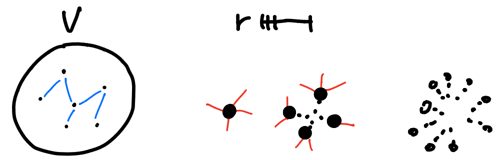
COMPONENTI CONNESSE

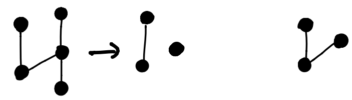


PARZIALE

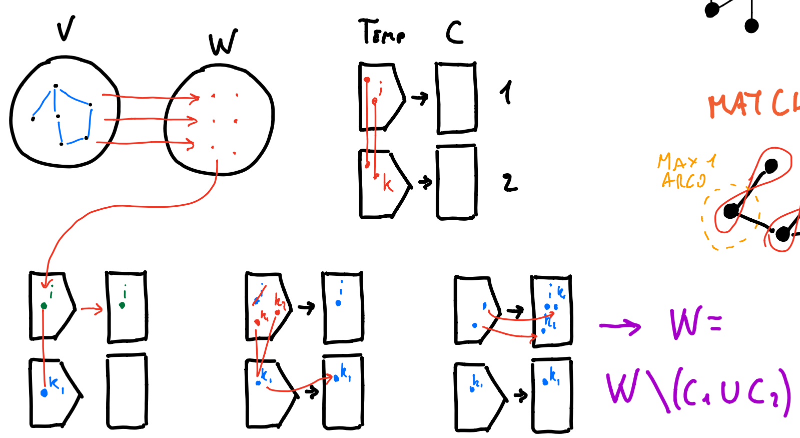
COMPLETO

SOTTOGRAFO

INDOTTO

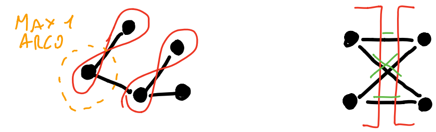


GRAFO BIPARTITO

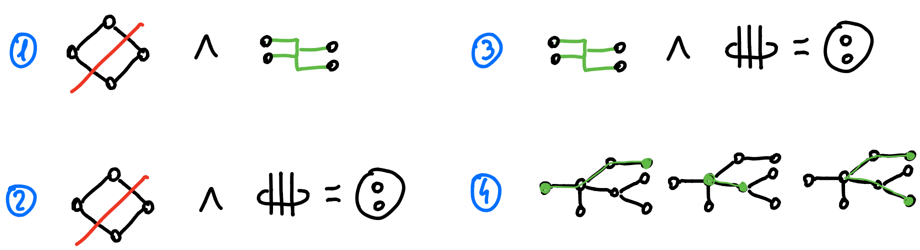


MATCHING

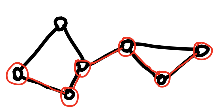
BIPARTITI



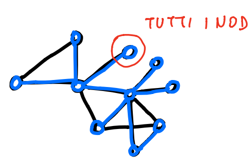
ALBERO



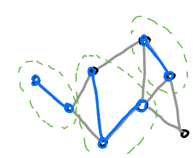
SHORTEST PATH



ALBERO DI SUPPORTO

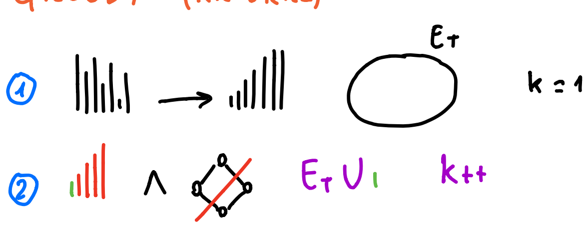


FORESTA



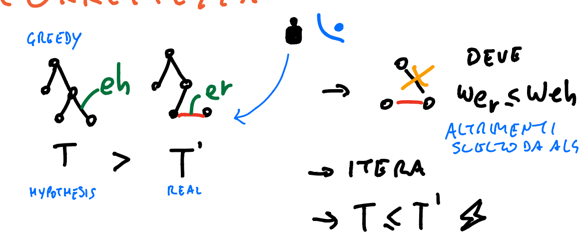
GREEDY (KRUSKAL)

$|E| \log(|E|)$

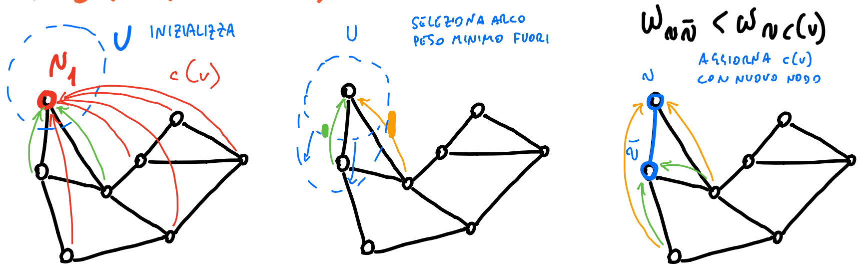


CORRETTEZZA

PER ASSURDO

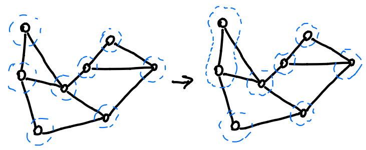


MST-1 $O(V^2)$

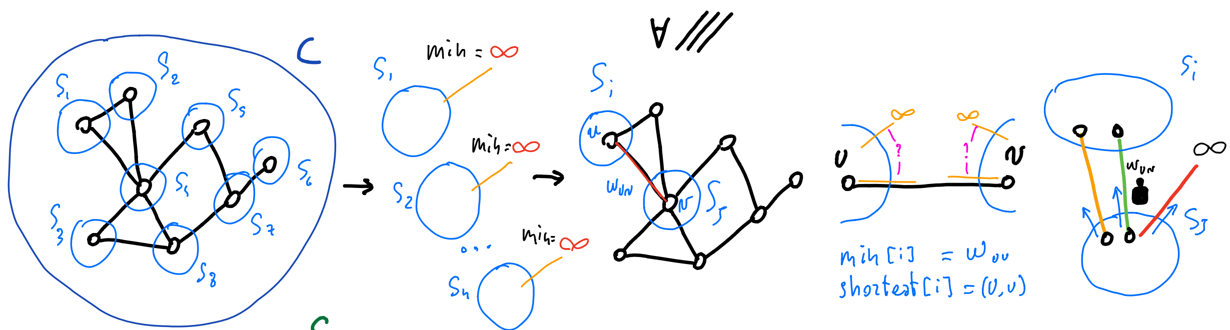


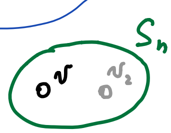
v	c(v)	wc(v)
2	v_1	3
3	v_1	5
...	v_1	...

CLOSEST VERTEX — NODO IN U PIÙ VICINO
WEIGHT OF... — PESO



MST-2 (BORŮVKA) $|E| \log(|V|)$





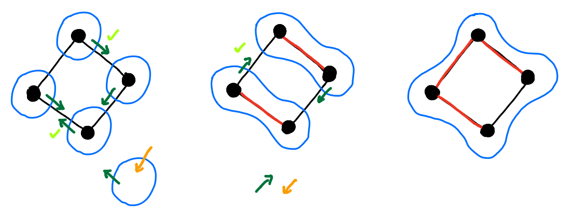
VALUTA TUTTI ARCHI

1	2	3	4	5
-	4	3	5	
	-	2	1	
		-	3	

→

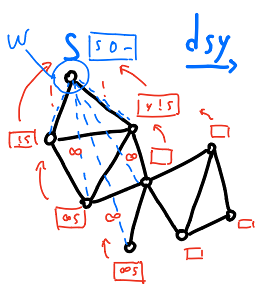
COMP	MIN	SHTST
S_1	~~∞~~ 3	~~/~~ (1,3)
⋮	⋮	⋮

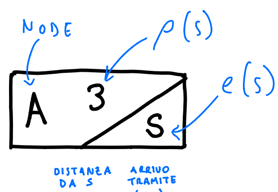
→ INSERISCI IN E_T TUTTI ARCHI SHORTEST → RICOSTRUISCI INSIEMI



SHORTEST PATH

DIJKSTRA $O(n^2)$



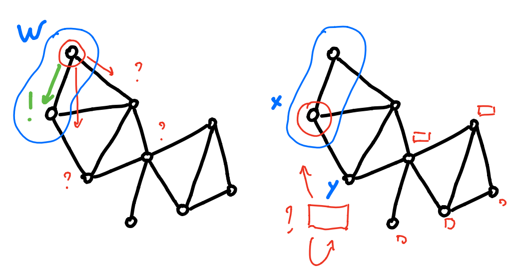


v	ρ	e
1		
2		
...		

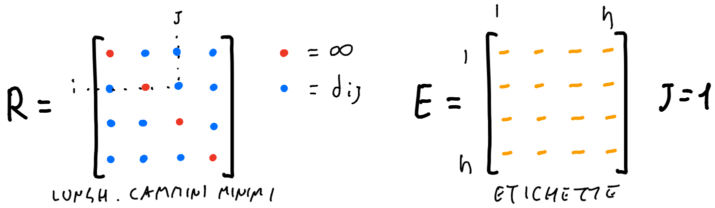
W
1
...

ARGMIN ρ → W
AGGIUNGI QUELLO MINIMO

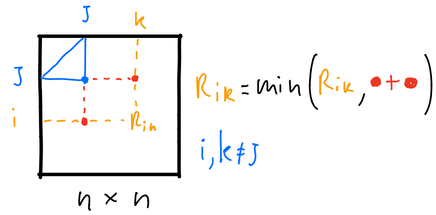
$\rho = \min(\rho(y), \rho(x) + d_{xy})$
AGGIORNA DISTANZE

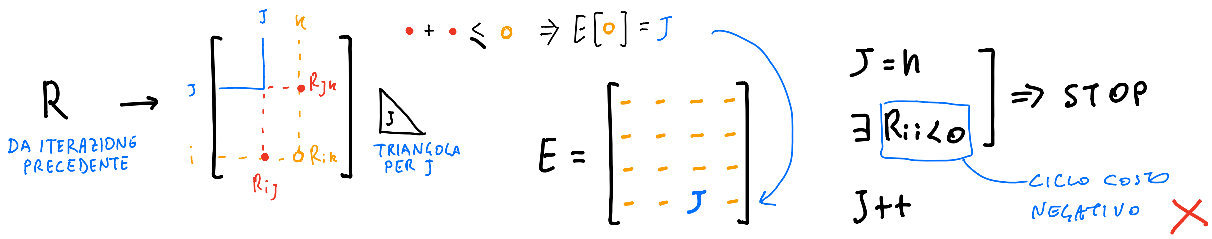


FLOYD-WARSHALL $O(n^3)$



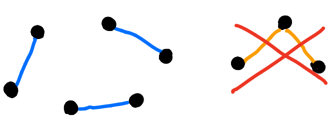
TRIANGOLAZIONE



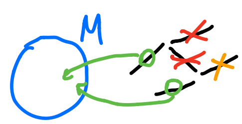


$R(\text{START}, \text{END}) = \text{DIST}$ $\qquad$ $E(\text{START}, \text{END}) = \text{NEXT}$

MATCHING



M. INIZIALE



∄ NON ADIACENTE A $\forall e \in M$

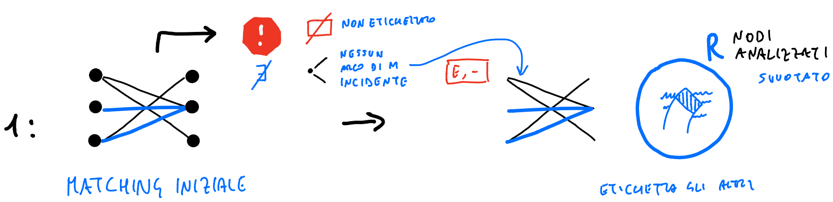
$M = M \cup \{\bar{e}\}$

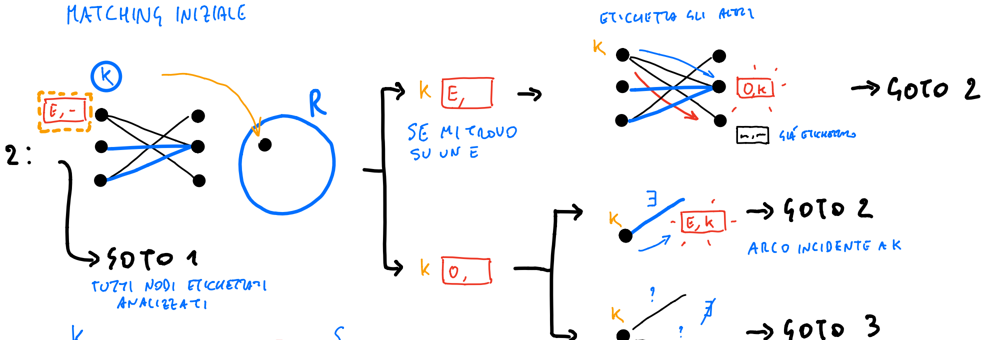
MATCHING CARDINALITÀ MAX $O(\min(|V_1|, |V_2|)\,|A|)$

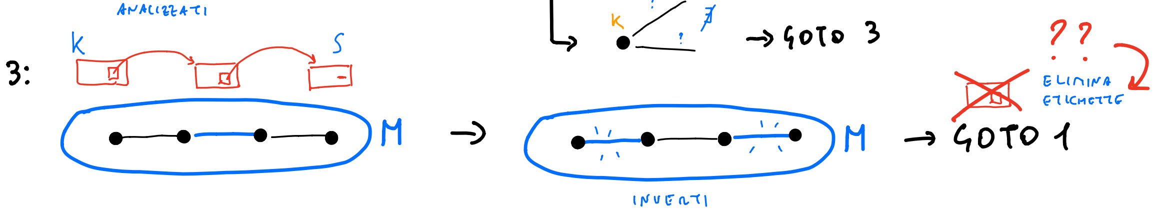
PESI EQUIVALENTI

$W_e = 1$

$\forall e \in E$





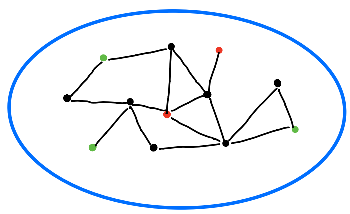


FLUSSO A COSTO MINIMO

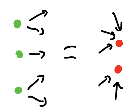
$G = (V, A)$

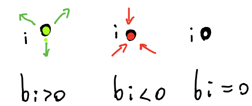
c_{ij} = COSTO $

b_i = GENERAZIONE

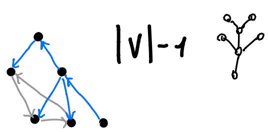


$\sum i = 0$

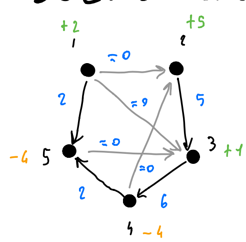




BASE → SOL. DI BASE



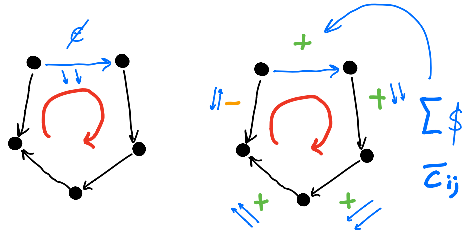
$|V| - 1$



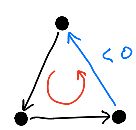
- > 0 → AMMISSIBILE
- $\forall < 0$ → DEGENERE
- → COEFF. COSTO RIDOTTO

FLUSSO +1
- → SE APPLICO, COSTA +
- → SE APPLICO, COSTA −

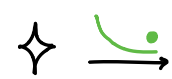
CALCOLO COEFFICIENTI



ILLIMITATEZZA



OTTIMALITÁ



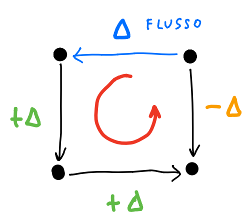
$\forall \bar{c} \geq 0$

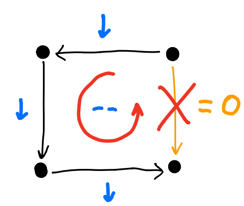
ARCHI FUORI BASE

ARCO ENTRANTE → USCENTE

< 0 $\Delta =$

$\bar{\Delta} = \downarrow\downarrow$



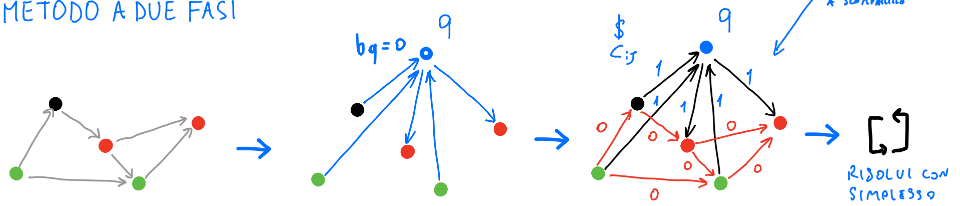


SIMPLESSO

- VERIFICA OTTIMALITÁ →
- VERIFICA ILLIMITATEZZA →
- CAMBIO BASE

BASE AMMISSIBILE

METODO A DUE FASI



VALORE OTTIMO	REGIONE AMMISSIBILE
$S > 0$	$\emptyset$
$S = 0$	$\neg \emptyset$ → ∃ albero !!! ...

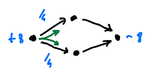
CAPACITÁ LIMITATE

d_{ij} = CAPACITÁ MASSIMA

STATI

- B IN BASE — FLUSSO ARBITRARIO
- N_0 FUORI BASE FLUSSO NULLO (0)
- N_1 FUORI BASE MAX CAPACITÁ ($\overline{}$)

LA BASE DEVE RIMANERE UN ALBERO



COSTO UNITARIO / CAPACITÁ MASSIMA

c_{ij} / d_{ij}

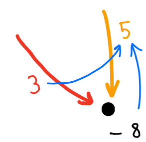
SOLUZIONE DI BASE

N_0: $x = 0$

N_1: $x = \overline{}$

$\Rightarrow$ x_{ij} B ! TROVA DI CONSEGUENZA

DERIVA DA DESTINAZIONE E N_0, N_1



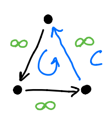
OTTIMALITÁ

N_0 $\bar{c}_{ij} \geq 0$ → \$ ↑ NON CONVIENE USARLI MAI RISPETTO AD ALTRI

N_1 $\bar{c}_{ij} \leq 0$ → \$ ↑ CI PIACEREBBE USARLI DI PIÚ

COEFF. COSTO RIDOTTO

ILLIMITATEZZA



$c_{ij} < 0$

AMMISSIBILE

x_{ij} $< d_{ij}$

DEGENERE

∃ x_{ij} $= 0$

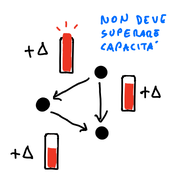
ARCHI IN BASE FLUSSO NULLO

ARCO ENTRANTE → USCENTE

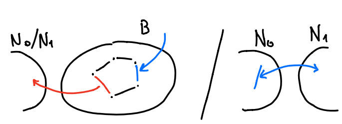
N_0 — $\bar{c}$ PIÚ BASSO

N_1 — $\bar{c}$ PIÚ ELEVATO

PIÚ PROMETTENTE

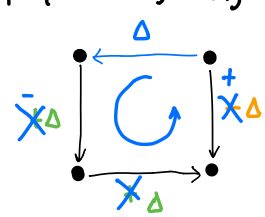


min (Δ)



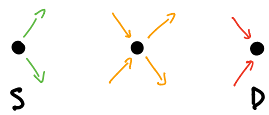
$N_0 \rightarrow \Delta$

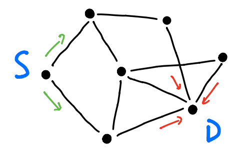
$N_1 \rightarrow d_{ij} = d_{ij} - \Delta$



REGOLE INVERSE

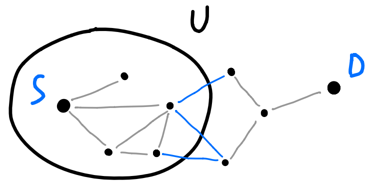
FLUSSO MASSIMO





MAX(FLOW)

TAGLIO



MASSIMO FLUSSO CON VINCOLI

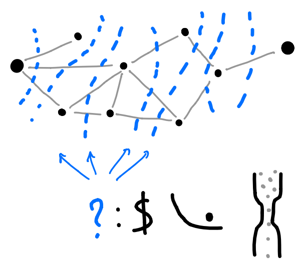
$$C(S_U) = \sum_U c_{ij}$$

COSTO TAGLIO — SOMMA CAPACITÀ

MAXFLOW = TAGLIO $ MIN

$$\text{MAX FLOW} \leq \forall\, C(S_U)$$

COSTO TAGLIO



TROVA IL COLLO DI BOTTIGLIA

FLUSSO AMMISSIBILE

$$\bar{X} = (\bar{x}_{ij})_{(i,j)\in A}$$

TUTTI I FLUSSI NEGLI ARCHI

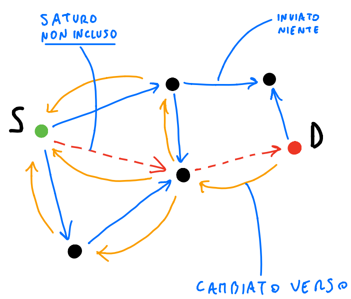
ARCHI SATURI

FLUSSO CAPACITÀ

$$x_{ij} = c_{ij}$$

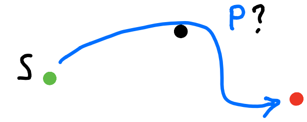
PERCORSO PRIVO DI ARCHI SATURI — NON OTTIMO — POSSO AUMENTARE IL FLUSSO

GRAFO ASSOCIATO AL FLUSSO

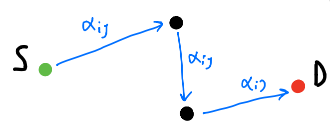


$A_f : \bar{x} < c$ — FORWARDS — NON SATURI

$A_b : \bar{x} > 0$ — BACKWARDS — INVIATO QUALCOSA

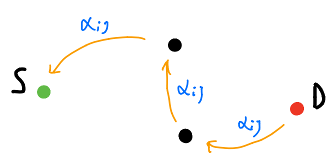


IPOTIZZO CAMMINO



$$\alpha_{ij} = c - \bar{x}$$

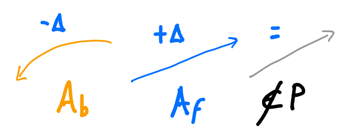
QUANTO POSSO ANCORA INVIARE



$$\alpha_{ij} = \bar{x}$$

QUANTO POSSO RISPEDIRE INDIETRO

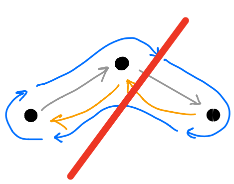
$$\Delta = \min_{(i,j)\in P} \alpha_{ij}$$



AGGIORNAMENTO FLUSSO

FLUSSO OTTIMO

$G(\bar{X})$ — GRAFO RELATIVO ALLA SOLUZIONE



NESSUN CAMMINO ORIENTATO

FORD-FULKERSON

① $\bar{x}!$ V $\bar{x}=0$

NOTO FLUSSO AMMISSIBILE — FLUSSO NULLO

② $\bar{x} \rightsquigarrow G(\bar{x})$

COSTRUIRE IL GRAFO

③ (∄ cammino) ? ($\bar{x}$ ✦) : ($\rightsquigarrow \pm\Delta$) → GOTO ①

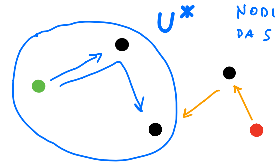
NON ESISTE CAMMINO ORIENTATO → x^* !

INDIVIDUA CAMMINO AGGIORNA FLUSSO

TAGLIO MINIMO

x^* DATA SOLUZIONE OTTIMA

U^* NODI RAGGIUNGIBILI DA S CON CAMMINO ORIENTATO



$G(x^*)$

ETICHETTATURA

PER STABILIRE IL CAMMINO

$G(\bar{x})$ ∃ S → … → D ?

⓪ $\bar{x}!$ V $\bar{x}=0$ FLUSSO NOTO O NULLO

① S [S,∞] — PARTI CON S ETICHETTATO

R ( ) NODI ANALIZZATI

E (·S) NODI ETICHETTATI

② $E \setminus R = \emptyset \Rightarrow \bar{x}$ ✦ !

| $i \in E \setminus R$ [K,Δ] i — ETICHETTATO NON ANALIZZATO

→ ANALISI

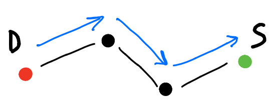
$\forall j \notin E : (i,j) \in A(\bar{x})$ (→ ORIENTATO!) — NODI NON ETICHETTATI CHE FORMANO ARCHI

j → A_f : [i, min Δ, c − $\bar{x}$]

j → A_b : [i, min Δ, $\bar{x}$]

→ R (j) E (i) → (D ∈ E) ? (→③) : (→②)

ETICHETTURA LA DESTINAZIONE

③  [!, Δ] → AGGIORNA $\bar{x}$ → ①

$S \to q_1 \to q_2 \to \dots \to D$!!

NODI ETICHETTATI E	NODI ANALIZZATI R
S	S
2	2
1	4
4	
D	

2 → ANALIZZATO 2 AGGIUNTE ETICHETTE 1, 4

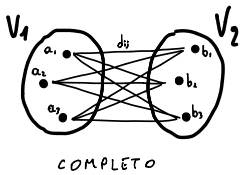
MOLTO PIÙ SEMPLICE DI QUELLO CHE SEMBRA

TEOREMA

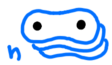
$U = E \to S_U$ ✦ TAGLIO MINIMO

≡ ✦ MAX FLOW

ASSEGNAMENTO COSTO MINIMO

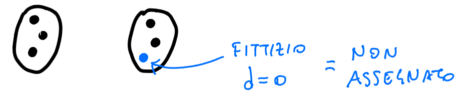


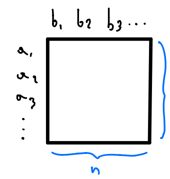
$d_{ij} = \$$



: $\sum d$

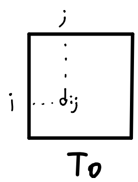
CARDINALITÁ DIVERSE



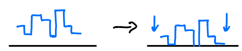


ALGORITMO UNGHERESE $O(n^3)$

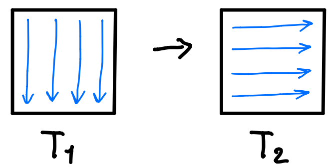
MATRICE COSTI → TRASFORMAZIONE



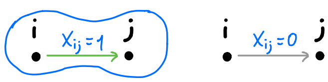
$d_j^0 = \min_i d_{ij}$



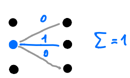
$d_{ij}^1 = d_{ij}^0 - d_j^0$



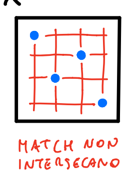
ASSEGNAMENTO



$\sum_{j \in V_2} x = 1 \quad \forall\, i \in V_1$



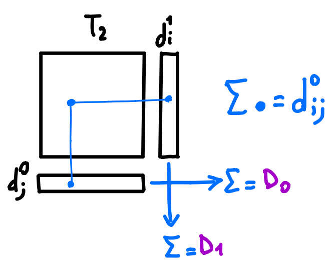
X

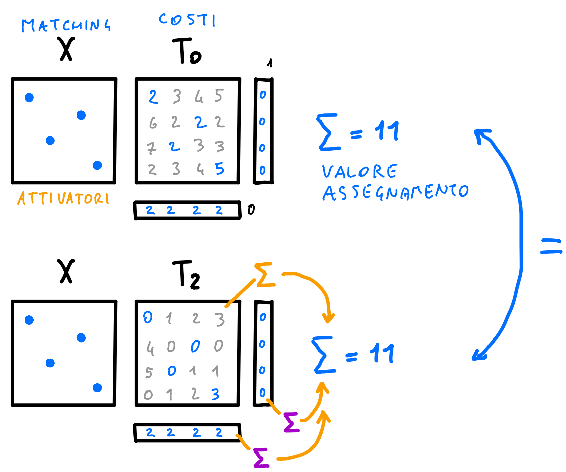


$\sum_i^n x_{ij} = 1 \quad \forall_j$

$\sum_j^n x_{ij} = 1 \quad \forall_i$

LOWER BOUND





$\sum d_{ij} x_{ij} \geq D_0 + D_1$

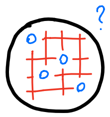
LOWER BOUND

↳ POSSIBILE SOLUZIONE OTTIMA → ESISTE MATCHING ?

PROBLEMA ASSOCIATO

T_2

0	1	2	3
4	0	0	0
5	0	1	1
0	1	2	3

$\Delta =$  ? $|\Delta| = ?$

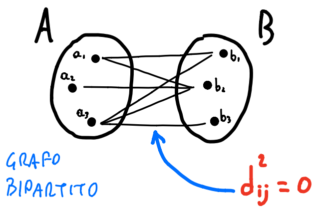
ZERI → COSTI PIÙ BASSI
INDIPENDENTI

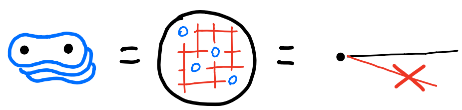
→

MATCHING

$\bar{x}_{ij} = \begin{cases} 1 & (i,j) \in \Delta \\ 0 \end{cases}$

→ $\$ = D_0 + D_1$ ⇒ ϟ





→ MATCHING CARD MAX → $\Delta = n$ ⇒ ! ϟ

↓

SE $|\Delta| < n$

T_2

0	1	2	3
4	0	0	0
5	0	1	1
0	1	2	3

→

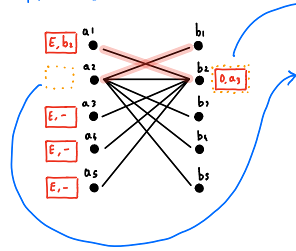
0	1	2	3
4	0	0	0
5	0	1	1
0	1	2	3

→

LINEE = RIGHE ∨ COLONNE

min (= ||) : 0 0 0
COPERTI TUTTI GLI ZERI

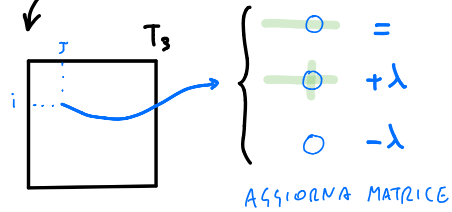
ULTIMA ITERAZIONE
PRIMA CHE FINISCA



T_2

2	0	3	5	5
0	0	0	0	0
1	0	2	1	1
2	0	3	3	3
3	0	1	1	2

→ $\lambda = \min(\ldots)$ = 1
NON COPERTI DALLE LINEE



LOWER BOUND

$$\sum d \cdot x = \underbrace{\sum d^3 \cdot x - \sum d_i^3 - \sum d_j^3}_{D_2} + D_0 + D_1$$

$-\lambda(n - h_1)$ RIGHE NON COPERTE

λh_2 COLONNE COPERTE

$$= \sum d^3 \cdot x + \lambda(n - |\Delta|) + D_0 + D_1$$

LIMITE INFERIORE

→ TROVA IN T_3 SOTTOINSIEME INDIP. → ϟ

↓

→ T_4 → T_5 ...

BRANCH AND BOUND

ENUM. IMPLICITA

PROBLEMA DI MASSIMO

$$\max_{x \in S} f(x)$$

- f: FUNZIONE OBIETTIVO
- x: SOLUZIONE PROBLEMA
- S: REGIONE AMMISSIBILE

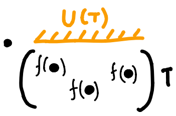
UPPER BOUND

DEL VALORE

$T \subseteq S$ REGIONE AMMISSIBILE

$$U(T) \geq f(x) \quad \forall x \in T$$

NON SI PUÓ FARE MEGLIO DI COSÍ IN T

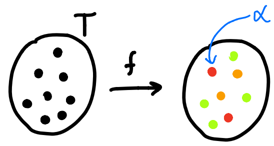
- 
TEMPI BREVI → DETERMINATO CON RILASSAMENTO
- 

VICINO A VALORI ALTI EFFETTIVI

RILASSAMENTO

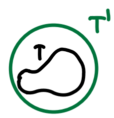
$$\alpha(f, T) = \max_{x \in T} f(x)$$

VALORE OTTIMO

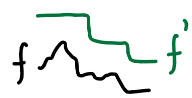


$$\rightsquigarrow \alpha(f', T') = \max_{x \in T'} f'(x)$$

$T \subseteq T'$ $\quad f'(x) \geq f(x) \quad \forall x \in T$



REGIONE PIÚ AMPIA

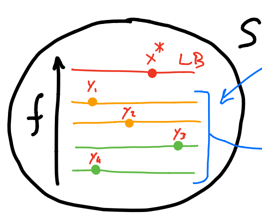


PROBLEMA PIÚ SEMPLICE DA RISOLVERE

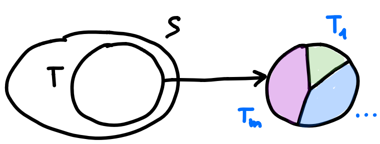
LOWER BOUND

$$LB \leq f(x^*) = \max_{x \in S} f(x)$$

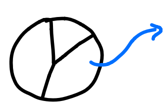
ALMENO QUESTO É FATTIBILE IN S



BRANCHING



CANCELLAZIONE SOTTOINSIEMI



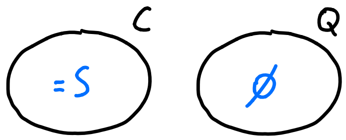
$U(T_2) \leq LB$ FA PEGGIO DEL MINIMO, NIENTE DI BUONO QUI DENTRO

$$\Rightarrow \forall x \in T_2 \;\; f(x) \leq LB$$

SENZA CALCOLARE f $\forall x$

ALGORITMO

①



SOTTOINSIEMI DA TENERE IN CONSIDERAZIONE — SOTTOINSIEMI ELIMINATI

$k = 1$

$U(s) = !$ ~ EURISTICA

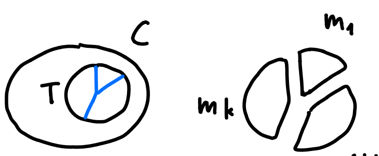
$LB = !$ ~ V $-\infty$

② SELEZIONE SOTTOINSIEME

→ $U(T) =$ … $= \max_{Q \in C} U(Q)$

UPPER BOUND PIÙ ELEVATO

③ BRANCHING



④ UPPER BOUNDING

SOTTOPROBLEMA RILASSAMENTO

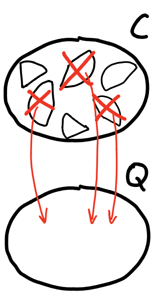
$U(T_1)$ … $U(T_k)$

⑤ LOWER

$LB = $ ↻

AGGIORNA CON IL PIÙ ELEVATO

⑥ ELIM. SOTTOINSIEMI



$U(Q) \leq LB$

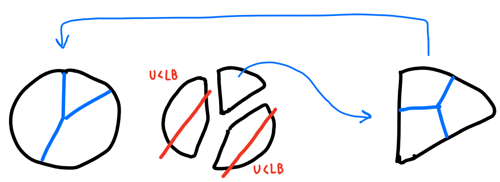
⑦ $C = \emptyset \Rightarrow$ (!) , $LB = f(x^*)$ LOWER BOUND È VALORE OTTIMO

k ++

GOTO ①

$T^* \in Q$

↖ x^*



PROBLEMI DI MINIMO

$U(T) \rightarrow L(T)$

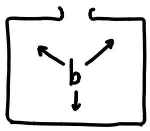
LOWER BOUND

$LB \rightarrow UB \geq f(x^*) = \min_{x \in S} f(x)$

~~∅~~ $L(Q) \geq UB$

② $U(T) =$

KNAPSACK 2^n



b = CAPACITÁ (BAG)

n = OGGETTI ▽ ▽ ▽

▽ : peso p_i, \$ v_i

$x_i = \begin{cases} 0 \\ 1 \end{cases}$

max $\sum^n v_i x_i$ ↗ Σ\$

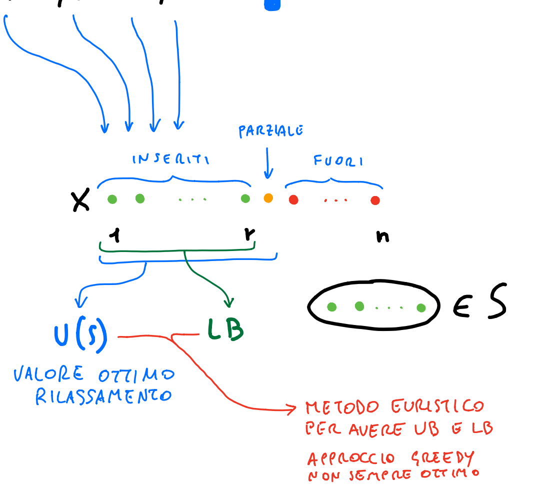
max $\sum^n p_i x_i \leq b$

UPPER BOUND $U(S)$ → RISOLUZIONE RILASSAMENTO

$x_i \in \{0, 1\} \longrightarrow x_i \in [0, 1]$

RILASSAMENTO LINEARE

$\frac{v_1}{p_1} \geq \frac{v_2}{p_2} \geq \dots \geq \frac{v_n}{p_n}$ — \$ / peso: RAPPORTO VALORE / PREZZO



INSERIMENTO PROGRESSIVO >_>

$b - p_1$

$- p_1 - p_2$

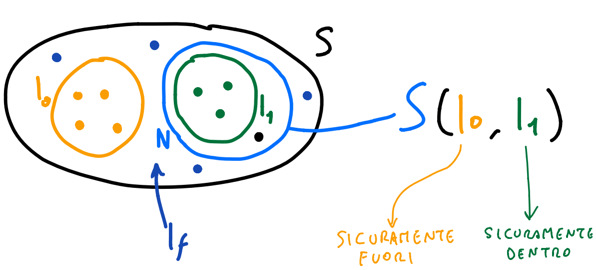
... → FINO A $b - \sum^{r+1} p_j < 0$

$\frac{b - \sum^r p_j}{p_{r+1}}$

$X = 1 \ 1 \ \dots \ 1 \ \downarrow \ 0 \ \dots \ 0$

1 ... r ... n

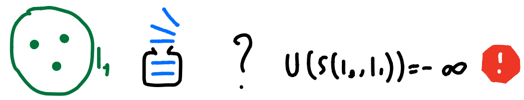
SOTTOINSIEMI PARTICOLARI → PROBLEMA RISTRETTO



- $\max \sum_{l_1} v_i + \sum_{l_f} v_i x_i$
- $\sum_{l_f} p_i x_i \leq b - \sum_{l_1} p_i$
- $x_i \in \{0, 1\}$

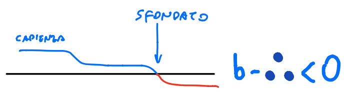
CALCOLO

①

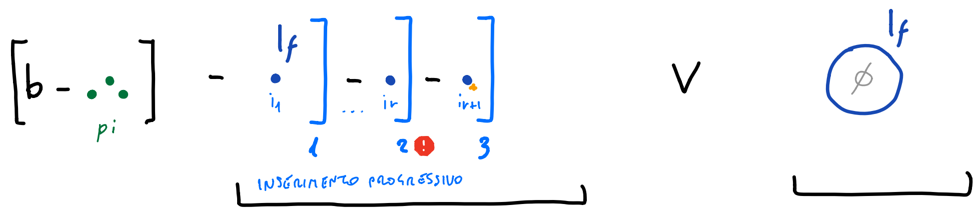


? $U(S(l_0, l_1)) = -\infty$ (!)

SE ELEMENTI DENTRO SFONDANO LO ZAINO



②



$U(S(l_0, l_1)) = (\ldots) \cdot (\ldots)$

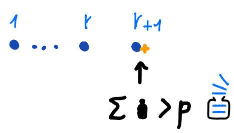
$U(S(l_0, l_1)) = \ldots$

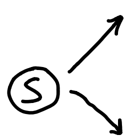
$N = l_1 \cup \ldots \in S$

SOLUZIONI AMMISSIBILI

$N = l_1 \cup l_f \in S$

BRANCHING





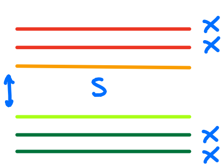
$S(\{r+1\}, \emptyset)$

NESSUN VINCOLO SU OGGETTI CHE DEVONO ESSERCI

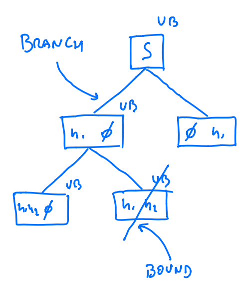
SOTTOINSIEME SOLUZIONI SENZA OGGETTO r+1

$S(\emptyset, \{r+1\})$

PRESO PARZIALMENTE



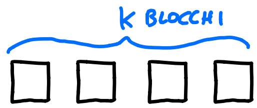
ESCLUDI SOTTOINSIEMI CON UPPER BOUND E LOWER BOUND PEGGIORI DI QUELLI TROVATI



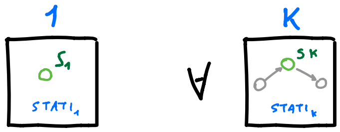
LB = C

PROGRAMMAZIONE DINAMICA

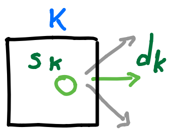
PROPRIETÁ PROBLEMA

- 

PROCESSO DECISIONALE

- 

IN OGNI BLOCCO CI SI TROVA IN UNO STATO

- 

BISOGNA PRENDERE UNA DECISIONE

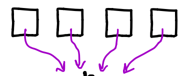
$d_k \in D_k(s_k)$

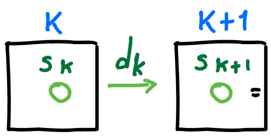
DIPENDONO DAGLI STATI

$u(d_k, s_k)$ → $f = \sum^n u$

CONTRIBUTO

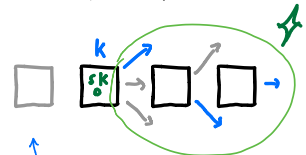
FUNZIONE OBIETTIVO



- 

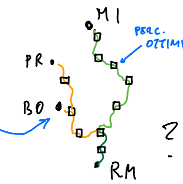
$= t(d_k, s_k)$ FUNZIONE DI TRANSIZIONE

OTTIMALITÁ



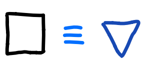
SEQUENZA DECISIONI OTTIME

NON DIPENDE DA BLOCCHI PRECEDENTI

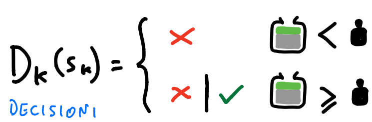


SE MI TROVO QUI NON MI SERVE SAPERE COME CI SONO ARRIVATO

KNAPSACK (PD)

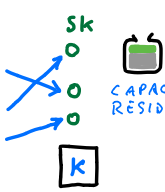


I BLOCCHI SONO GLI OGGETTI



DECISIONI

NON CI STAREBBE



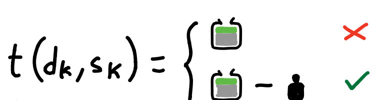
CAPACITÁ RESIDUA

$S_1 = b$

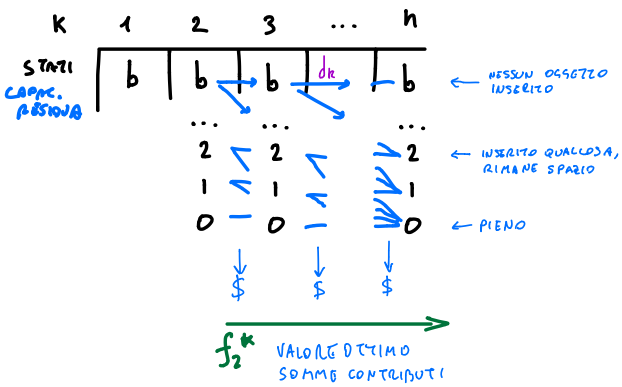
ZAINO VUOTO ALL'INIZIO

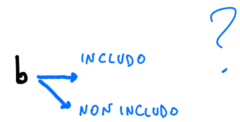
$u(d_k, s_k) = \begin{cases} 0 & \times \\ \$ & \checkmark \end{cases}$

CONTRIBUTO



TRANSIZIONE

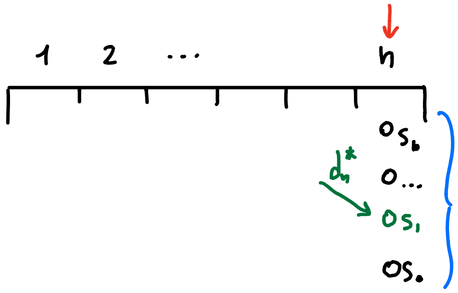




ALGORITMO - VALORE OTTIMO

→ GUARDA GLI ESERCIZI

①

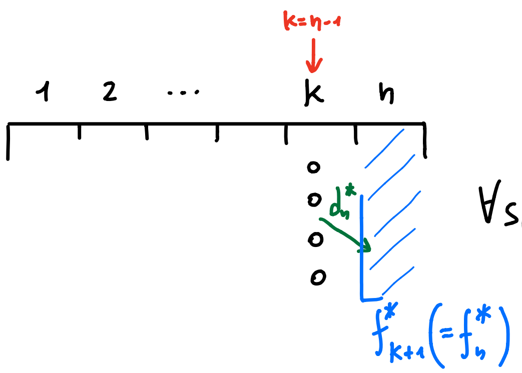


$\forall s_n \rightarrow f_n^*(s_n)$, $d_n^*(s_n)$

- f_4^*, d_4^*
- f_3^*, d_3^*
- f_2^*, d_2^*
- ... ...

VALORI E DECISIONI OTTIME PER OGNI STATO E n

②



$$\forall s_k \rightarrow f_k^*(s_k) = \max_{d_k \in D_k(s_k)} \left[u(d_k, s_k) + f_{k+1}^*(\ldots) \right]$$

$$d_k^*(s_k)$$

③ $k = 1 \Rightarrow f_1^*(s_1)$ ✧

: $k = k-1$, GOTO ①

ALGORITMO - SOLUZIONE OTTIMA

① $S_1^* = S_1$, $K = 1$

② K: $d_k^*(s_k^*)$ $\quad s_{k+1}^* = t(d_k^*(s_k^*), s_k^*)$

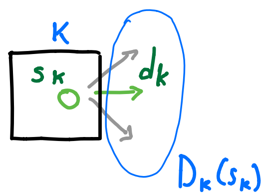
DECISIONE OTTIMA

③ $k = n \Rightarrow$ ❗ $\rightarrow d_1^*(s_1^*) \ldots d_n^*(s_n^*)$

SOLUZIONE OTTIMA
LISTA DI DECISIONI

: $k = k+1$, GOTO ②

COMPLESSITÁ



DA CALCOLARE

$\rightarrow u(\ldots)$ CONTRIBUTO

$\rightarrow t(\ldots)$ TRANSIZIONE

$\rightarrow u(\ldots) + f^*_{k+1}(t(\ldots))$

$\rightarrow \max\limits_{d_k \in D_k(s_k)}$

TROVA IL MASSIMO DEI VALORI AL VARIARE DI d_k

4x

NUMERO OPERAZIONI RICHIESTE

$$: \sum_{s_k \in S_k} 4 \cdot |D_k(s_k)|$$

CARDINALITÁ

TUTTI STATI DI K

BLOCCHI   STATI   DECISIONI PER STATO

$$\rightarrow \sum_{k=1}^{n} \sum_{s_k \in S_k} 4 \cdot |D_k(s_k)|$$

NUMERO COMPLESSIVO OPERAZIONI

... KNAPSACK

STATO PIENO

$|S_1| = 1 \qquad |S_k| = b+1$

$|D_k(s_k)| = 1 \vee 2$

NO   SI/NO

$\rightarrow O(nb)$

ESPONENZIALE

$b = 2^k$

BIT NECESSARI PER RISOLVERE IL PROBLEMA

IN CASI PEGGIORI FUNZIONA MEGLIO DI BRANCH & BOUND

	1	2	3	4
\$	3	2	4	5
8	2	1	3	4

	f^*_4	d_4
1	0	NO
2	0	:
3	0	NO
4	5	SI
5	5	:
6	5	
7	5	SI

Righe 1–3: NON CI STA

Righe 4–7: CI STA DECISIONE OTTIMA $\rightarrow$

CONSIDERO SOLO OGGETTO 4

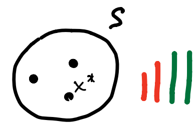
PROBLEMI OTTIMIZZAZIONE

$$f : S \to \mathbb{R}$$

- ISTANZA: $f : S$
- f: FUNZIONE (CHE CALCOLA) OBIETTIVO
- S: REGIONE AMMISSIBILE
- $\mathbb{R}$: RISULTATO?

- **COMBINATORIA** — PUNTI FINITI O ∞ NUMERABILI
- **CONTINUA** — PUNTI ∞ NON NUMERABILI

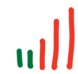
MASSIMO



SOLUZIONE ISTANZA: $f(x^*) \geq f(x) \ \forall x$

- $f(x^*)$: VALORE OTTIMO
- x^*: SOLUZIONE OTTIMA

MINIMO



$\leq$

COMPLESSITÁ

$\dim(I)$ — DIMENSIONE IN BIT; I: ISTANZA

$n_{op_A}(I)$ — NUMERO OPERAZIONI; A: ALGORITMO

ANALISI WORST CASE

$$t_A(k) = \max_{I : \dim(I) = k} n_{op_A}(I)$$

- t_A: TEMPO RISOLUZIONE
- k: DIMENSIONE ISTANZA
- n_{op_A}: TEMPI DI ESECUZIONE
- $I : \dim(I) = k$: ISTANZE DI DIMENSIONE K

ORDINE DI GRANDEZZA

$$t_A(k) = O(g(k))$$

- g: FUNZIONE PARAGONATA

$$\Leftrightarrow \exists \mu > 0 : t_A(k) \leq \mu g(k)$$

COMPLESSITÁ

$$g(k) \begin{cases} \log_2(k) \\ k \\ k^2 \\ 2^k \\ k! \end{cases}$$

POLINOMIALE k^p

ESPONENZIALE

CLASSE P

$\Rightarrow \exists \mathcal{A} : O(k^p)$

ESISTE ALGORITMO DI RISOLUZIONE DI COMPLESSITÀ POLINOMIALE

CLASSE NP

NON-DETERMINISTIC

$! \, x^* \Rightarrow \$^* : O(k^p)$

- x^*: NOTA SOLUZIONE OTTIMA
- $\$^*$: VALORE OTTIMO
- $O(k^p)$: CALCOLABILE IN TEMPO POLINOMIALE

→ NON SI SA COMPLESSITÀ RISOLUZ. ALGORITMO

$$P \subseteq NP$$

TRASFORMAZIONE IN TEMPO POLINOMIALE

$$R_1 \xrightarrow{p} R_2 \iff \forall I \in R_1 \text{ RISOLTA TRAMITE } I' \in R_2 : \dim(I') \leq p(k)$$

- R_1: PROBLEMA OTTIMIZZAZIONE
- $\xrightarrow{p}$: TRASFORMABILE IN TEMPO POLINOMIALE
- I: ISTANZA
- $\dim$: DIMENSIONE
- p: QUALCHE POLINOMIO
- k: DIMENSIONE ISTANZA IN R_1

$$\Rightarrow \left.\begin{array}{ll} \text{TRASF} & O(k^p) \\ \text{SOLUZ.} & O(k^p) \end{array}\right\} R_1 \text{ SOLUZ. } O(k^p)$$

RISOLVIBILE IN TEMPO POLINOMIALE

PROBLEMI NP-COMPLETI

- $R \in NP$
- $\forall \, Q \in NP \;\; \exists \;\; Q \xrightarrow{p} R$ (RIDUZIONE POLINOMIALE)

ε-APPROSSIMAZIONE

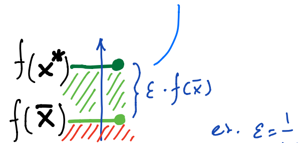
$I : \;\; \forall x \in S \;\; f(x) \geq 0 \qquad opt = f(x^*) \qquad \varepsilon \geq 0$

- I: ISTANZE
- x: SOLUZIONE
- $f(x)$: VALORE SOLUZIONE
- opt: VALORE OTTIMO
- x^*: SOLUZIONE OTTIMA
- ε: QUANTO VIENE APPROSSIMATO IL PROBLEMA

PROB. MAX

$$\bar{x} \in S : \frac{opt}{f(\bar{x})} \leq 1 + \varepsilon$$

- opt: VALORE OTTIMO
- $f(\bar{x})$: VALORE APPROSSIMATO



PROB. MIN

$$\frac{f(\bar{x})}{opt} \leq 1 + \varepsilon$$

ALGORITMI

$\mathcal{A}_\varepsilon \longrightarrow \checkmark$ PROBLEMA ε-APPROX $\quad \forall I$ (ISTANZA PROBLEMA)

COMPLESSITÁ $\mathcal{A}_\varepsilon$ PER NP-COMPLETE

$\forall \varepsilon > 0$

$\exists \mathcal{A}_\varepsilon$

ESISTE ALGORITMO ε-APPROSSIMAZIONE

- FPTAS (FULLY POLYNOMIAL)

$O(k^n)$: dim I $\wedge$ $O(k^n)$: $\frac{1}{\varepsilon}$

TEMPO POLINOMIALE — INVERSO PRECISIONE RICHIESTA — RISPETTO DIMENSIONE ISTANZE

- PTAS

$O(k^n)$: dim I $\wedge$ $O(n^k)$: $\frac{1}{\varepsilon}$

TEMPO ESPONENZIALE

SCHEMA APPROX. POLINOMIALE

- CASO 3

ε PICCOLO $\Rightarrow$ NP-COMPLETO

ε GRANDE $\Rightarrow$ P

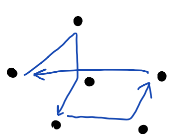
- CASO 4

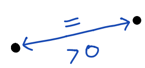
$\forall \varepsilon \Rightarrow$ NP-COMPLETO

TSP

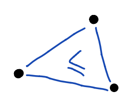
TRAVELLING SALESMAN PROBLEM

SIMMETRICO $\subseteq$ METRICO





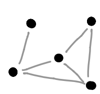
$d_{ij} = d_{ji}$



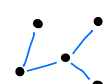
DISUGUAGLIANZA TRIANGOLARE

ALG. DOUBLE SPANNING TREE (DST)

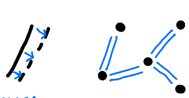
$G = (V, A)$



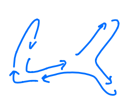
① MST $T = (V, A_T)$



②



DUPLICA ARCHI

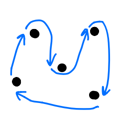


CICLO EULERIANO

$i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{2n-1} = i_1$

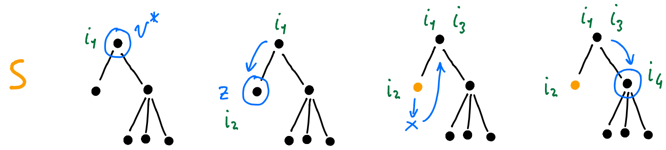
③ A → B → ~~A~~ → C →

RIMUOVI RIPETIZIONI



CIRCUITO HAMILTONIANO

CALCOLO CICLO EULERIANO



. . .

VISITA TUTTI I NODI IN ORDINE

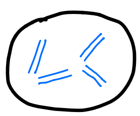
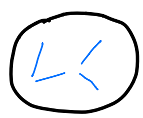
APPROSSIMAZIONE

$\mathcal{A}$ DST É DI 1-APPROX PER PROBLEMA TSP

→ 1-APPROX

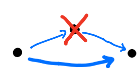
$$L_{DST} \leq 2 \cdot L^*$$

LUNGHEZZA PERCORSO — ALGORITMO — OTTIMA

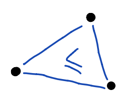
$A_Q =$  $A_T =$ 

TREE

$$\sum_{e \in A_Q} d_e = 2 \sum_{e \in A_T} d_e$$

3: 

QUANDO SI SOSTITUISCONO DUE ARCHI



RISULTANTE CON DISTANZA MINORE

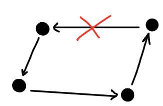
$$\Rightarrow \sum_{e \in A_c^*} d_e \leq \sum_{e \in A_c} d_e \leq \sum_{e \in A_Q} d_e$$

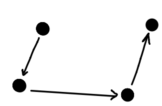
OTTIMA — ATTUALE — MASSIMO (CICLO EULERIANO)

$$\sum_{e \in A_c^*} d_e \geq \sum_{e \in A_T} d_e$$

SE RIMUOVI ARCO DA CIRCUITO

RIMANE UN ALBERO ≥ COSTO MINIMO





$$\sum_{e \in A_c} d_e \leq 2 \sum_{e \in A_T} d_e \leq 2 \sum_{e \in A_c^*} d_e$$

1-APPROSSIMAZIONE DIMOSTRATA