Problema di minimizzazione

- ▷ Problema: P
- ▷ Insieme delle soluzioni ammissibili: SA
- ▷ Funzione costo $C : SA \to \mathbb{R}$
- ▷ Trovare $s_o \in SA$ tale che

$$C(s_o) = \min_{s \in SA}\{C(s)\}$$

Problema di massimizzazione

- ▷ Problema: P
- ▷ Insieme delle soluzioni ammissibili: SA
- ▷ Funzione ricavo $R : SA \to \mathbb{R}$
- ▷ Trovare $s_o \in SA$ tale che

$$R(s_o) = \max_{s \in SA}\{R(s)\}$$

Algoritmi di approssimazione

Minimizzazione

Sia P un problema di minimizzazione. A è un algoritmo rho $\rho > 1$ approssimato per P se e solo se per ogni istanza I di P

$$C(A(I)) \leq \rho \cdot C(soluzione\ ottima)$$

COSTO SOLUZIONE MASSIMO ρ VOLTE PEGGIORE OTTIMO

Massimizzazione

Sia P un problema di minimizzazione. A è un algoritmo $0 < \rho < 1$ approssimato per P se e solo se per ogni istanza I di P

$$R(A(I)) \geq \rho \cdot R(soluzione\ ottima)$$

RICAVO SOLUZIONE MINIMO ρ VOLTE MIGLIORE OTTIMO

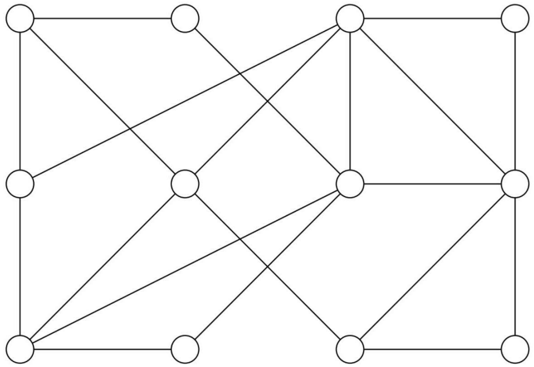
Vertex Covering VC

Vertex Covering

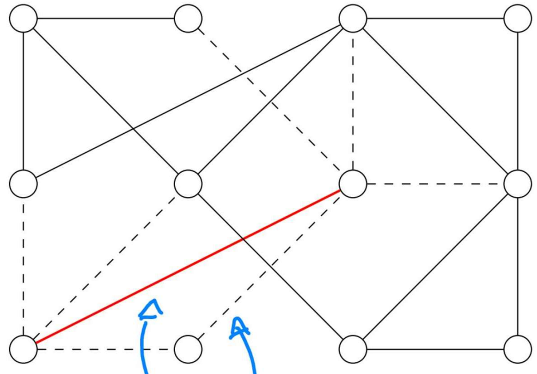
INPUT: un grafo $G = (V, E)$ non orientato E NON PESATO
OUTPUT: $V' \subseteq V$ tale che
V' copre tutti gli archi di E (ammissibilità) e
V' è di cardinalità minima (ottimalità)

ANALOGO: EDGE COVERING

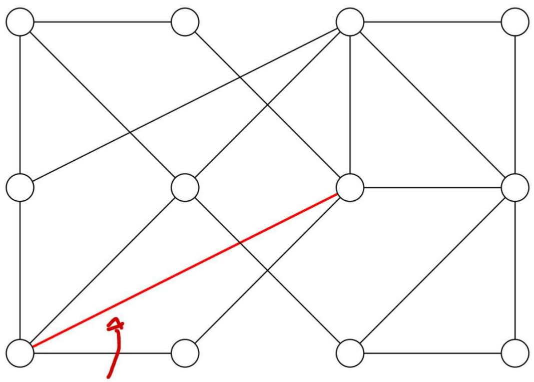
VC: esempio



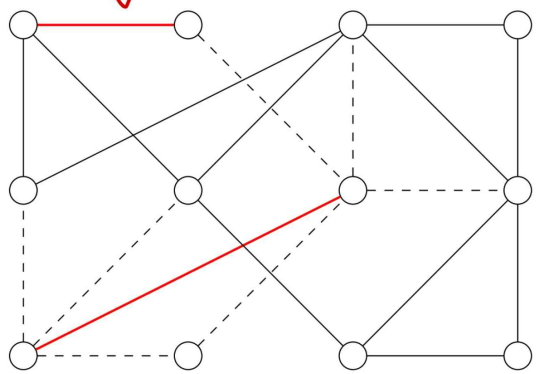
VC: esempio



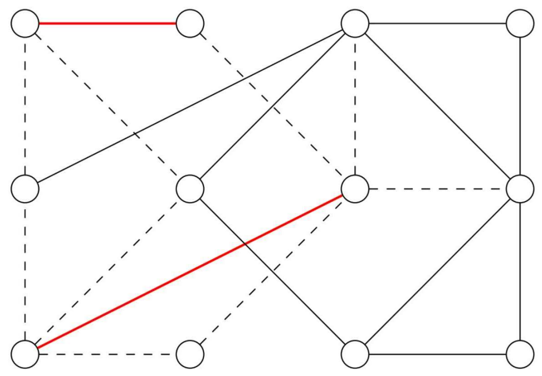
VC: esempio



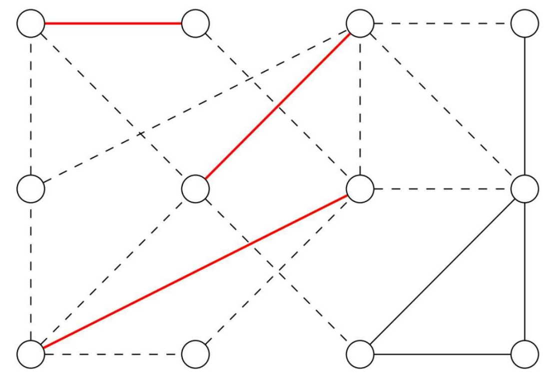
VC: esempio



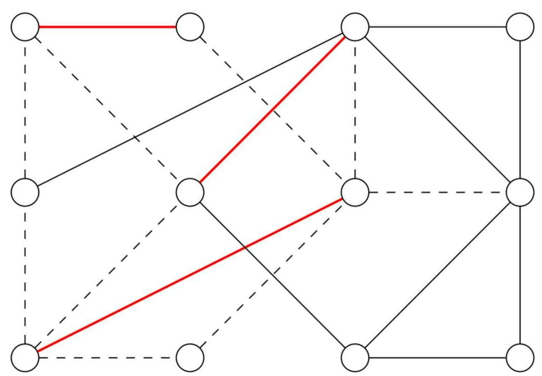
VC: esempio



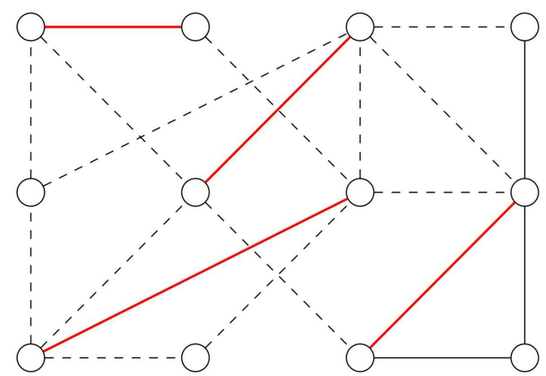
VC: esempio



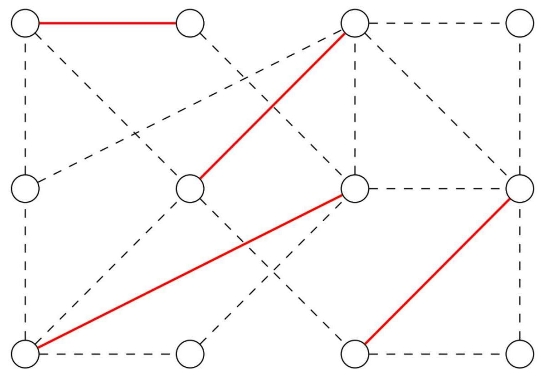
VC: esempio



VC: esempio



VC: esempio

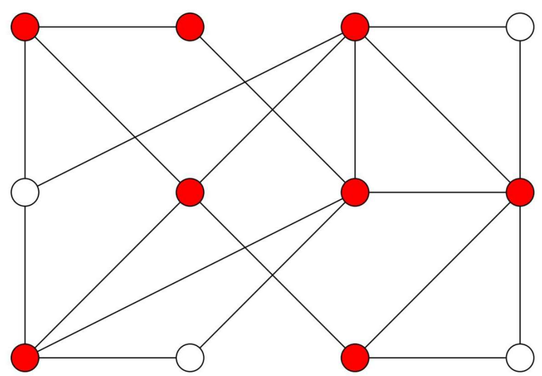


VC: algoritmo 2 approssimato

```
APPROX-VERTEX-COVER(G)
1  C = ∅
2  E' = G.E
3  while E' ≠ ∅
4      let (u, v) be an arbitrary edge of G.E
5      C = C ∪ {u, v}
6      remove from E' every edge
       incident on either u or v
7  return C
```

Costo polinomiale

VC: esempio



VC: algoritmo 2 approssimato

- Siano $(u_1, u_2), (u_3, u_4), \ldots, (u_{2k-1}, u_{2k})$ i k archi selezionati al passo 4 dell'algoritmo
- Non è difficile dimostrare che $i \neq j \implies u_i \neq u_j$ NO NODI COMUNI
- L'arco (u_i, u_{i+1}) può essere coperto solo da u_i o da u_{i+1}
- La soluzione ottima deve contenere almeno k nodi
- La soluzione restituita dall'algoritmo è 2 approssimata

O-TSP

COMMESSO-VIAGGIATORE EUCLIDEO

PUNTI SU PIANO

⇒ PESI = DISTANZA

TUTTI CON TUTTI

O-TSP

INPUT: un grafo $G = (V, E)$ non orientato completo con pesi sugli archi non negativi

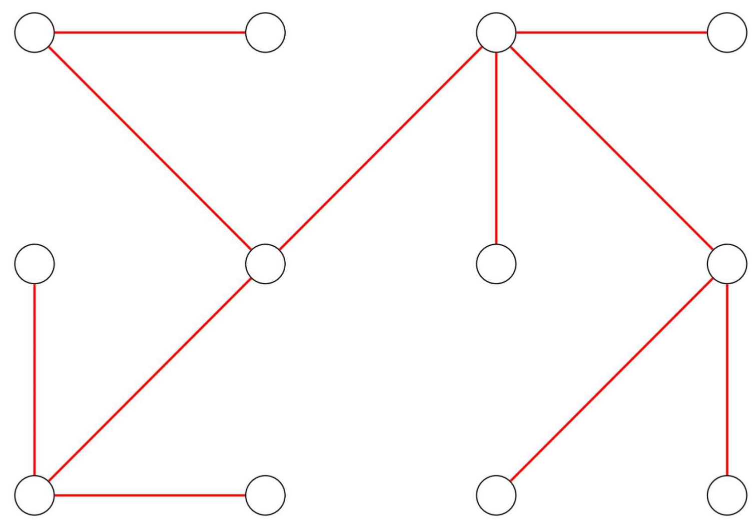
OUTPUT: $T \subseteq E$ tale che

T è un ciclo di Hamilton che tocchi tutti i nodi di G (ammissibilità) e VISITE NODI NON RIPETUTE

T ha costo minimo (ottimalità)

Algoritmo 2 approssimato per DT-O-TSP: esempio

MST



TSP con disuguaglianza triangolare

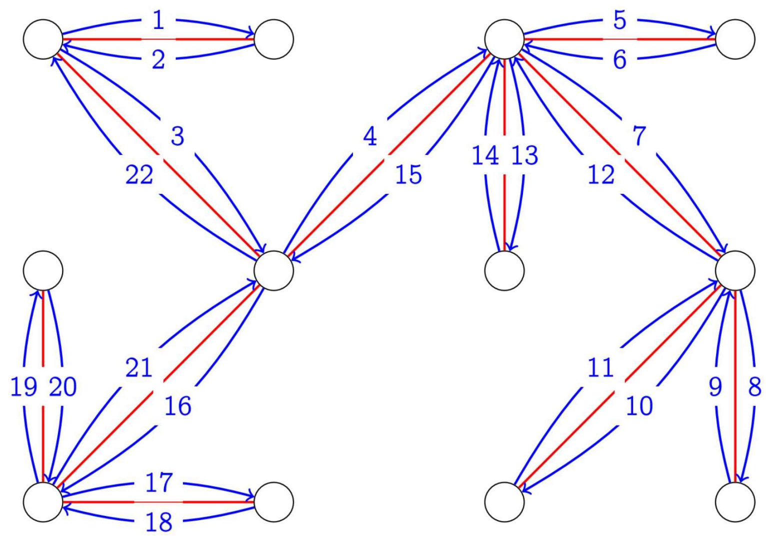
DT-O-TSP

DT-O-TSP è definito come O-TSP con un vincolo aggiuntivo: per ogni terna (v_i, v_j, v_k) di nodi del grafo abbiamo

$$w(v_i, v_j) + w(v_j, v_k) \leq w(v_i, v_k)$$

Algoritmo 2 approssimato per DT-O-TSP: esempio

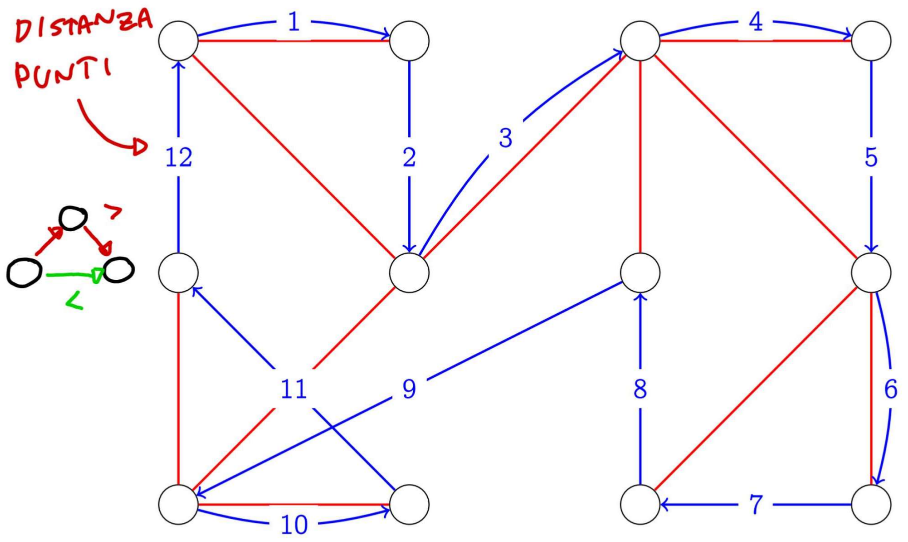
CICLO CON CITTÁ RIPETUTE



Algoritmo 2 approssimato per DT-O-TSP: esempio

RIMOSSE RIPETIZIONI

DISTANZA PUNTI



DT-O-TSP: algoritmo 2 approssimato

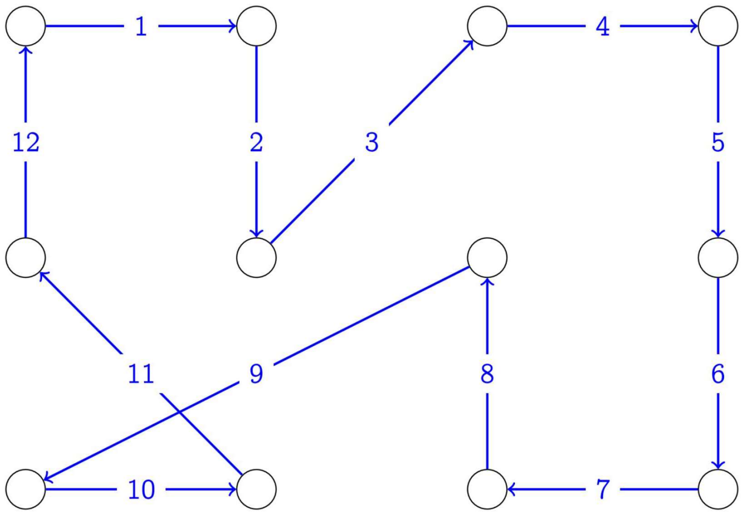
DT-O-TSP(G)

mst = $MST(G)$
$doublemst$ = contorno di mst
sol = elimina nodi ripetuti in $doublemst$
return sol

Costo polinomiale

Algoritmo 2 approssimato per DT-O-TSP: esempio

AMMISSIBILE



DT-O-TSP

- ▷ mst = albero di copertura minimo (archi rossi)
- ▷ sol = soluzione prodotta dall'algoritmo (blu)
- ▷ $doublemst$ = archi blu
- ▷ shp = spanning hamiltonian path TOCCA TUTTI MA NON CICLO (TOUR)
- ▷ opt = soluzione ottima
- ▷ opt^- = soluzione ottima a cui togliamo un arco qualsiasi

DT-O-TSP

- La proprietà della disuguaglianza triangolare ci assicura che $C(sol) \leq C(doublemst)$
- $C(doublemst) = 2C(mst)$
- Uno spanning hamiltonian path shp è uno spanning tree quindi $\forall shp: \; C(mst) \leq C(shp)$
- $C(opt) \geq C(opt^-)$
- opt^- è un shp e quindi $C(opt^-) \geq C(mst)$

DT-O-TSP

$$
\begin{aligned}
C(sol) &\leq C(doublemst) \\
&= 2C(mst) \\
&\leq 2C(opt^-) \\
&\leq 2C(opt)
\end{aligned}
$$

NO DISUGUAGLIANZA TRIANGOLARE ⇒ DIMOSTRATO NO APPROSSIMAZIONE

PROBLEMA DIFFICILE ⇔ BASTA POCHISSIMO PER CAMBIARE MOLTO