TSP SIMMETRICO CON BRANCH & BOUND

OTTIMIZZAZIONE COMBINATORIA 2021/22

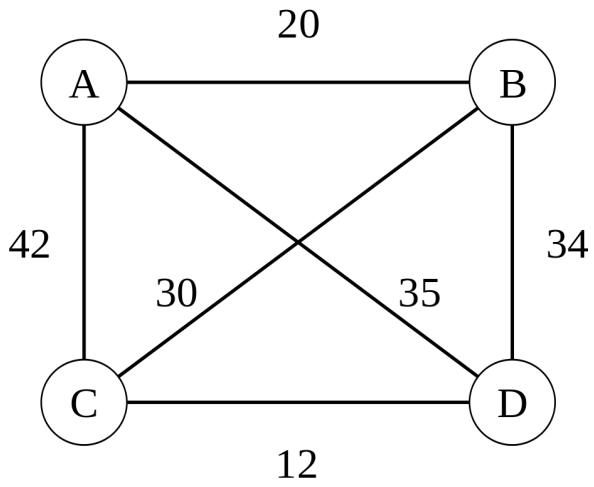
WALTER MALTESE

TSP SIMMETRICO

- Il problema del commesso viaggiatore (“Travelling Salesman Problem”) ha la seguente formulazione:
  ”Dato un insieme di città, e note le distanze tra ciascuna coppia di esse, trovare il tragitto di minima percorrenza che un commesso viaggiatore deve seguire per visitare tutte le città una ed una sola volta e ritornare alla città di partenza.”
- Il TSP risulta un problema NP-hard, nella precisione NP-completo e pertanto gli algoritmi che lo risolvono all’ottimo richiedono una complessità in tempo più che polinomiale (P != NP).

TSP SIMMETRICO

- Al problema TSP è associabile un grafo non orientato G = (V, E) con costi cij associati agli archi, da cui si richiede di determinare un insieme di archi C∗ ⊂ E con costo minimo possibile. Tale insieme C∗ deve formare un **circuito hamiltoniano**, cioè un *ciclo che passa una ed una sola volta per ogni nodo del grafo*.
- Il problema che analizzeremo è la versione *simmetrica* del TSP, dove cij = cji ∀i, j ∈ V . Ogni arco è quindi percorribile in entrambe le direzioni spendendo lo stesso costo.



FORMULAZIONE MATEMATICA

$$\min z = \sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$$

$$(1) \quad \sum_{j\in V, i\neq j} x_{ij} = 2 \quad \forall i \in V$$

$$(2) \quad \sum_{(i,j)\in E, i,j\neq n} x_{ij} = |V| - 2$$

$$(3) \quad \sum_{(i,j)\in E(U)} x_{ij} \leq |U| - 1 \quad \forall\, U \subseteq V \setminus \{n\} : |U| \geq 3$$

$$(4) \quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in E$$

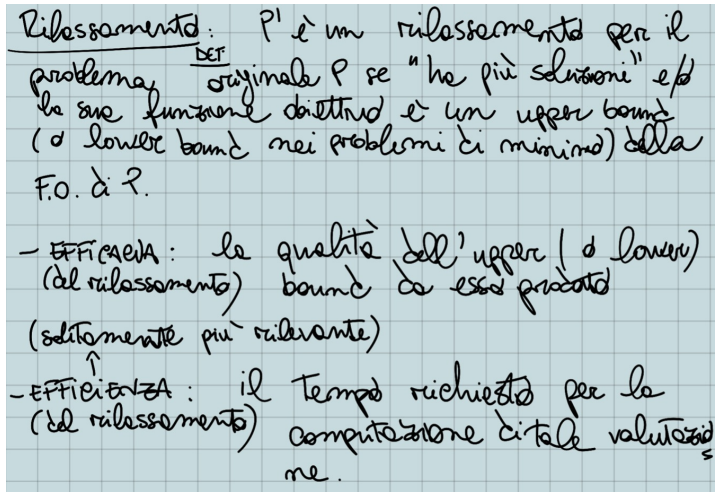
- Un circuito è detto hamiltoniano se su ogni nodo incidono esattamente due archi e, rimuovendo un nodo n a piacere e i suoi due archi incidenti, si ottiene un albero sui rimanenti $|V| - 1$ nodi.
- (1) Implica che su ogni nodo $I \in V$ incidano esattamente 2 archi.
- (2) e (3) Servono, invece, per garantire che una volta scelto un nodo n, $V \setminus \{n\}$ risulti un *albero* e quindi abbia $|V| - 2$ *archi e* sia *privo di circuiti*.
- (4) Modella il dominio delle variabili decisionali che assumono valore 1 se il corrispondente arco viene inserito nel circuito hamiltoniano e 0 altrimenti.

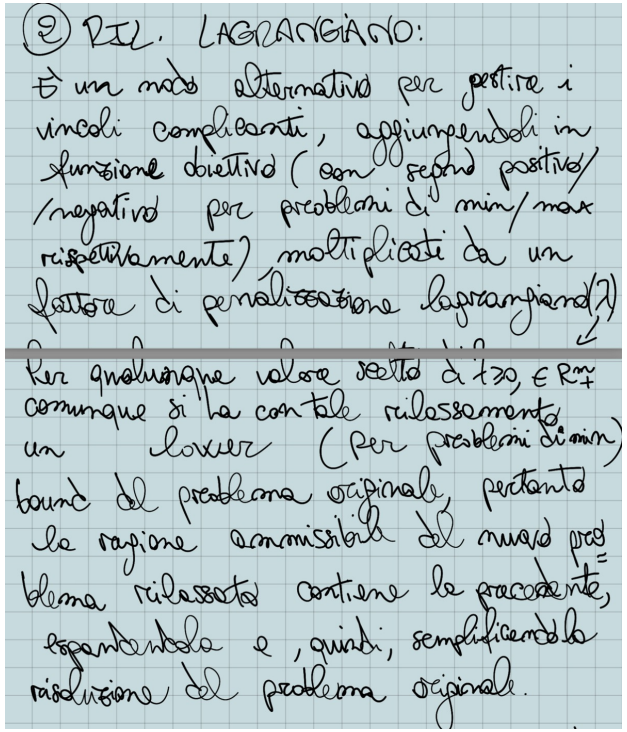
FORMULAZIONE MATEMATICA

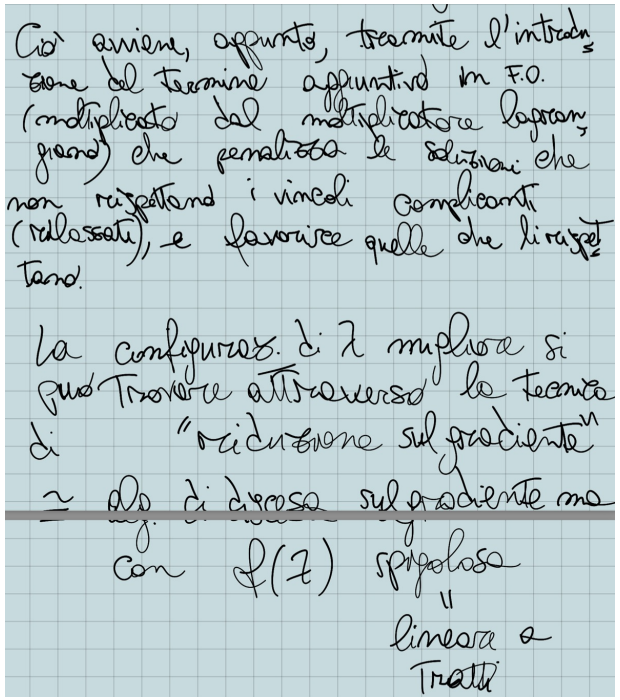
$$(3) \qquad \sum_{(i,j)\in E(U)} x_{ij} \leq |U| - 1 \qquad \forall\, U \subseteq V \setminus \{n\} : |U| \geq 3$$

- Dato un sottoinsieme di nodi U ⊆ V definiamo E(U) come l'insieme di archi {(i, j) | i, j ∈ U}. Osservando che ogni circuito sui nodi in U deve avere |U| archi in E(U), il vincolo (3) viene inserito al fine di eliminare il possible crearsi di tali circuiti. Imponiamo |U| ≥ 3, poichè se fosse per esempio uguale a 2 allora E(U) conterrebbe solamente l'arco tra i due nodi e non ci sarebbe alcun circuito.

RILASSAMENTO LAGRANGIANO – TEORIA







LOWER BOUND: RILASSAMENTO LAGRANGIANO

- L'approccio adottato prevede invece un rilassamento Lagrangiano sul vincolo (1) che ci conduce al problema di trovare il minimo 1-Tree, una volta impostati i moltiplicatori lagrangiani λi a 0, ossia eliminando il vincolo (1) per tutti i nodi tranne che per il nodo selezionato n.
- Il problema rilassato ha la seguente interpretazione: selezionare n lati di costo complessivo minimo, di cui 2 incidenti sul vertice 1, che garantiscano la connessione. Il problema risultante è quindi 1' 1-albero.
- La qualità (in termini di valore del bound) fornito da questo rilassamento è generalmente abbastanza scarsa per le istanze della letteratura.
  È possibile migliorare la qualità dei bound ottenuti tenendo conto in modo lagrangiano dei vincoli (di grado) rilassati.

LOWER BOUND: RILASSAMENTO LAGRANGIANO

- Una volta introdotti dei moltiplicatori lagrangiani λi per ogni nodo e portato in funzione obiettivo il vincolo otteniamo:

$$\min z = \sum_{(i,j)\in E} c_{ij} \cdot x_{ij} + \sum_{k\in V} \lambda_k (2 - \sum_{j\in V, k\neq j} x_{kj})$$

$$(5) \quad \sum_{j\in V, j\neq n} x_{nj} = 2$$

$$(2) \quad \sum_{(i,j)\in E, i,j\neq n} x_{ij} = |V| - 2$$

$$(3) \quad \sum_{(i,j)\in E(U)} x_{ij} \leq |U| - 1 \quad \forall\, U \subseteq V \setminus \{n\} : |U| \geq 3$$

$$(4) \quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in E$$

LOWER BOUND: RILASSAMENTO LAGRANGIANO

- Per valori fissati dei vari λk il rilassamento lagrangiano è facilmente risolvibile con una procedura che consente di **individuare l'1-tree di costo minimo**.

  Riformulando la funzione obiettivo si ottiene:

$$\min \sum_{(i,j)\in E} (c_{ij} - \lambda_i - \lambda_j) \cdot x_{ij} + 2 \cdot \sum_{k\in V} \lambda_k$$

- I costi associati agli archi si aggiornano, pertanto, secondo la seguente equazione : c′ij = cij − λi − λj
- Da questa riformulazione si può facilmente passare al duale lagrangiano, che prevede di partire da una combinazione ammissibile di moltiplicatori come λk = 0 ∀k, provando in seguito a migliorare la soluzione attraverso la ricerca di nuovi possibili moltiplicatori.

1-TREE

- Dato un grafo G = (V,E) non orientato ed un suo nodo n chiamiamo **1-tree** un sottografo H = (V,Eh) di G con Eh ⊂ E e con le seguenti proprietà:
- 1) in Eh ci sono esattamente 2 archi incidenti sul nodo n;
- 2) se escludiamo da H il nodo n ed i suoi 2 archi incidenti su di esso ne risulta un albero sull'insieme di nodi V \ {n}. Ovvero, H contiene un circuito passante per il nodo candidato n, e pertanto: |Eh| = |V|

1-TREE

- Avendo rimosso il vincolo (1) abbiamo ottenuto la possibilità di ottimizzare sulla famiglia degli 1-alberi anzichè sulla famiglia dei cicli hamiltoniani; è evidente che la prima contiene la seconda come sottoinsieme stretto, per cui il valore della soluzione trovata sarà sempre minore o uguale al valore della soluzione ottima del problema di partenza.
- *Ogni circuito hamiltoniano, infatti, risulta un 1-tree*, **non** è invece vero il **viceversa**.
- Se indichiamo con S′ l'insieme di tutti gli 1-tree calcolabili dato un grafo G e con S la regione ammissibile del TSP, abbiamo che $S \subset S'$.
  Pertando **il problema di trovare il 1-Tree di costo minimo**:

$$\min_{H \in S'} \sum_{(i,j) \in E_H} c_{ij}$$

  risulta essere **un rilassamento per il problema del TSP simmetrico** e la sua risoluzione restituisce quindi un **lower bound** per il valore ottimo del problema del TSP.

1-TREE

- La formulazione matematica dell'1-tree è la seguente:
- Essa risulta equivalente a quella del rilassamento lagrangiano mostrato, una volta impostati i vari coefficienti λi = 0.

$$\min z = \sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$$

$$(5) \quad \sum_{j\in V, j\neq n} x_{nj} = 2$$

$$(2) \quad \sum_{(i,j)\in E, i,j\neq n} x_{ij} = |V| - 2$$

$$(3) \quad \sum_{(i,j)\in E(U)} x_{ij} \leq |U| - 1 \quad \forall\, U \subseteq V \setminus \{n\} : |U| \geq 3$$

$$(4) \quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in E$$

1-TREE – PROCEDURA DI RISOLUZIONE

- Si mostra, adesso, una procedura per calcolare in **tempo polinomiale** un **1-Tree di costo minimo**, generando un **lower bound** al problema del TSP:
- 1) Si calcoli l'MST T sul grafo ottenuto da G scartando il nodo prescelto n e tutti gli archi incidenti su di esso. Sia Et l'insieme degli archi della soluzione trovata;
- 2) Si aggiungano ad Et i due archi (n, k) e (n, h) a distanza minima tra quelli incidenti sul nodo n.
- 3) Si restituisca l'1-tree H = (V, Eh) con Eh = Et ∪ {(n, k), (n, h)}.

1-TREE – PROCEDURA DI RISOLUZIONE

- Il costo totale della procedura proposta è dominato dal costo del calcolo del Minimum Spanning Tree, computabile agevolmente, tramite un algoritmo greedy come quello di Kruskal, in tempo $O(m \cdot \log n)$.
  La selezione, al passo 2, della coppia degli archi incidenti di costo minore è ottenibile invece con una semplice scansione degli archi $O(m)$.
- Al costo di un maggiore sforzo computazionale è anche possibile calcolare tutti gli 1-tree minimi variando la scelta del nodo candidato n tra tutti i $|V|$ nodi del grafo.
  Il lower bound complessivamente migliore risulterà, in tal caso, il maggiore fra quelli trovati.

- Riguardo la computazione e l'aggiornamento del valore dell'**upper bound**, si ricorda che esso si basa sull'identificazione di soluzioni ammissibili trovate durante l'esecuzione dell'algoritmo.
  Di conseguenza, ogni volta che si calcola un 1-tree, se questo risulta essere anche un circuito hamiltoniano (ogni suo nodo presenta grado 2), tale valore risulta una soluzione ammissibile (ottimalità) e si aggiorna l'upper bound se esso presenta un costo minore di quello per ora trovato. All'inizio della procedura di risoluzione l'upper bound verrà impostato a $+\infty$.

SCHEMA DI BRANCH

- Come dovrà essere partizionata la regione ammissibile iniziale S del problema in più sottoinsiemi?
- Se non ricadiamo nel caso ottimo in cui la soluzione del rilassamento (il 1-Tree ottenuto) è anche un circuito hamiltoniano, tale soluzione risulterà essere un 1-tree che contiene esattamente un ***sottocircuito***.
  Dobbiamo introdurre una **regola di suddivisione** il cui scopo è *impedire il formarsi nei nodi figli di tale sottocircuito*.

SCHEMA DI BRANCH

- Indichiamo con: {(i1, j1), (i2, j2), . . . , (ir, jr)} gli archi che compongo tale sottocircuito individuato nel 1-Tree risultante (su cui fare branch).
- Il primo nodo figlio verrà generato imponendo che l'arco (i1,j1) non faccia parte della soluzione, ossia impostando x(i1j1) = 0;
- Il secondo nodo figlio sarà ottenuto imponendo che sia presente l'arco (i1, j1), ma sia assente l'arco (i2, j2), ovvero impostando x(i1j1) =1 e x(i2j2) = 0;
- Il procedimento continua analogamente fino all' r-esimo figlio che avrà presenti tutti i primi r−1 archi del sottocircuito e non conterrà l'ultimo, quindi ∀k=1,2,…,r−1 x(ikjk) =1 e x(irjr) =0.

SCHEMA DI BRANCH

- Ad ogni nodo dell'albero di branch saranno quindi associati due insiemi:
- 1) E0 = insieme degli archi da non considerare in soluzione;
- 2) E1 = insieme degli archi che devono essere presenti in soluzione.

- Per ogni figlio si dovrà quindi risolvere un sottoproblema S(E0,E1) contenente tutti i circuiti hamiltoniani formati obbligatoriamente dagli archi in E1 e senza far uso degli archi in E0.
- Per il calcolo del lower bound di un sotto problema S(E0, E1) si usa il medesimo criterio precedentemente analizzato, imponendo però la presenza degli archi E1 ed escludendo quelli in E0 per il calcolo dell'MST e nella scelta dei 2 archi incidenti nel nodo candidato n.

SCHEMA DI BRANCH

- Per il branching di un nodo interno, al sottocircuito {(i1, j1), (i2, j2), . . . , (ir, jr)} individuato andranno tolti gli archi nell'insieme E1 associato al nodo stesso, che saranno obbligatoriamente presenti nei figli che verranno generati. Una volta scremato tale insieme di archi dal sottocircuito individuato, si procederà in maniera analoga e verranno estesi gli insiemi E0 ed E1 del padre per la generazione dei nodi figli.
- Seguendo la presente regola di branching i nodi figli di un dato nodo padre continueranno ad essere sottoinsiemi della regione ammissibile del padre.

CRITERI DI CHIUSURA DEI NODI

Un nodo dell'albero di branch Pi viene chiuso quando:

- 1) $z^{\wedge} \leq LB(Pi)$, ovvero quando il lower bound fornito risulta maggiore o uguale dell'attuale soluzione ammissibile, ossia il migliore (di costo minore) circuito Hamiltoniano trovato fino a quel momento.
  In tal caso il nodo viene **chiuso per bound**;
- 2) Risulta impossibile computare un 1-tree per il dato nodo.
  Nel presente sottocaso, si dice che non esiste soluzione ammissibile per il rilassamento e il nodo verrà **chiuso per inammissibilità**.
- 3) Il 1-tree generato dal rilassamento risulta essere anche un circuito hamiltoniano.
  Il nodo in questione verrà **chiuso** in quanto **possibile candidato ottimo**.

CRITERI DI CHIUSURA DEI NODI

- Se alcuni nodi risultano aperti la scelta del prossimo nodo (in ordine) su cui fare Branch ricade su quello che presenta un Lower Bound minore, ossia quello che ci permette, in caso di ottimalità, di chiudere prima eventuali nodi aperti e terminare l'esecuzione. Un approccio di questo tipo viene detto "**Best First**".

IMPLEMENTAZIONE – STRUTTURE DATI

- Strutture dati: Grafo e 1-Tree, calcolati con liste di adiacenza (HashMap)
- Il Grafo è costruito come un insieme di nodi, a loro volta costituiti da un insieme di archi (in essi incidenti).
- Tale approccio risulta conveniente a livello di complessità riguardo i task richiesti:
  Per le operazioni di scansione di adiacenti di un nodo e dell'intero grafo, le liste di adiacenza si dimostrano ottime rispetto alle matrici di adiacenza (esse risultano vantaggiose per il controllo di esistenza di un arco nel grafo, operazione non eseguita)

IMPLEMENTAZIONE - MST

- Tra le procedure più frequentemente eseguite troviamo il calcolo dell' **MST** (**Minimum Spanning Tree**), ripetuto per ogni nodo dell'albero di Branch che andiamo a generare.
  Per tale calcolo si è usufruito dell'algoritmo greedy di Kruskal che presenta complessità ottima $O(m \cdot \log n)$.
- Tale costo risiede principalmente nell'ordinamento decrescente iniziale degli archi, e nell'uso di Merge Find Set per controllare che un i-esimo arco possa essere incluso in soluzione con la certezza che non formi un ciclo.

IMPLEMENTAZIONE – RICERCA SOTTOCIRCUITO

- Per quanto riguarda l'individuazione del sottocircuito all'interno di un 1-tree che non risulta un circuito hamiltoniano (non è una soluzione ammissibile), e dunque sul quale far branch, si è usufruito di una ricerca di tipo Depht First, con complessità O(n + m).
- L'idea è usare un vettore di padri per evitare di visitare più volte i nodi del grafo e per memorizzare il padre di ogni nodo.
- Conclusa l'esplorazione in profondità del grafo, semplicemente partendo dal nodo candidato n e procedendo con i vettori dei padri a ritroso, si individua il ciclo che ci servirà per il fare il Branch di un problema Pi .

IMPLEMENTAZIONE

- La procedura centrale per la risoluzione del TSP risulta essere TSP_Resolution(), che restituisce un'istanza della classe TSP_Results, contenente il circuito hamiltoniano con costo minore (o, se impossibile da calcolare, il grafo originale) e una serie di statistiche riguardo ai nodi analizzati durante il processo di branching.
- Questo metodo richiama gli algoritmi sopra descritti e altre procedure di supporto al fine di gestire una coda di SubProblem_computation, e terminando l'esecuzione non appena quest'ultima viene completamente svuotata in seguito alla chiusura di tutti i nodi per una delle ragioni indicate.
  Nonostante tali sottoprocedure sono polinomiali in tempo, il numero di sottoproblemi totali da dover gestire può risultare esponenziale.
- La gestione dell'insieme dei sottoproblemi aperti viene realizzata con una coda di priorità **min-heap**, ossia con una struttura dati che presenta complessità O(1) per leggere l'elemento con lower bound minore e O(log n) per l'inserimento/rimozione di un elemento.

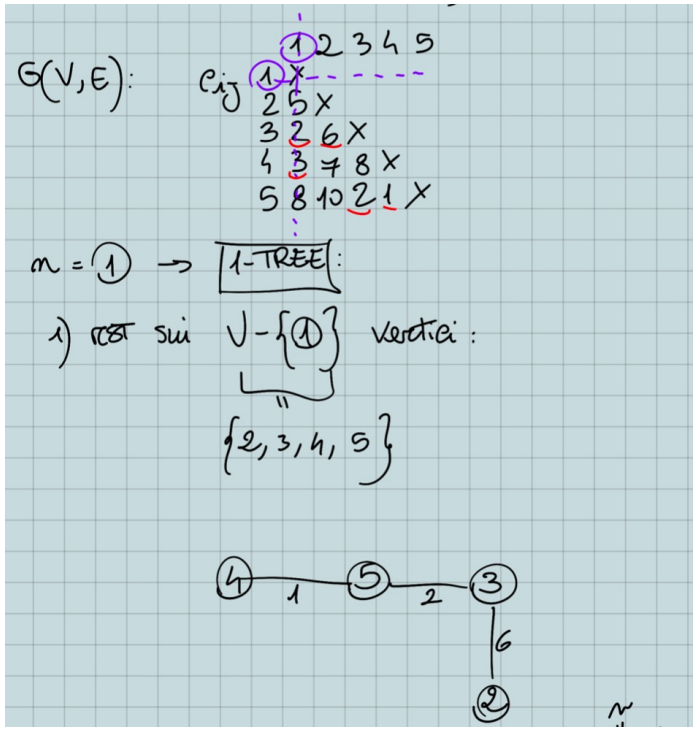
IMPLEMENTAZIONE – PRESTAZIONI

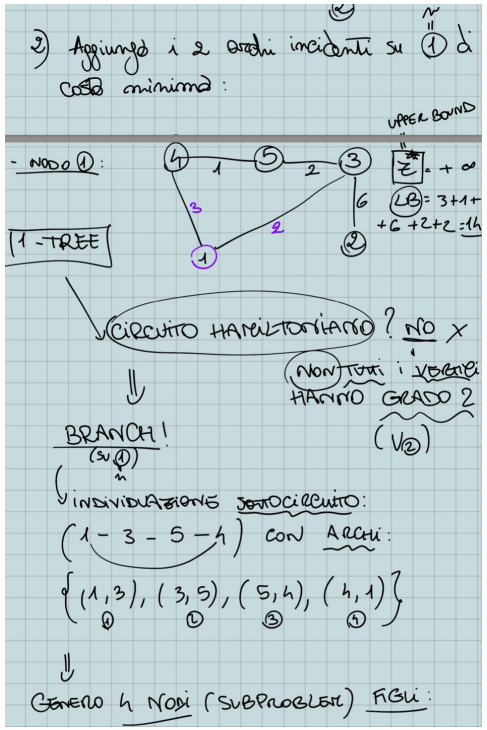
- Il tempo necessario per l'esecuzione dipende fortemente dal numero di archi che popolano il grafo, pertanto esecuzioni del TSP su grafi sparsi (nonostante un numero superiore di nodi) richiedono molto meno tempo di esecuzioni su grafi completi o maggiormente densi.

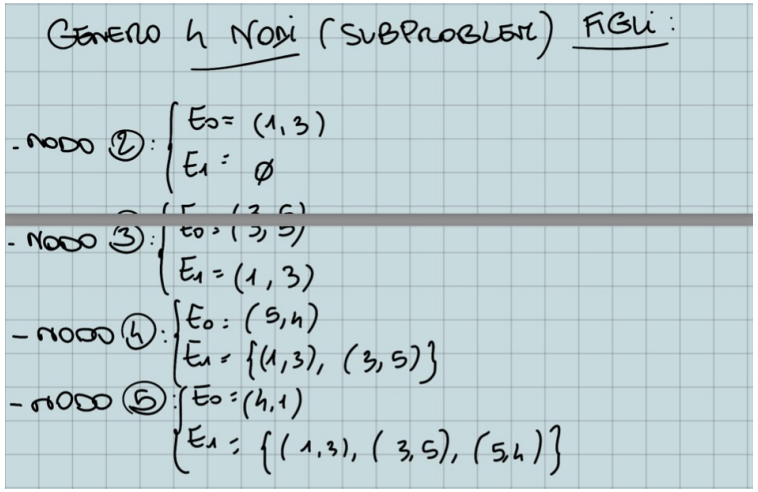
```
E'stata trovato una soluzione migliore con costo 7293 e percorso
(1)--213--(8)--851--(3)--355--(4)--700--(5)--371--(13)--701--(7)--678--(9)--403--(2)--357--(12)--1017--(11)--225--(6)--547--(10)--875--(1)


 Durante la ricerca sono stati creati:
 (6768) Nodi intermedi per la generazione di Branchess;
 (7) Nodi chiusi per possibile soluzione;
 (9724) Nodi chiusi per Bound;
 (0) Nodi chiusi per Unfeasibility.

Tempo complessivo di esecuzione: 17622 millisecondi

Process finished with exit code 0
```

```
Sono state trovate 2 soluzioni migliori con costo 17 e percorso
(1)--5--(2)--6--(3)--2--(5)--1--(4)--3--(1)
(1)--5--(2)--7--(4)--1--(5)--2--(3)--2--(1)


 Durante la ricerca sono stati creati:
 (2) Nodi intermedi per la generazione di Branchess;
 (2) Nodi chiusi per possibile soluzione;
 (3) Nodi chiusi per Bound;
 (0) Nodi chiusi per Unfeasibility.

Tempo complessivo di esecuzione: 92 millisecondi

Process finished with exit code 0
```

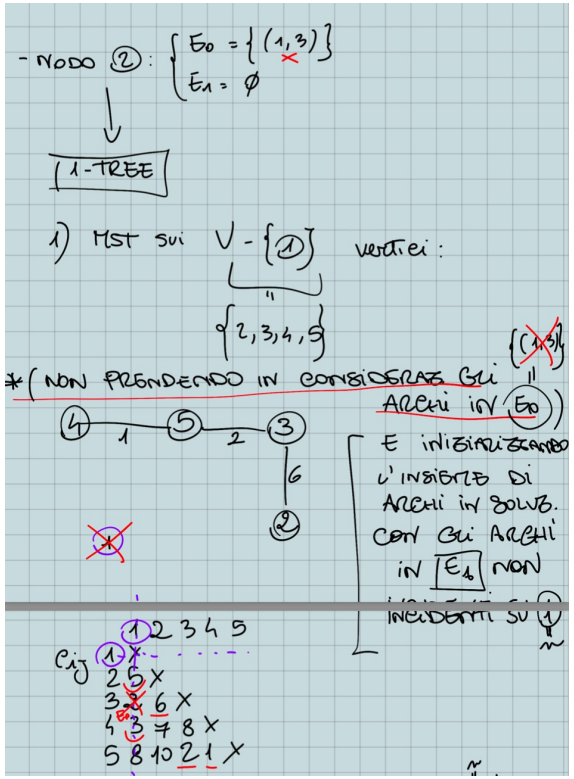
```
E'stata trovato una soluzione migliore con costo 61 e percorso
(1)--4--(2)--8--(3)--7--(4)--9--(5)--10--(6)--2--(7)--6--(9)--7--(8)--8--(1)


 Durante la ricerca sono stati creati:
 (30) Nodi intermedi per la generazione di Branchess;
 (1) Nodi chiusi per possibile soluzione;
 (4) Nodi chiusi per Bound;
 (52) Nodi chiusi per Unfeasibility.

Tempo complessivo di esecuzione: 212 millisecondi

Process finished with exit code 0
```

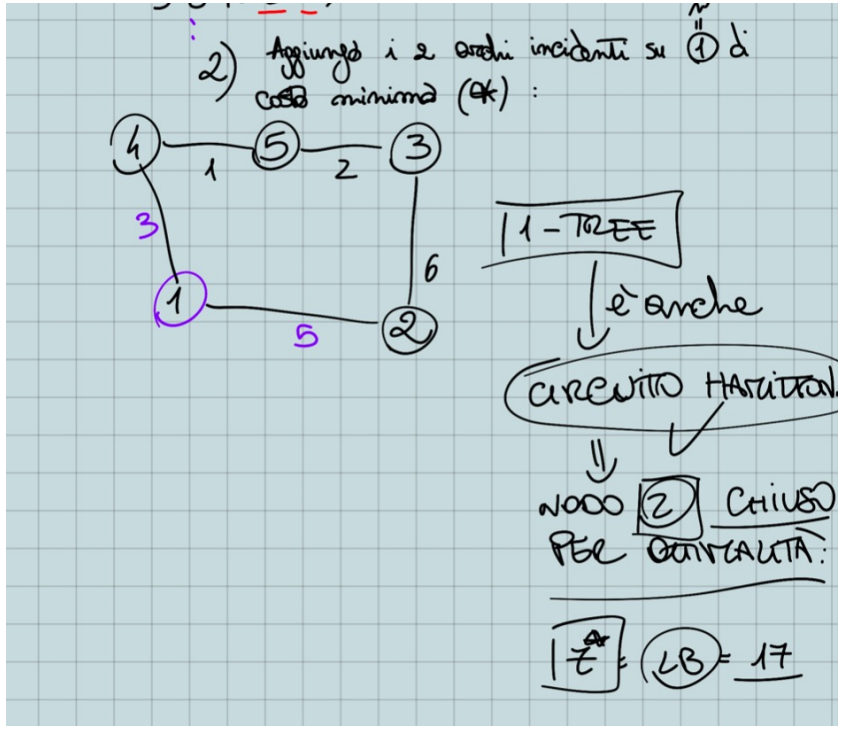




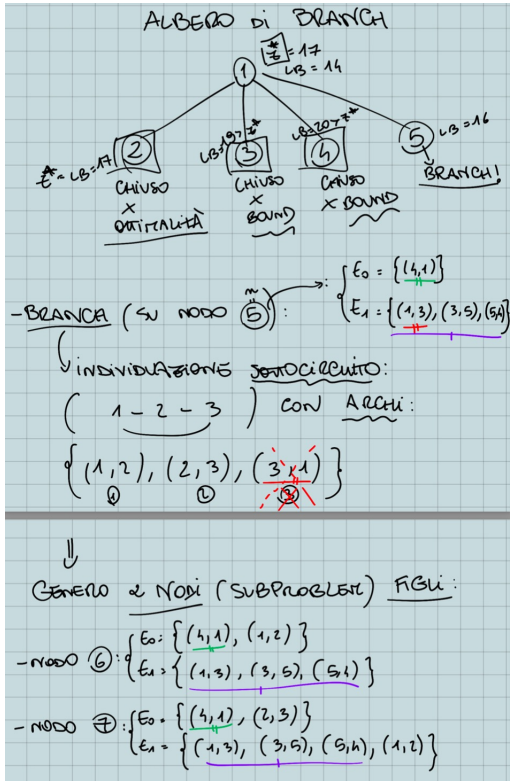


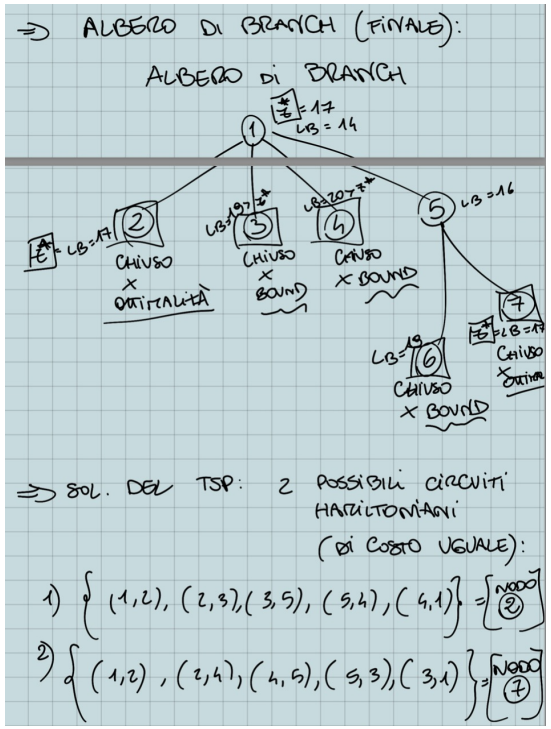
ESEMPIO – RISOLUZIONE TSP



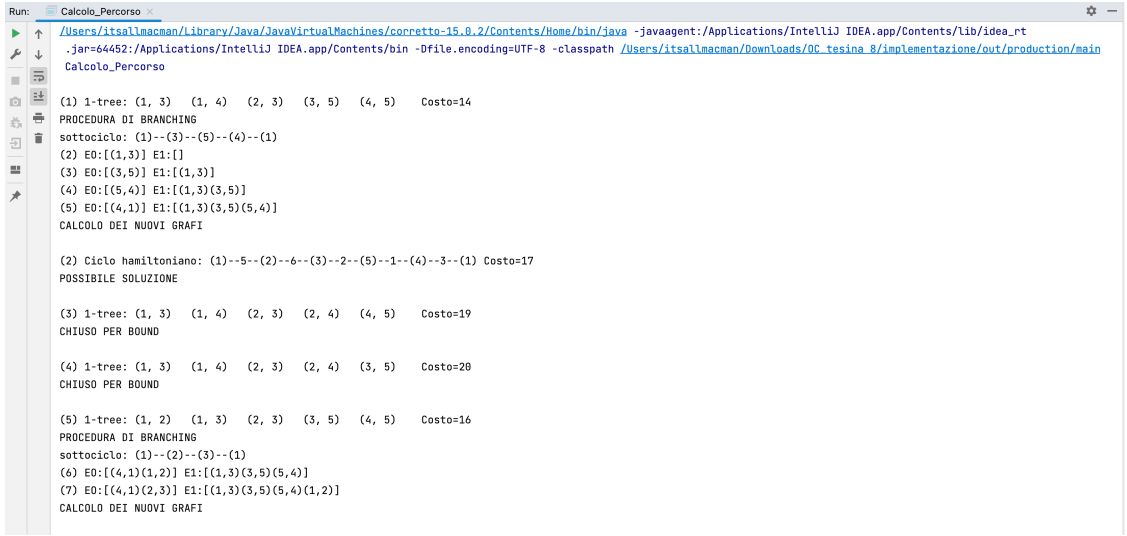


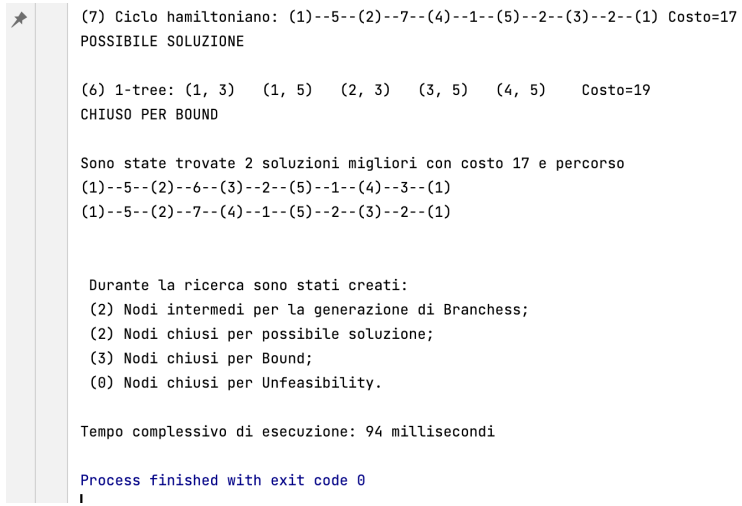
ESEMPIO – RISOLUZIONE TSP





ESEMPIO – RISOLUZIONE TSP





ESEMPIO – RISOLUZIONE TSP

GRAZIE PER L'ATTENZIONE

WALTER
MALTESE