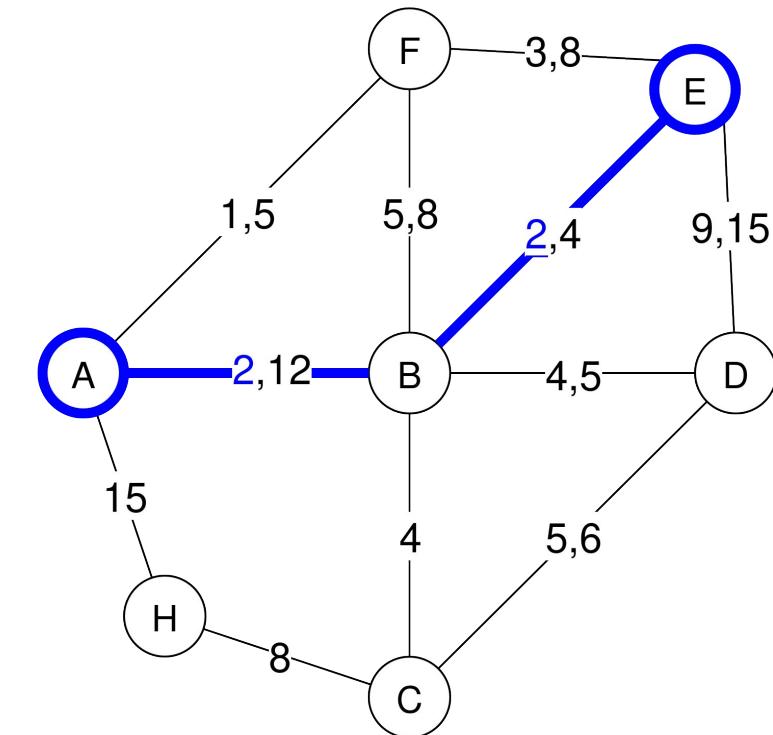
- ❖ **Cammino Temporale valido:** $\pi(x,y)$ con $x,y \in V$
- Affinché $\pi(x,y)$ sia valido deve attraversare archi con ordine di timestamp non decrescente



$\pi(A,C)$ non valido

Definizioni

- ❖ **Earliest arrival time:**
 $ea_x[y]$ è definito come l'istante di tempo minimo con cui si arriva a y partendo da x
- ❖ **Earliest arrival path:**
 $ea(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile raggiungere y con tempo minimo partendo da x
- ❖ **Earliest arrival Tree:**
 sottografo $T_{ea}[x]$ di G con topologia ad albero contenente $\forall y \in V$ gli earliest arrival path $ea(x,y)$ dove x è la radice di $T_{ea}[x]$



$$ea_A[E] = 2$$

Definizioni

- ❖ **Earliest arrival time:**

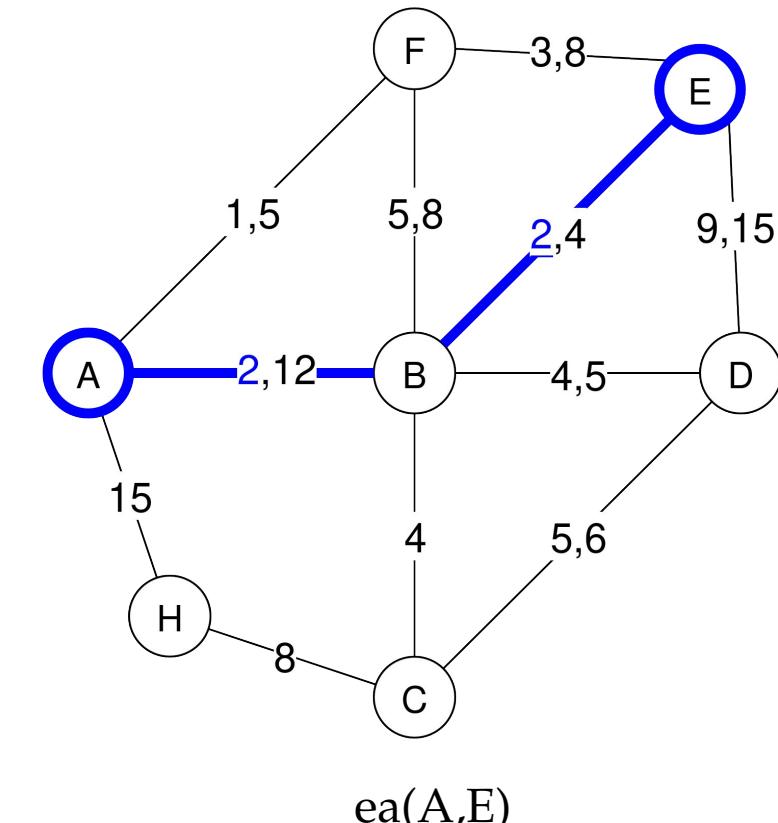
$ea_x[y]$ è definito come l'istante di tempo minimo con cui si arriva a y partendo da x

- ❖ **Earliest arrival path:**

$ea(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile raggiungere y con tempo minimo partendo da x

- ❖ **Earliest arrival Tree:**

sottografo $T_{ea}[x]$ di G con topologia ad albero contenente $\forall y \in V$ gli earliest arrival path $ea(x,y)$ dove x è la radice di $T_{ea}[x]$



Definizioni

- ❖ **Earliest arrival time:**

$ea_x[y]$ è definito come l'istante di tempo minimo con cui si arriva a y partendo da x

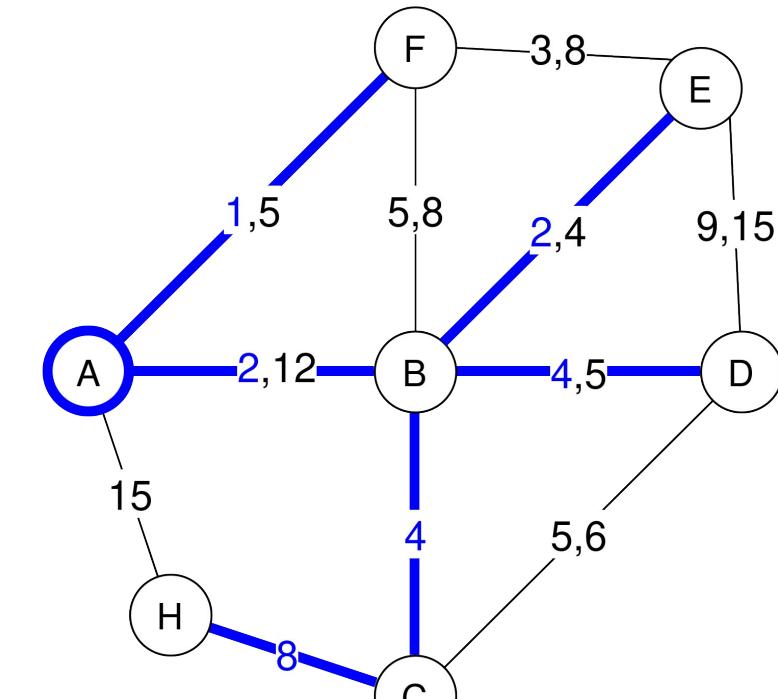
- ❖ **Earliest arrival path:**

$ea(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile raggiungere y con tempo minimo partendo da x

- ❖ **Earliest arrival Tree:**

sottografo $T_{ea}[x]$ di G con topologia ad albero contenente $\forall y \in V$ gli earliest arrival path $ea(x,y)$ dove x è la radice di $T_{ea}[x]$

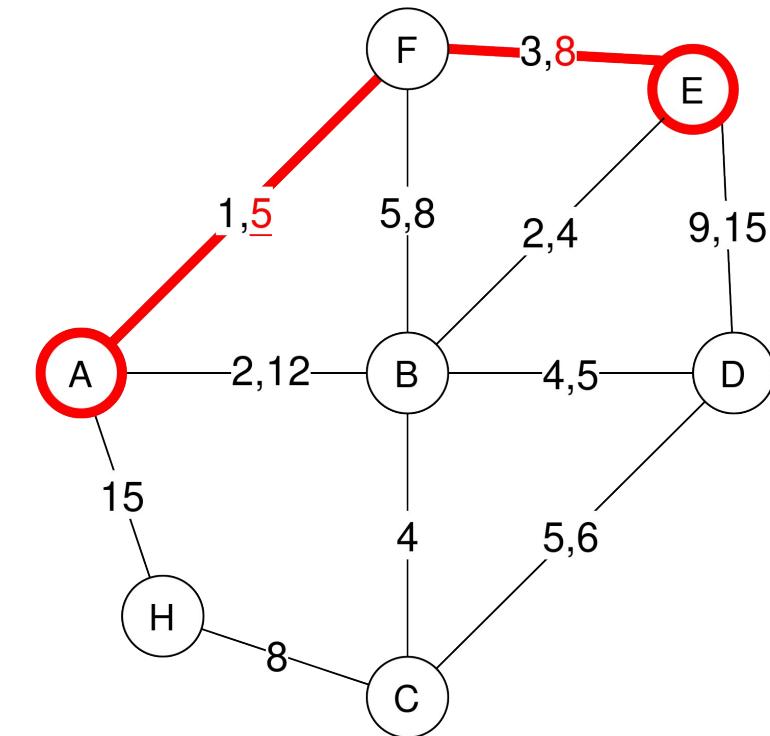
Calcolabile in tempo $O(n + M)$



$T_{ea}[A]$

Definizioni

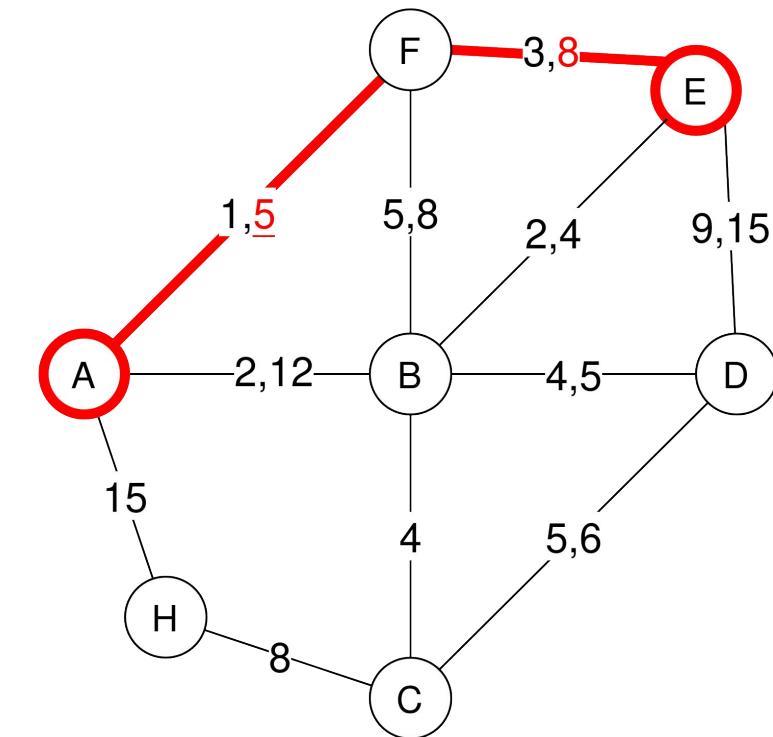
- ❖ **Latest departure time:**
 $ld_y[x]$ è definito come l'istante di tempo massimo con cui si parte da x per raggiungere y
- ❖ **Latest departure path:**
 $ld(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile partire da x con tempo massimo per raggiungere y
- ❖ **Latest departure Tree:**
 sottografo $T_{ld}[y]$ di G con topologia ad albero contenente $\forall x \in V$ i latest departure path $ld(x,y)$ dove y è la radice di $T_{ld}[y]$



$$ld_E[A] = 5$$

Definizioni

- ❖ **Latest departure time:**
 $ld_y[x]$ è definito come l'istante di tempo massimo con cui si parte da x per raggiungere y
- ❖ **Latest departure path:**
 $ld(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile partire da x con tempo massimo per raggiungere y
- ❖ **Latest departure Tree:**
 sottografo $T_{ld}[y]$ di G con topologia ad albero contenente $\forall x \in V$ i latest departure path $ld(x,y)$ dove y è la radice di $T_{ld}[y]$



$ld(A,E)$

Definizioni

- ❖ **Latest departure time:**

$ld_y[x]$ è definito come l'istante di tempo massimo con cui si parte da x per raggiungere y

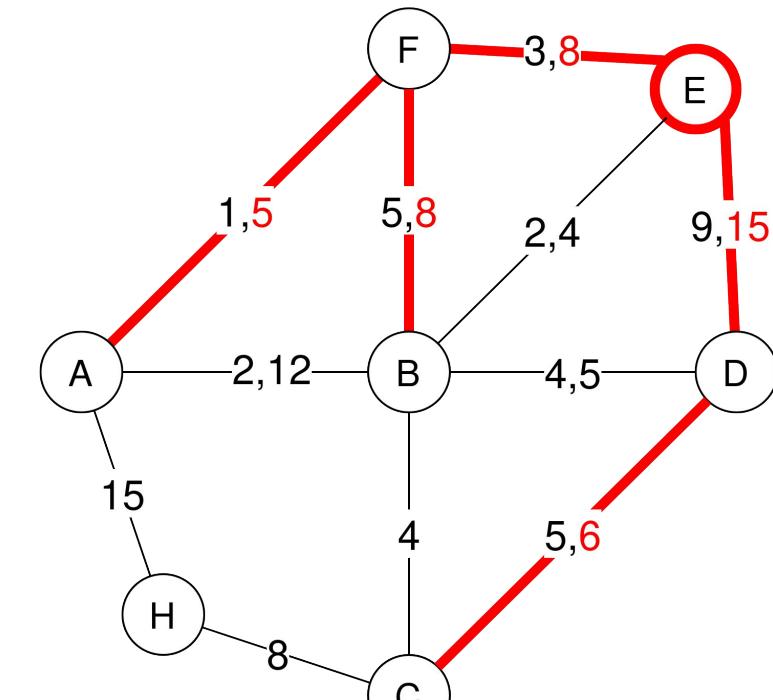
- ❖ **Latest departure path:**

$ld(x,y)$ è definito come $\pi(x,y)$ valido tale che è possibile partire da x con tempo massimo per raggiungere y

- ❖ **Latest departure Tree:**

sottografo $T_{ld}[y]$ di G con topologia ad albero contenente $\forall x \in V$ i latest departure path $ld(x,y)$ dove y è la radice di $T_{ld}[y]$

Calcolabile in tempo $O(n + M)$



$T_{ld}[E]$

Definizione dei problemi

Definizione e risultati per i problemi studiati

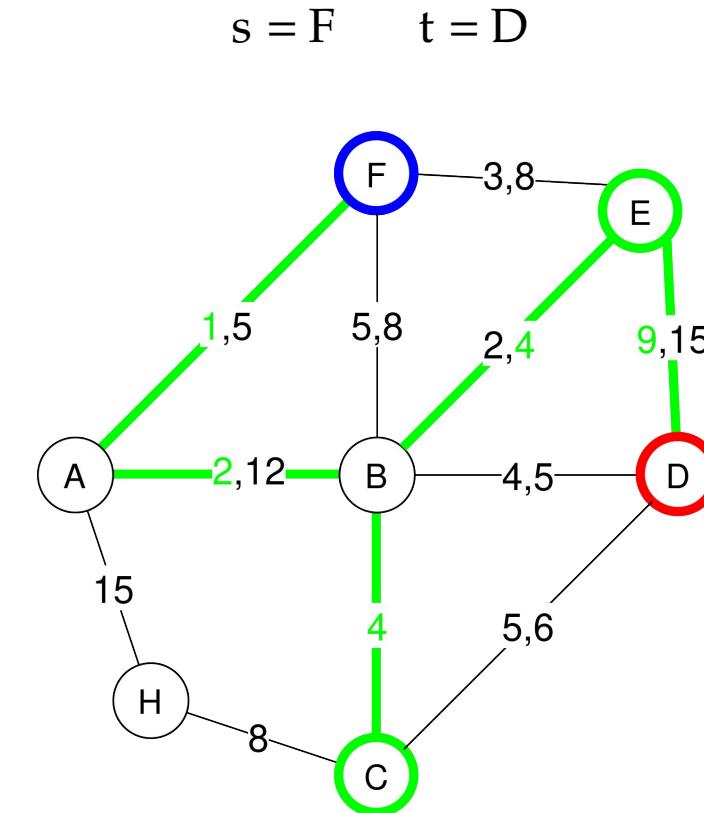
k-waypoint routing: Oracolo

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$



$$Q(\{C, E\}) = \text{True}$$

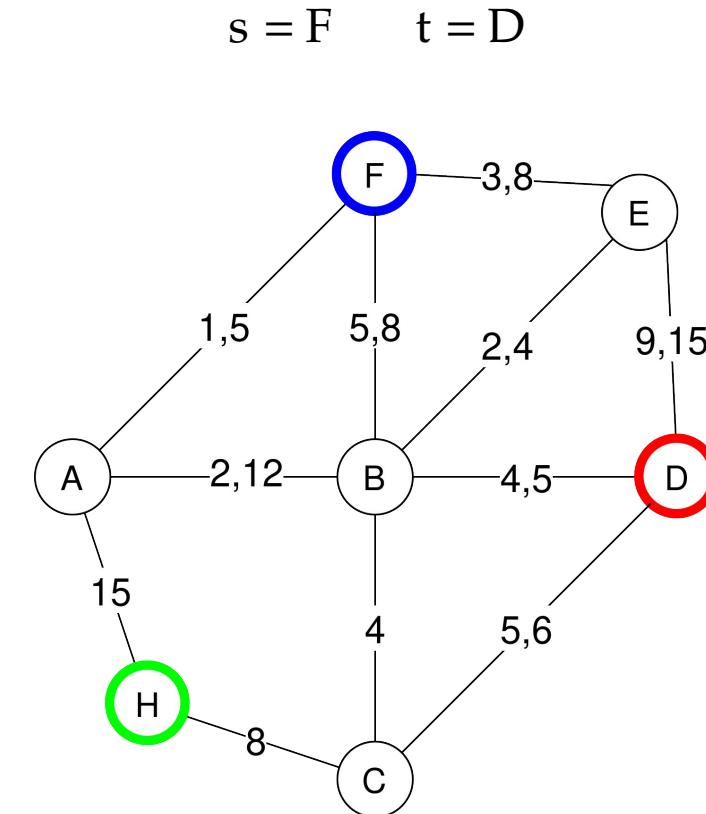
k-waypoint routing: Oracolo

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$



$$Q(\{H\}) = \text{False}$$

k-waypoint routing: Oracolo

❖ Input:

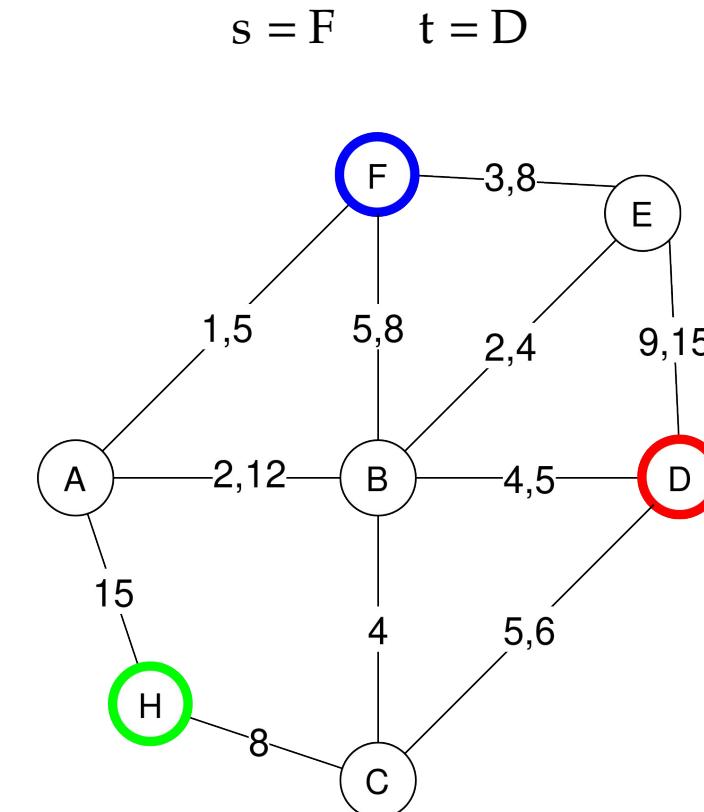
- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$

➤ Misure di qualità:

- Building time
- Dimensione
- Query time



$$Q(\{H\}) = \text{False}$$

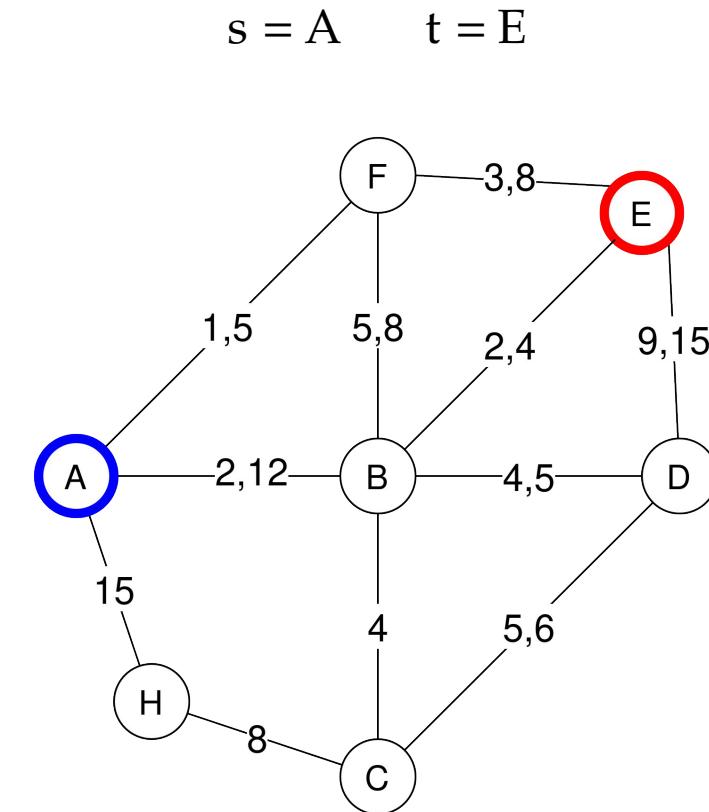
k-waypoint routing: Spanner

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Uno spanner temporale H di G in cui $\forall K = \{x_1, \dots, x_k\} \subseteq V$ valga la relazione:
- $\exists \pi(s, t | x_1, \dots, x_k) \in H \Leftrightarrow \exists \pi(s, t | x_1, \dots, x_k) \in G$



Grafo originale G

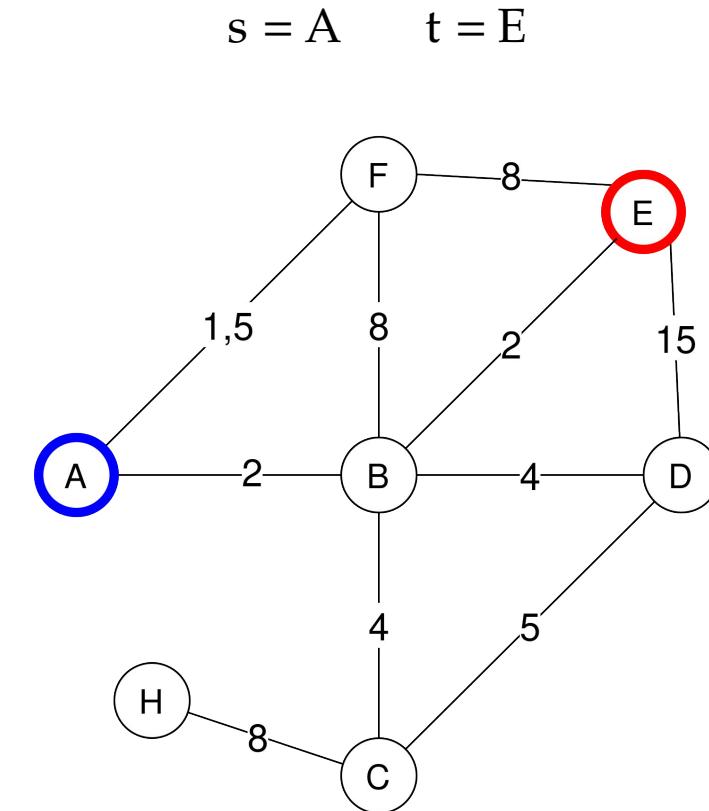
k-waypoint routing: Spanner

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Uno spanner temporale H di G in cui $\forall K = \{x_1, \dots, x_k\} \subseteq V$ valga la relazione:
- $\exists \pi(s, t | x_1, \dots, x_k) \in H \Leftrightarrow \exists \pi(s, t | x_1, \dots, x_k) \in G$



Spanner H per $k = 1$

k-waypoint routing: Spanner

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

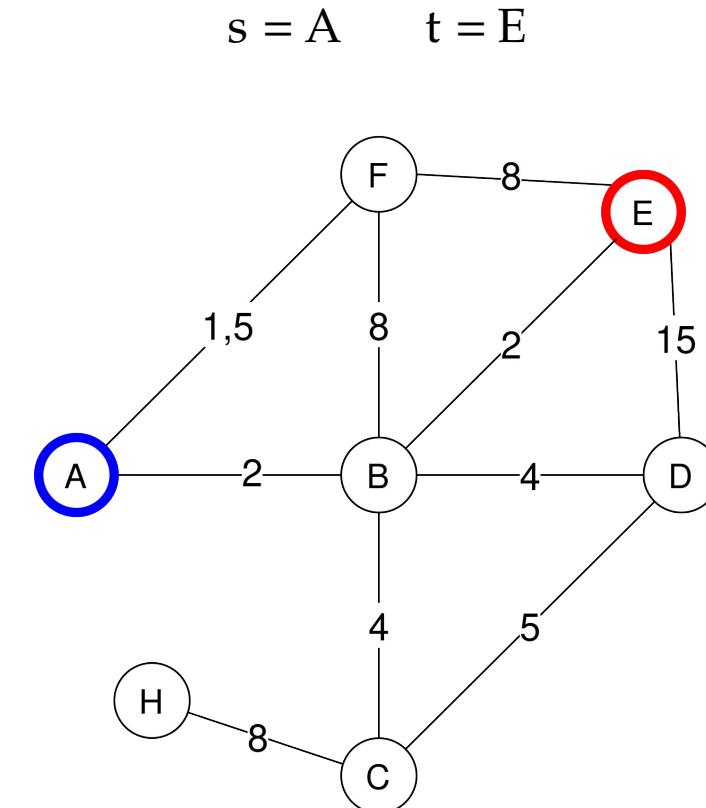
❖ Output:

- Uno spanner temporale H di G in cui $\forall K = \{x_1, \dots, x_k\} \subseteq V$ valga la relazione:
- $\exists \pi(s, t | x_1, \dots, x_k) \in H \Leftrightarrow \exists \pi(s, t | x_1, \dots, x_k) \in G$

➢ Misure di qualità:

- Dimensione
- Building time

in termini di numero di archi in H



Spanner H per $k = 1$

k-waypoint routing: Risultati

	Oracolo	Spanner
Parametro	$k = 1$	
Building Time	$O(n + M)$	$O(n + M)$
Dimensione	$O(n)$	$O(n)$
Query time	$O(1)$	
Parametro	$k \geq 2$	
Dimensione		$\Omega(n^2)$

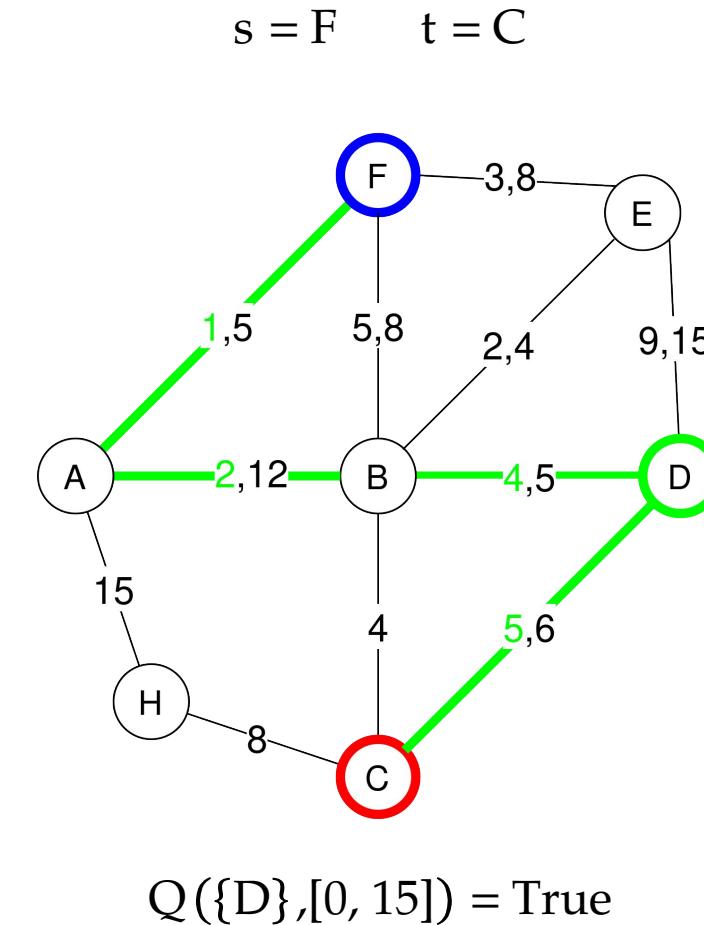
k-waypoint routing with window: Oracolo

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Dato un'intervallo $[\alpha, \beta] \in \mathbb{N}$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$ tale che ogni arco viene attraversato ad un istante $t \in [\alpha, \beta]$



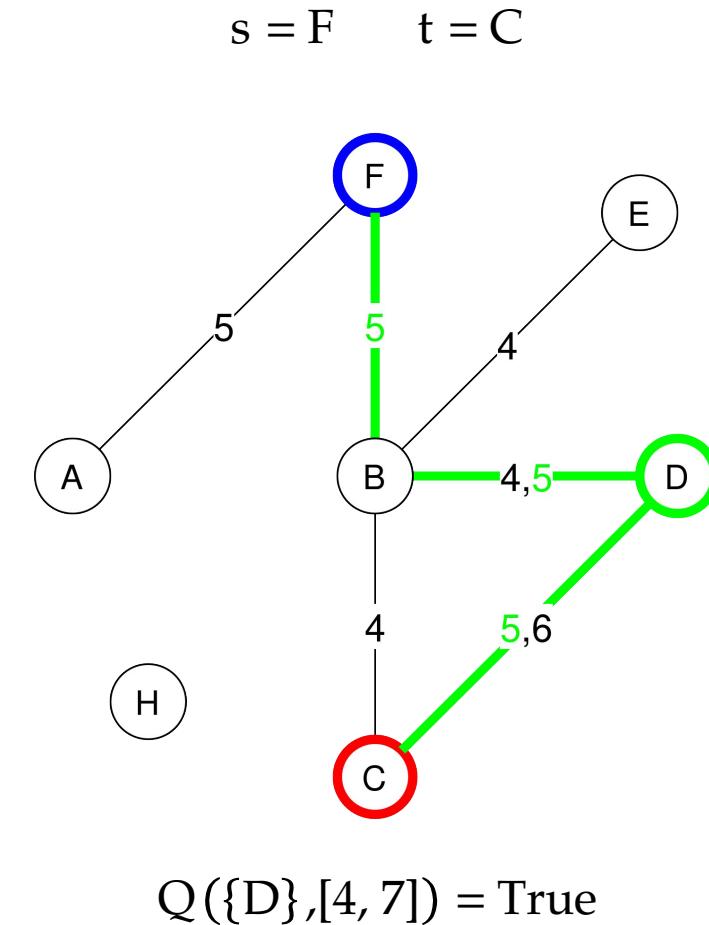
k-waypoint routing with window: Oracolo

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Dato un'intervallo $[\alpha, \beta] \in \mathbb{N}$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$ tale che ogni arco viene attraversato ad un istante $t \in [\alpha, \beta]$



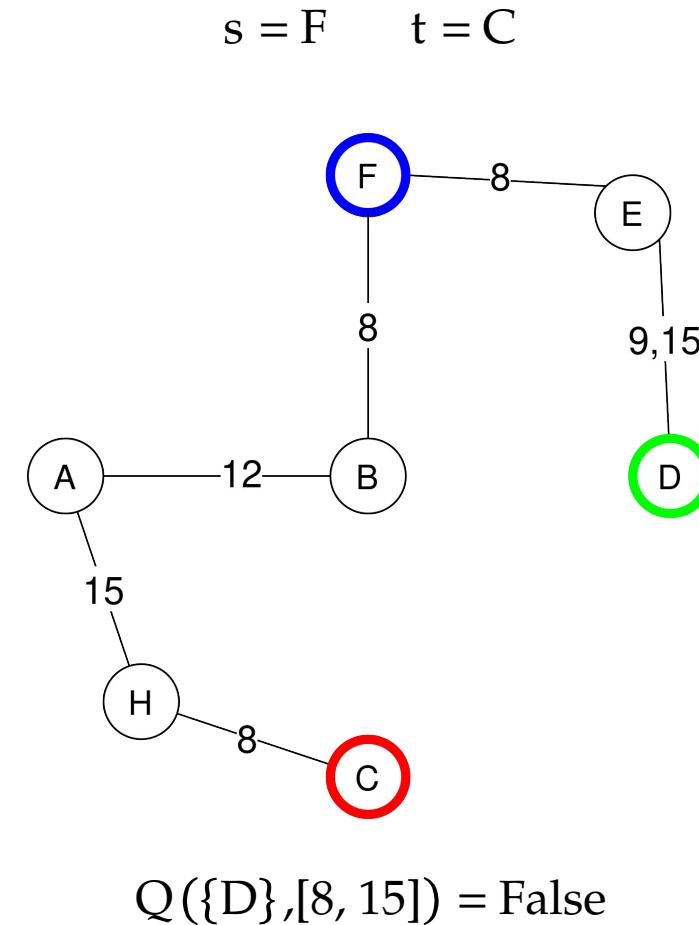
k-waypoint routing with window: Oracolo

❖ Input:

- Grafo temporale $G = (V, E, \lambda)$
- Una sorgente $s \in V$
- Una destinazione $t \in V$

❖ Output:

- Un'oracolo che sia in grado di rispondere a query del tipo:
 - Dato un'insieme $K = \{x_1, \dots, x_k\} \subseteq V$ con $|K| = k$
 - Dato un'intervallo $[\alpha, \beta] \in \mathbb{N}$
 - Determinare se $\exists \pi(s, t | x_1, \dots, x_k)$ tale che ogni arco viene attraversato ad un istante $t \in [\alpha, \beta]$



k-waypoint routing with window: Risultati

	K-waypoint routing		K-waypoint routing with window
	Oracolo	Spanner	Oracolo
Parametro	$k = 1$		
Building Time	$O(n + M)$	$O(n + M)$	$O(n + M)$
Dimensione	$O(n)$	$O(n)$	$O(n + M)$
Query time	$O(1)$		$O(\log M)$
Parametro	$k \geq 2$		
Dimensione		$\Omega(n^2)$	

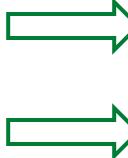
Soluzione per k-waypoint routing: oracolo per $k = 1$

Spiegazione della procedura utilizzata per la risoluzione del problema

k-waypoint routing: Costruzione dell'oracolo per $k = 1$

❖ Costruzione dell'oracolo:

- Si calcola un vettore ea_s contenente tutti gli earliest arrival time da s verso ogni nodo $v \in V$ in tempo $O(n + M)$
- Si calcola un vettore ld_t contenente tutti i latest departure time da ogni nodo $v \in V$ verso t in tempo $O(n + M)$
- Si uniscono i due vettori di grandezza $O(n)$ in modo da comporre un oracolo contenente per ogni $v \in V$ i rispettivi valori di $ea_s[v]$ e $ld_t[v]$



Parametro	$k = 1$
Building Time	$O(n + M)$
Dimensione	$O(n)$
Query time	$O(1)$

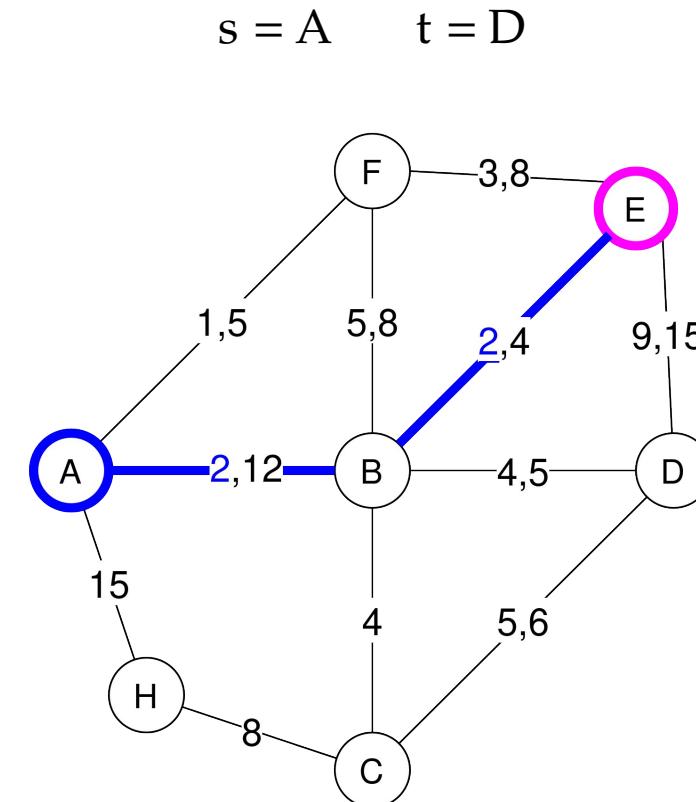
k-waypoint routing: Lemma

❖ **Lemma:**

➢ Siano $G = (V, E, \lambda)$ e $s, t, x \in V$

➢ Allora:

$$ea_s[x] \leq ld_t[x] \Leftrightarrow \exists \pi(s, t | x)$$



$$ea_A[E] = 2$$

k-waypoint routing: Lemma

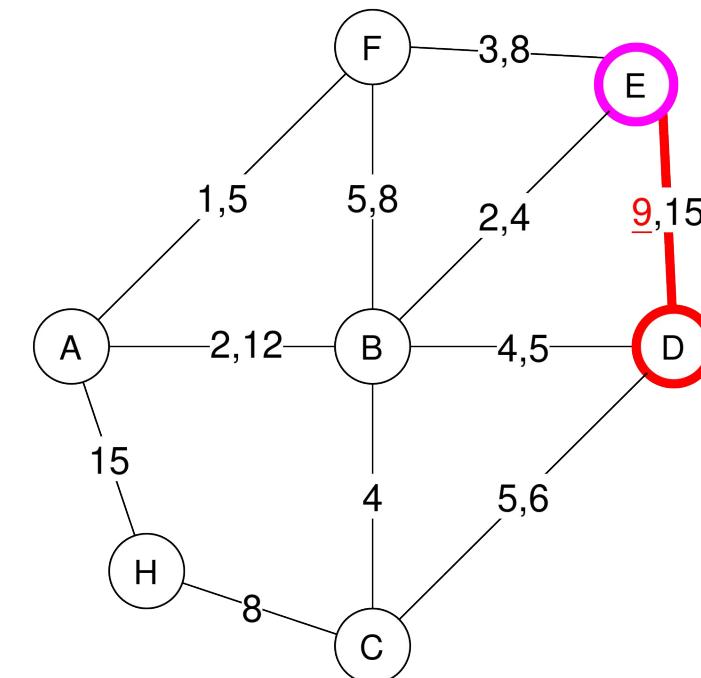
❖ **Lemma:**

➢ Siano $G = (V, E, \lambda)$ e $s, t, x \in V$

➢ Allora:

$$ea_s[x] \leq ld_t[x] \Leftrightarrow \exists \pi(s, t | x)$$

$$s = A \quad t = D$$



$$ld_D[E] = 9$$

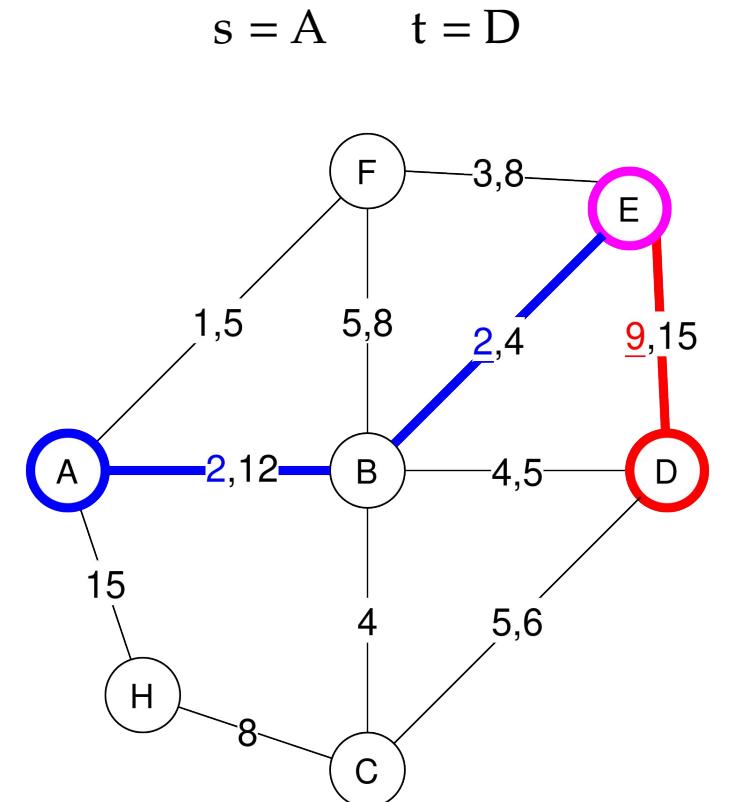
k-waypoint routing: Lemma

❖ **Lemma:**

➢ Siano $G = (V, E, \lambda)$ e $s, t, x \in V$

➢ Allora:

$$ea_s[x] \leq ld_t[x] \Leftrightarrow \exists \pi(s, t | x)$$



$$ea_A[E] \leq ld_D[E] \Rightarrow \exists \pi(A, D | E)$$

k-waypoint routing: Query dell'oracolo per k = 1

❖ Esecuzione della query:

- Vengono estratti i valori di $ea(s,x)$ e $ld(x,t)$ contenuti all'interno dell'oracolo
- Verrà effettuato il controllo $ea(s,x) \leq ld(x,t)$ e l'oracolo risponderà:
 - *True* se la relazione sarà verificata
 - *False* altrimenti



Parametro	$k = 1$
Building Time	$O(n + M)$
Dimensione	$O(n)$
Query time	$O(1)$

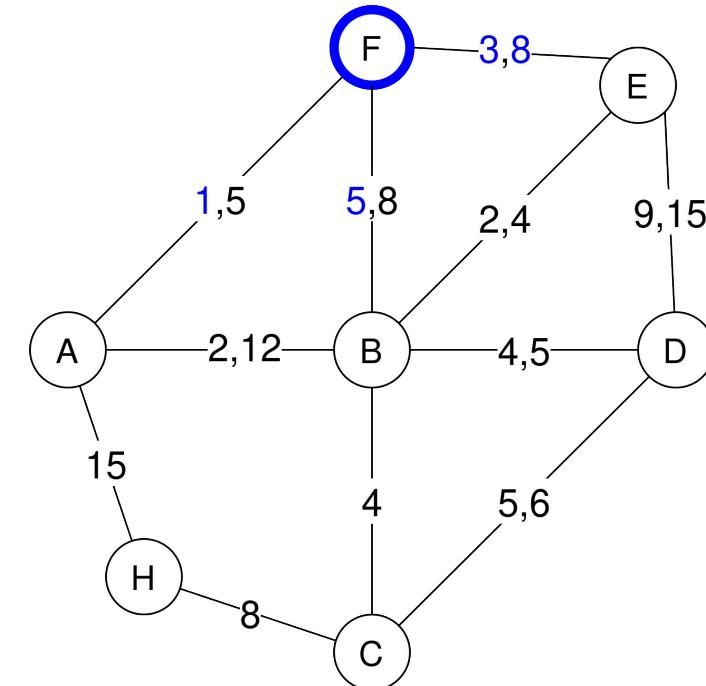
Soluzione per k-w. routing with window: oracolo per $k = 1$

Spiegazione della procedura utilizzata per la risoluzione del problema

k-waypoint routing with w.: Definizione oracolo per k = 1

- ❖ **Definizione dell'oracolo:**

- ❖ Sia $\Theta = \{\theta_1, \dots, \theta_{\Delta_s}\}$ l'insieme disgiunto dei timestamp associati agli archi uscenti da s in ordine crescente
- ❖ Sia $\Phi = \{\phi_1, \dots, \phi_{\Delta_t}\}$ l'insieme disgiunto dei timestamp associati agli archi entranti in t in ordine decrescente
- ❖ L'oracolo sarà costituito da due strutture dati:
 - ❖ Una struttura dati EA contenente $\forall v \in V$ tutti gli earliest arrival time verso v di cammini partiti da s con tempo di partenza minimo $\theta_j \in \Theta$
 - ❖ Una struttura dati LD contenente $\forall v \in V$ tutti gli latest departure time di cammini partiti da v con tempo di arrivo massimo a $t \phi_j \in \Phi$

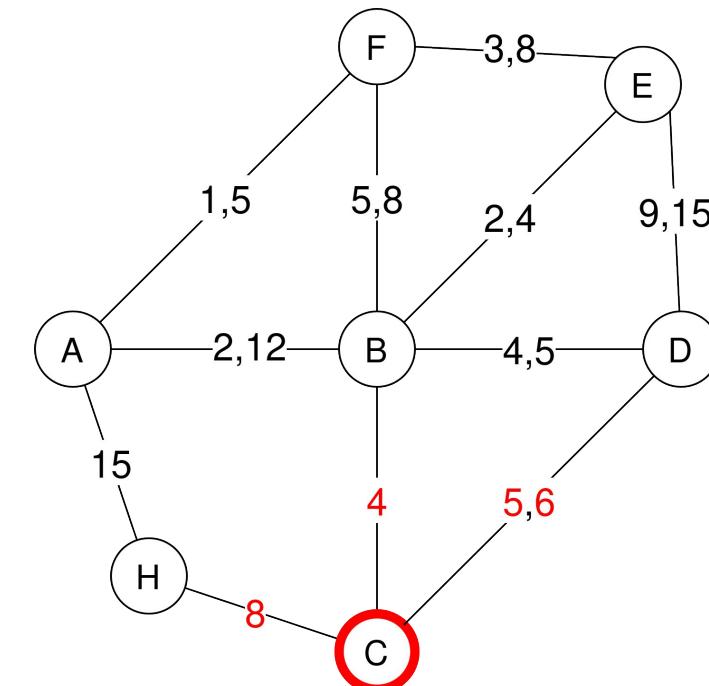


$$\Theta = \{1, 3, 5, 8\}$$

k-waypoint routing with w.: Definizione oracolo per k = 1

- ❖ **Definizione dell'oracolo:**

- ❖ Sia $\Theta = \{\theta_1, \dots, \theta_{\Delta_s}\}$ l'insieme disgiunto dei timestamp associati agli archi uscenti da s in ordine crescente
- ❖ Sia $\Phi = \{\phi_1, \dots, \phi_{\Delta_t}\}$ l'insieme disgiunto dei timestamp associati agli archi entranti in t in ordine decrescente
- ❖ L'oracolo sarà costituito da due strutture dati:
 - ❖ Una struttura dati EA contenente $\forall v \in V$ tutti gli earliest arrival time verso v di cammini partiti da s con tempo di partenza minimo $\theta_j \in \Theta$
 - ❖ Una struttura dati LD contenente $\forall v \in V$ tutti gli latest departure time di cammini partiti da v con tempo di arrivo massimo a $t \phi_j \in \Phi$

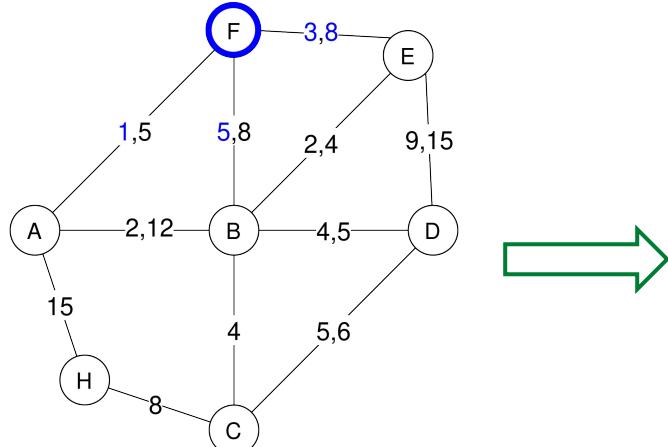


$$\Phi = \{8, 6, 5, 4\}$$

k-waypoint routing with w.: Definizione oracolo per k = 1

❖ Definizione di EA:

- ❖ Si definisce la matrice $EA \in \mathbb{N}^{n \times \Delta_s}$ dove:
 - ❖ Ogni colonna corrisponde ad un nodo $v \in V$
 - ❖ Ogni riga è associata ad un tempo di partenza $\theta_j \in \Theta$ da s
 - ❖ $EA[v, \theta_j] = \text{tempo minimo di arrivo al nodo } v \text{ partendo da } s \text{ in un istante } t' \geq \theta_j$



→

	A	B	C	D	E	F	H
$\theta_1 = 1$	1	2	4	4	3	1	8
$\theta_2 = 3$	5	4	4	4	3	3	8
$\theta_3 = 5$	5	5	5	5	8	5	8
$\theta_4 = 8$	12	8	$+\infty$	9	8	8	15

k-waypoint routing with w.: Definizione oracolo per k = 1

❖ Definizione di LD:

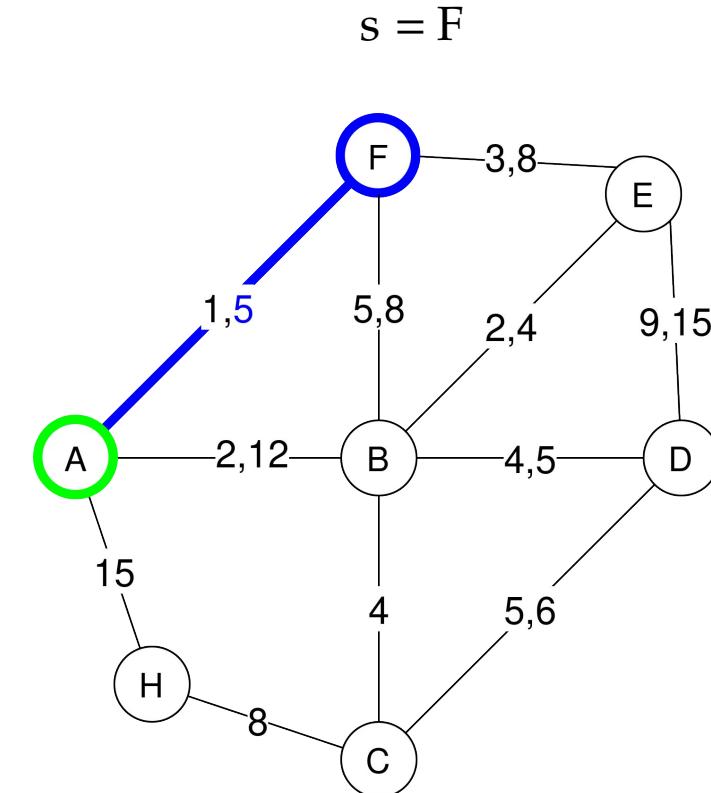
- ❖ Si definisce la matrice $LD \in \mathbb{N}^{n \times \Delta_t}$ dove:
 - ❖ Ogni colonna corrisponde ad un nodo $v \in V$
 - ❖ Ogni riga è associata ad un tempo di arrivo $\phi_j \in \Phi$ da t
 - ❖ $LD[v, \phi_j] = \text{tempo massimo di partenza da } v \text{ consentendo di arrivare a } t \text{ in un istante } t' \leq \phi_j$

k-waypoint routing with window: Oracolo

❖ Lemma:

- Sia EA la struttura dati appena definita e sia $x \in V$
- Vale la seguente proprietà di monotonicità:

$$\forall \theta_j \leq \theta_i \quad EA[\theta_j, x] \leq EA[\theta_i, x]$$



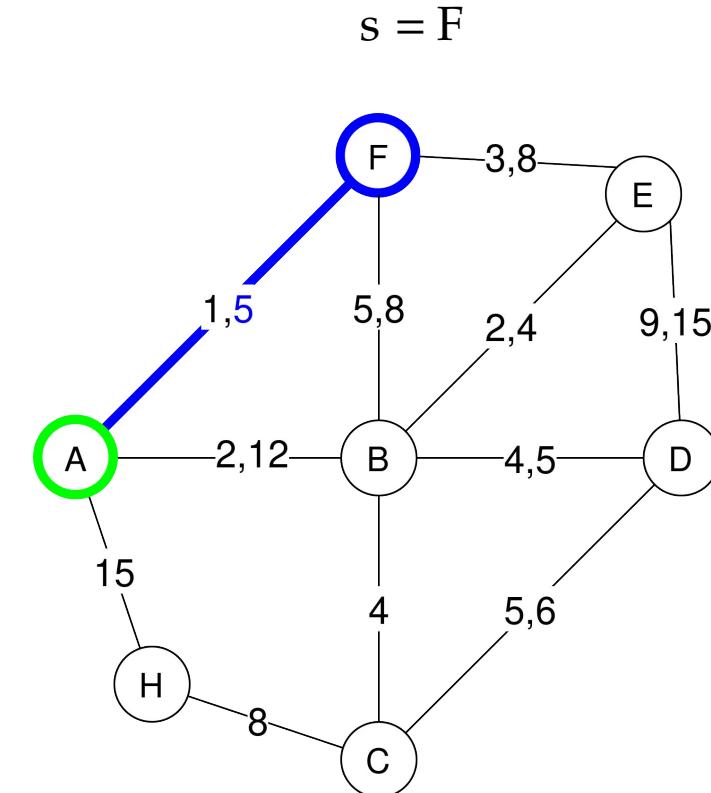
$$EA[5, A] = 5$$

k-waypoint routing with window: Oracolo

❖ Lemma:

- Sia EA la struttura dati appena definita e sia $x \in V$
- Vale la seguente proprietà di monotonicità:

$$\forall \theta_j \leq \theta_i \quad EA[\theta_j, x] \leq EA[\theta_i, x]$$



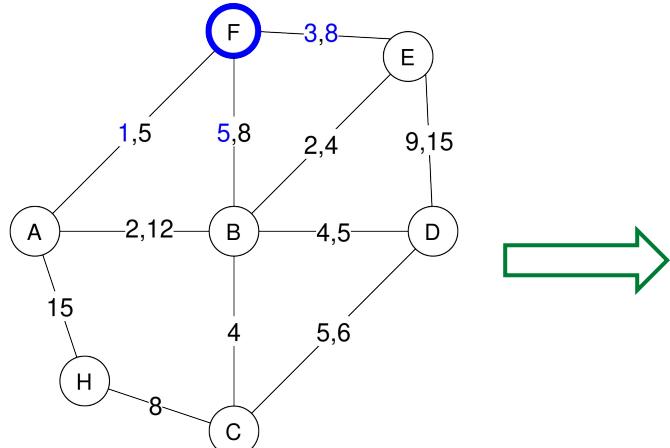
$$EA[3,A] = 5$$

k-waypoint routing with window: Ottimizzazione oracolo

❖ Ottimizzazione dell'oracolo:

- ❖ EA presenta molti valori ripetuti
- ❖ Sfruttando il lemma precedente è possibile ridefinire EA come un vettore dove ad ogni nodo $v \in V$ venga associata una pila di coppie $(\theta_i, ea_{\theta_i}[v])$ tali che:

$$\forall j < i, \quad ea_{\theta_j}[v] \neq ea_{\theta_i}[v]$$



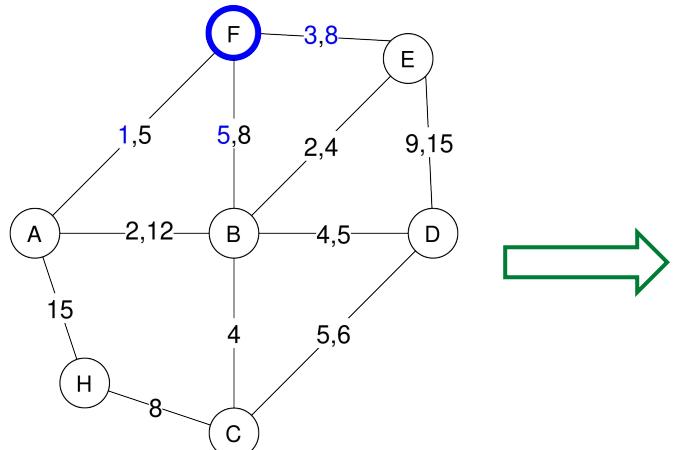
	A	B	C	D	E	F	H
$\theta_1 = 1$	1	2	4	4	3	1	8
$\theta_2 = 3$	5	4	4	4	3	3	8
$\theta_3 = 5$	5	5	5	5	8	5	8
$\theta_4 = 8$	12	8	$+\infty$	9	8	8	15

k-waypoint routing with window: Ottimizzazione oracolo

❖ Ottimizzazione dell'oracolo:

- ❖ EA presenta molti valori ripetuti
- ❖ Sfruttando il lemma precedente è possibile ridefinire EA come un vettore dove ad ogni nodo $v \in V$ venga associata una pila di coppie $(\theta_i, ea_{\theta_i}[v])$ tali che:

$$\forall j < i, \quad ea_{\theta_j}[v] \neq ea_{\theta_i}[v]$$



A	B	C	D	E	F	G
(1,1)	(1,2)	(3,4)	(3,4)	(3,3)	(1,1)	(5,8)
(5,5)	(3,4)	(5,5)	(5,5)	(8,8)	(3,3)	(8,15)
(8,12)	(5,5)		(8,9)		(5,5)	
	(8,8)				(8,8)	

k-waypoint routing with window: Ottimizzazione oracolo

❖ Analisi della dimensione del nuovo oracolo:

- È stato dimostrato che $\forall v \in V, |EA[v]| \leq t_{\deg}(v)$
- Di conseguenza $\sum_{v \in V} |EA[v]| \leq \sum_{v \in V} t_{\deg}(v) = O(M)$
- Per cui è possibile dire che la dimensione dell'oracolo sia $O(n + M)$



Parametro	$k = 1$
Building Time	$O(n + M)$
Dimensione	$O(n + M)$
Query time	$O(\log M)$

k-waypoint routing with window: Creazione dell'oracolo

❖ Costruzione dell'oracolo:

- Si calcola un vettore di pile EA come definito precedentemente in tempo $O(n + M)$
- Si calcola un vettore di pile LD come definito precedentemente in tempo $O(n + M)$



Parametro	$k = 1$
Building Time	$O(n + M)$
Dimensione	$O(n)$
Query time	$O(1)$

k-waypoint routing with window: Query dell'oracolo

❖ Esecuzione della query:

- Viene estratta la tupla $(\theta_\alpha, ea_\alpha[x])$ dal vettore EA[x] dove θ_α è il più piccolo $\theta_\alpha \geq \alpha$ tramite ricerca binaria
- Viene estratta la tupla $(\phi_\beta, ld_\beta[x])$ dal vettore LD[x] dove ϕ_β è il più grande $\phi_\beta \leq \beta$ tramite ricerca binaria
- Verrà effettuato il controllo $ea_\alpha[x] \leq ld_\beta[x]$ e l'oracolo risponderà:
 - *True* se la relazione sarà verificata
 - *False* altrimenti



Parametro	$k = 1$
Building Time	$O(n + M)$
Dimensione	$O(n + M)$
Query time	$O(\log M)$

Conclusioni

❖ Problemi aperti:

- Applicare il lowerbound trovato sulla dimensione dello spanner per $k \geq 2$ anche sulla variante riguardante l'oracolo
- Trovare un upper bound per la variante del problema utilizzante l'oracolo con il parametro $k \geq 2$



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA