Ottimizzazione combinatoria
Branch and Bound per il TSP

Francesco Iodice

1 Introduzione

In questo elaborato sarà analizzato il problema del commesso viaggiatore (*Travelling salesman problem*) che richiede di determinare un percorso che passa per un numero prefissato di città, permette di ritornare alla città di partenza dall'ultima città visitata, nessuna città sia visitata più di una volta e fra tutti i percorsi possibili deve essere quello più breve. Tale problema può essere modellato tramite i grafi, in cui i nodi corrispondono alle città e gli archi ai possibili collegamenti esistenti tra di esse, per caratteriazzare la soluzione al problema si può riccorrere ad un modello di programmazione linare intera (PLI). Verrano analizzate diversi varianti del problema, riconoscibili in base a determinate proprietà dell'istanza, e una metodologia di tipo Branch-and-Bound per la ricerca della soluzione ottima del problema.

2 Travelling Salesman Problem

Dato un grafo $G(V, E)$ con costi c_{ij} associati agli archi, il *Travelling salesman problem* (TSP) richiede di determinare un insieme di archi $C^* \subset E$ che costituisca un ciclo hamiltoniano, ovvero un ciclo nel grafo in modo che ogni vertice $i \in V$ sia visitato esattamente una volta e il costo degli archi selezionati sia minimo.

Nel caso in cui il grafo sia orientato, si parla di problema del commesso viaggiatore asimmetrico (ATSP).

Il problema ATSP può essere formulato come programma lineare a variabili intere, basato su variabili binarie x_{ij} tale che $ij \in E$ e $x_{ij} = 1$ se e solo se l'arco ij del grafo fa parte del ciclo hamiltoniano, 0 altrimenti.

Un possibile modello di PLI potrebbe essere il seguente:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in V} x_{ij} = 1 \qquad \forall j \in V \qquad (1)$$

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall i \in V \qquad (2)$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \leq |S| - 1 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2 \qquad (3)$$

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

La funzione obbiettivo impone di minimizzare la somma dei c_{ij} associati agli archi, nonché la somma delle distanze fra le città visitate presenti nel ciclo hamiltoniano. I vincoli (1) impongono che su ogni nodo $i \in V$ ci sia esattamente un arco entrante del

ciclo hamiltoniano, in modo complementare i vincoli (2) impogono che su ogni nodo ci sia esattamente un arco uscente. Per garantire l'unicità del ciclo sono presenti i vincoli (3), noti come *Subtour Elimination Constraints* (SECs).

Dato un qualunque insieme (non nullo) $S \subset V$ di vertici, i SECs impongono che il numero di archi scelti con entrambi gli estremi in S sia al più pari a $|S|-1$, ossia che non ci siano dei cicli in S; ovviamente S deve essere un sottoinsieme non vuoto di vertici e non deve coincidere con V (altrimenti sarebbero proibiti anche i cicli hamiltoniani).

Un modo alternativo di proibire i sottocicli è quello di rimpiazzare i (3) con i seguenti vincoli:

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 1 \tag{5}$$

Questa famiglia di vincoli impone la connessione tra l'insieme dei vertici S (con $1 \in S$ dove 1 è il nodo di partenza) e l'insieme $V \setminus S$, ogni sottoinsieme proprio di nodi $S \subset V$ deve essere connesso al resto del grafo, impedendo quindi che il sottografo di archi $ij \in E : x_{ij} = 1$ sia costituito da componenti connesse separate.

Il numero di SECs, espressi tramite la famiglia di vincoli (3) o (5), è esponenziale. Pertanto, non solo il problema del commesso viaggiatore è NP-completo (in quanto vi sono i vincoli di interezza delle variabili), ma la dimensione della formulazione (intesa come numero di variabili e vincoli) rende impraticabile anche la soluzione del rilassamento continuo (almeno per istanze di dimensioni interessanti).

Se l'istanza specifica del problema presenta alcune proprietà, possiamo riconoscere alcuni casi speciali e utilizzare procedure 'ad hoc' per la risoluzione esatta del problema.

Nel caso particolare in cui il grafo $G(V, E)$ è non orientato il problema del commesso viaggiatore è detto **simmetrico**, il costo per andare in un nodo i ad un nodo j è uguale in entrambe le direzione, $c_{ij} = c_{ji}$, $\forall i, j \in V$.

Anche il problema TSP simmetrico può essere formulato come programma lineare a variabili intere, basato sulle variabili binarie x_{ij}, in modo analogo a quanto fatto per l'ATSP, ma con una diversa formulazione dei vincoli resa possibile dalla non orientazione del grafo.

$$\min \sum_{ij \in E} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{ij \in E} x_{ij} = 2 \qquad \forall i \in V \tag{6}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 2 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 1 \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{8}$$

I vincoli (6) impogono che su ogni nodo incidano esattamente 2 archi. Analogamente a quanto accade per l'ATSP, i vincoli (7) impediscono la presenza di sottocicli in soluzione, rispetto al caso dell'ATSP la non orientazione degli archi permette di mettere a 2 il termine noto. Si noti che il problema del TSP simmetrico è un problema NP-completo.

3 Branch and Bound

Branch and bound è un paradigma di progettazione di algoritmi per problemi di ottimizzazione discreta e combinatoria, nonché ottimizzazione matematica. Gli algoritmi Branch and Bound sono detti di enumerazione implicita perché si comportano come un algoritmo di enumerazione - cioè "provano" tutte le soluzioni possibili fino a trovare quella ottima, scartandone alcune dimostrando a priori la loro non ottimalità. L'insieme delle soluzioni può essere rappresentato come un insieme di nodi organizzati ad albero e l'algoritmo mira ad esplorare questo albero e potare alcuni sottoalberi che sicuramenente non produrranno una buona soluzione. Queste tecniche sono applicate sopprattuto ai problemi NP in quanto il loro spazio delle soluzioni cresce esponenzialmente con le dimensioni del problema.
La procedura è molto generale e applicata al contesto della programmazione lineare intera può essere usata identificando un **rilassamento** del problema in esame e uno schema di suddivisione che permette di generare dei sottoproblemi più semplici.

Definizione. *Dato un problema di ottimizzazione* $P = \max\{c^T x : x \in F\}$ *dove* F *è la sua regione ammissibile, il* ***rilassamento*** *di* P *è definito come* $P' = \max\{c'^T x : x \in F'\}$ *tale che* $F \subset F'$ *e* $c'^T x \geq c^T x \forall x \in F$

Dato un problema (complesso) P sia P' un rilassamento di P deve essere possibile calcolare in tempi brevi la rispettiva soluzione s'. Dato che P' è un rilassamento di P conterrà solo un suo sottoinsieme di vincoli, di conseguenza è possibile che s' non sia ammissibile per P. Per questo a partire di P' si generano due o più sottoproblemi, a seconda dello schema adattato, in cui ricercare la soluzione. Nell'albero delle soluzioni (o albero di branch) la radice sarà rappresentata da P' e i nodi figli dai sottoproblemi prodotti dal nodo padre.

Gli algoritmi di tipo Branch and Bound si basano sui seguenti elementi:

- **bound**: determinare un *Bound* (o Upper Bound nel caso dei problemi di massimizzazione o Lower Bound nei problemi di minimizzazione), ovvero la bontà della soluzione prodotto da un sottoproblema, per evitare lo sviluppo completo di sottoalberi non promettenti.
- **branch**: costruzione dell'albero di branch, dopo la valutazione di un nodo, se la soluzione non è ammissibile per P deve è possibile realizzare nuovi nodi figli con delle restrizioni ulteriori che sperabilmente rendano ammissibile una futura soluzione prodotta dai nodi discendenti.
- sapere indetificare quando la soluzione prodotta da un sottoproblema P_i è ammissibile per P.

Ogniqualvolta viene trovata una nuova soluzione ammissibile per P, la si valuta e se è meglio di quella corrrente, aggiorniamo il valore della milgiore soluzione, se però la soluzione non è ammissibile per P la ricerca continua sviluppando i nodi figli. Nel caso in cui la valutazione di un nodo restituisce un valore non migliore rispetto alla migliore soluzione corrente allora non si procede con la generazione dei figli ma con la chiusura del nodo evitando così la generazione del sottoalbero ad esso associato.

Per svolgere l'operazione di branch necessitiamo di un qualche schema, che a partire da una una soluzione ammissibile per P_i permetta di generare sottoproblemi con ulteriori vincoli con lo scopo di rendere le loro soluzioni ammissibili per P.

E'possibile sviluppare una tecnica di tipo branch and bound per il TSP.

Sia x_i il set di variabili (istanziate) associato al modello PLI di una formulazione rilassata del TSP e $c(x_i) = LB_i$ una funzione che valuta la bontà di una soluzione associato al problema P_i. Di seguito lo pseudocodice di una generica procedura Branch and Bound per la risoluzione esatta del problema:

Algorithm 1 *BranchBoundTSP(P)*

Input: P instance of TSP problem
Output: z^* optimal objective function x^* optimal values of variables

```
1  P' ← Relaxation(P)
2  z* ← +∞, x* ← null, OpenNodes ← {P'}
3  x' ← Solve(P')
4  if x' is an hamiltonian cycle then
5      z* ← c(x) x* ← z'
6  while OpenNodes ! = ∅ do
7      P_i ← LowestLB(OpenNodes)
8      BranchingNodes ← Branch(P_i)
9      for every node x in BranchingNodes do
10         if x is an hamiltonian cycle and c(x) < z* then
11             z* ← c(x) x* ← x
12             remove from OpenNodes every P_i with LB(P_i) > z*
13         else if c(x) < z* then
14             OpenNodes ← OpenNodes ∪ {x}
15 return z*, x*
```

L'operazione di $Branch(P_i)$ genera e risolve i sottoproblemi figli di P_i. Si noti come ad ogni iterazione solo un sottoinsieme dei nodi prodotti dal branching vengono inseriti nell'insieme $OpenNodes$ in particolare solo quelli la cui soluzione non è un ciclo hamiltoniano e il valore della funzione obbiettivo è minore di quella attualemente memoriazzata. Ogni nodo P_i, dell'albero viene chiuso (nell'Algoritmo 1 non viene inserito nella lista dei nodi aperti) quando:

- $LB_i \leq z^*$.
- Non c'è soluzione ammissibile per P_i.
- l'1-albero T^* generato da P_i è una soluzione ammissibile migliore di z in quel caso $z^* = LB_i$.

Nella sezione successive sarà analizzato un rilassamento, compatibile con il TSP simmetrico, basato puramente su proprietà specifiche del problema e diversi schemi di braching.

4 Rilassamento 1-albero

Consideriamo una qualunque soluzione ammissibile per il TSP simmetrico: il numero di lati incidenti su ciascun vertice è pari a 2 e, immaginando di rimuovere i due archi incidenti su un vertice a piacere (ad esempio il vertice x), quel che rimane è un albero ricoprente per il sottografo indotto da $V \setminus x$. Pertanto, qualunque soluzione ha la seguente struttura:

1. l'insieme degli archi individuato crea un ciclo hamiltoniano;
2. ci sono due archi incidenti nel vertice x;

3. togliendo il vertice x rimane un albero ricoprente sul sottografo $V \setminus x$.

Il vincolo (1) è quello difficile da imporre, per cui lo si può rimuovere ed ottenere una soluzione rilassata nel seguente modo:

- calcolando il minimum spanning tree (MST) sul sottografo ottenuto eliminando il vertice x;
- aggiungendo i due archi di costo minimo incidenti in x.

La soluzione così ottenuta è definita **1-albero**.

Definizione. *Un 1-albero di un grafo $G(V, E)$ è un suo sottografo connesso $T(V, E')$, con $E' \subset E$, tale che:*

- *un nodo $x \in V$ ha grado 2*
- *T contiene esattamente un ciclo che passa per x*

Avendo rimosso il vincolo (1) abbiamo ottenuto la possibilità di ottimizzare sugli 1-alberi anzichè sui cicli hamiltoniani. La famiglia degli 1-alberi contiene la famiglia dei cicli hamiltoniani come sottoinsieme stretto, infatti **ogni ciclo hamiltoniano su G è anche un suo 1-albero, ma non viceversa** quindi il problema di determinare l'1-albero di costo minimo di G è un rilassamento del TSP.

Di seguito è riportato un possibile modello PLI per l'1-albero dove $x = 1$ che mette in evidenza la relazione con il problema del TSP.

$$\min \sum_{ij \in E} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{1j \in E} x_{1j} = 2 \qquad \forall j \in V \tag{9}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 1 \tag{10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{11}$$

Si noti come la famiglia dei vincoli (9) sono solo un sottoinsieme dei vincoli (7), in quanto si impone che solo sul nodo 1 incidano 2 archi.

Proprietà. *Se T^* è un 1-albero di costo minimo di G, allora $T[V \setminus x]$ è un minimum spanning tree di $G[V \setminus x]$.*

Se T^* è un 1-albero di costo minimo, $T[V \setminus x]$ deve essere un MST di $G[V \setminus x]$, altrimenti rimpiazzandolo con un MST si otterrebbe un 1-albero di costi inferiore a quello di T.
Un 1-albero T^* di costo minimo del grafo $G(V, E)$ è ottenibile completando un MST del sottografo $G[V \setminus x]$ e aggiungendo a T il nodo x connettendolo con i due archi a costo minore possibile, di conseguenza è possibile risolvere il problema dell'MST sul grafo $G(V, E)$ e con un accorgimento ottenere un 1-albero valido.

Anche il problema del MST può essere espresso come modello di PLI

$$\min \sum_{ij \in E} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{ij \in E} x_{ij} = n - 1 \qquad \forall j \in V \tag{12}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 1 \tag{13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{14}$$

Il modello del MST mette in evidenza la relazione con il modello dell'1-albero, riportano la stessa funzione obbiettivo e una famiglia di vincoli analoghi. L'unica differenza è data dai vincoli (12) in quanto nel modello l'MST forzano la soluzione ad avere cardinalità pari a $n - 1$ dove $n = |V|$, tale vincolo è unico e risulta meno stringente della famiglia dei vincoli (9) del modello PLI dell'1-albero.

Calcolo del lower bound per il problema originario Possiamo utilizzare la seguente procedura per realizzare un 1-albero ottimo T^*

- **Passo 1.** Si risolva il problema MST sul grafo ottenuto scartando da $G = (V, A)$ il nodo x e tutti gli archi incidenti su di esso. Sia A_T la soluzione trovata;
- **Passo 2.** Si aggiungano ad A_T i due archi (x, k) e (x, h) a distanza minima tra tutti quelli incidenti sul nodo x .
- **Passo 3** Si restituisca $T^* = (V, E')$ con $E' = A_T \cup (x, k); (x, h)$.

In conclusione, l'1-albero di costo minimo fornisce un lower bound per il valore dell'ottimo del TSP simmetrico, il valore della soluzione trovata sarà sempre minore o uguale al valore della soluzione ottima del problema di partenza: $LB = c(T) \leq c(C^*)$, dove C^* è il ciclo hamiltoniano ottimo.

5 Branching

Una soluzione ottima del rilassamento 1-albero non soddisfa in generale, i vincoli (9) per qualche nodo, per cui un algoritmo branch-and-bound basato sul rilassamento 1-albero deve prevedere uno schema di branching che consenta di la generazione dei sottoproblemi P_i, che mirino a risolvere la non ammissibilità dettata dei vincoli.

Supponiamo quindi di avere a disposizione un 1-albero ottimo T^*. Dato che T^* è un 1-albero, per definizione contiene almeno un ciclo, se siamo fortunati tale ciclo è il ciclo hamilitaniano possiamo aggiornare il valore di z^* e chiudere il nodo, altrimenti possiamo quindi supporre che esista un nodo i in cui incidano piu' di due archi di T^*. Siano $(a_1, a_2)(a_2, a_3)..., (a_k, a_1)$ questi archi.

Un semplice modo per effettuare il branching corrisponde a selezionare uno qualsiasi di questi archi ed a costruire due figli, uno nel quale si è deciso che l'arco faccia parte del ciclo Hamiltoniano e l'altro in cui si è deciso che non ne faccia parte, similmente a quanto avviene nel Branch and Bound ad hoc per il problema dello zaino.

Questa regola è semplice da implementare, potenzialmente equipartiziona l'insieme (ma si noti che ogni nodo ha solo due archi incidenti nel ciclo e molti archi incidenti che non appartengono al ciclo, quindi in uno dei due figli viene presa una decisione "piu' forte") e crea esattamente due figli per nodo. Per contro, nel figlio in cui si

decide che l'arco fa parte del ciclo, la soluzione ottima del rilassamento non cambia. Con questo schema ad ogni nodo discenscente stiamo fissando alcuni archi come presenti o assenti a priori, per tenerne conto ad ogni sottoproblema sono associati due insiemi $E_0 = \{\forall x_{ij} \in E_0 : x_{ij} = 0\}$ e $E_1 = \{\forall x_{ij} \in E_0 : x_{ij} = 1\}$ che rappresentano rispettivamente l'insieme degli fissati a priori come o assenti o presenti.

5.1 Branching sugli archi nel ciclo del 1-tree

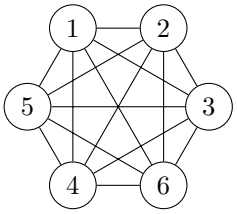
E' possibile sviluppare una tecnica più sofisticata sempre utilizzando gli archi nel ciclo presenti in T^*, tale branching prevede di costruire k figli del nodo: in ciascuno dei figli si è fissato uno degli archi $(a_1, a_2)..., (a_k, a_1)$ come non appartenente al ciclo. Si noti che k è il numero degli archi nel ciclo di T^*.

Il nostro scopo è fornire una regola di suddivisione il cui scopo è quello di impedire il formarsi dei ciclo identificato nei nodi figli. Il primo nodo figlio viene ottenuto imponendo che in esso non sia presente l'arco (a_1, a_2) (cioè si impone $x_{a_1,a_2} = 0$), il secondo nodo figlio viene ottenuto imponendo che sia presente l'arco (a_1, a_2) ma non sia presente l'arco (a_2, a_3) (cio´e si impone $x_{a_1,a_2} = 0, x_{a_2,a_3} = 1$), e così via fino al k-esimo figlio in cui si impone che siano presenti tutti gli archi (a_j, a_{j+1}) con $j = 1, ..., k-2$, ma non sia presente l'arco (a_k, a_1).
Il seguente ciclo $(a_1, a_2), (a_2, a_3), (a_3, a_4), (a_4, a_1)$ genererebbe 4 problemi figli con i seguenti insiemi E_0 e E_1.

- Nodo 2: $E_0 = \{(a_1, a_2)\}, E_1 = \emptyset$
- Nodo 3: $E_0 = \{(a_2, a_3)\}, E_1 = \{(a_1, a_2)\}$
- Nodo 4: $E_0 = \{(a_3, a_4)\}, E_1 = \{(a_1, a_2), (a_2, a_3)\}$
- Nodo 5: $E_0 = \{(a_4, a_1)\}, E_1 = \{(a_1, a_2), (a_2, a_3), (a_2, a_3)\}$

In altre parole si fa in modo che in ciascuno dei figli sia assente almeno un arco del ciclo, il che esclude la presenza di tale ciclo in ciascuno dei nodi figli.

Esempio 1 *Si consideri l'istanza di TSP simmetrico corrispondente ad un grafo completo $G(V, E)$ da 6 nodi indicato in Figura 1, la non orientazione degli permette la rappresentazione dei costi c_{ij} mediante una matrice triangolore bassa.* Sarà



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \infty & & & & & \\ 4 & \infty & & & & \\ 3 & 6 & \infty & & & \\ 7 & 9 & 8 & \infty & & \\ 5 & 1 & 4 & 2 & \infty & \\ 9 & 2 & 3 & 4 & 7 & \infty \end{pmatrix} \end{array}$$

Come primo passo si calcola il rilassamento al nodo radice (nodo 1). Utilizzando, arbitrariamente, $x = 1$, si determina l'1-tree di costo minimo formato dagli archi $T_1^* = \{(1, 2), (1, 3), (2, 5), (2, 6), (3, 6), (4, 5)\}$, dove $A_{T1} = \{(2, 5), (2, 6), (3, 6), (4, 5)\}$ è l'albero di supporto a costo minimo determinato per il sottografo indotto dai nodi $V \setminus 1$, e $(1, 2), (1, 3)$ sono i due archi di costo minimo incidenti sul nodo $x = 1$.

Risulta $LB_1 = \sum_{(i,j)\in T_1} c_{ij} = 15$. Si osserva che questo 1-tree non corrisponde ad una soluzione ammissibile, in quanto esso contiene il ciclo $(1, 2, 6, 3, 1)$ — archi

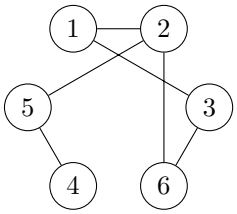


Figura 1: 1-albero T_1

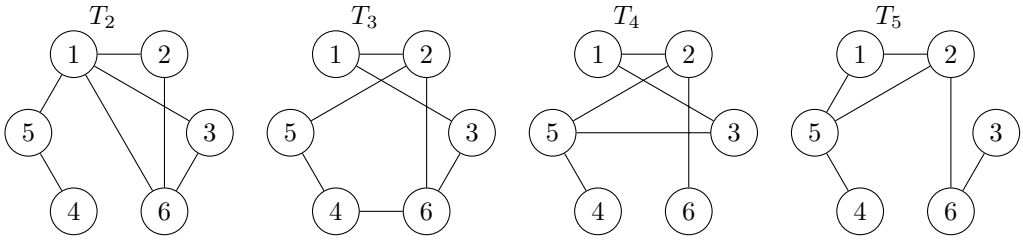
$(1,2),(2,6),(3,6),(1,3)$.
Dato che è un problema di minimo, si pone quindi il lower bound iniziale $z = +\infty$ e si procede al branch. Il branch del nodo 1 produce i seguenti nodi figli, con i rispettivi insiemi E_0, E_1.

- Nodo 2: $E_0 = \{(1,2)\}, E_1 = \emptyset$
- Nodo 3: $E_0 = \{(2,6)\}, E_1 = \{(1,2)\}$
- Nodo 4: $E_0 = \{(3,6)\}, E_1 = \{(1,2),(2,6)\}$
- Nodo 5: $E_0 = \{(1,3)\}, E_1 = \{(1,2),(2,6),(3,6)\}$

Tenuto conto dei vincoli imposti da E_0, E_1, si generano gli 1-alberi di costo minimo associati ai rispettivi nodi:

- $T_2 = \{(1,3),(1,5),(1,2),(2,6),(3,6),(4,5)\}, c(T_2) = 16.$
- $T_3 = \{(1,2),(1,3),(2,5),(3,6),(4,5),(4,6)\}, c(T_3) = 17.$
- $T_4 = \{(1,2),(1,3),(2,5),(2,6),(3,5),(4,5)\}, c(T_4) = 16.$
- $T_5 = \{(1,2),(1,5),(2,5),(2,6),(3,6),(4,5)\}, c(T_4) = 17.$

Di seguito una rappresentazione dei T_i generati ai rispettivi nodi.

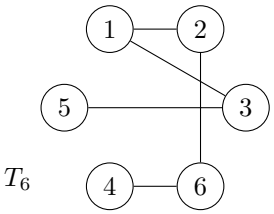


Valutando in sequenza i nodi ottenuti si nota che al nodo 3 l'insieme di archi in T_3 è un ciclo hamiltoniano di costo $z_3 = 17 < z$. Quindi si pone $z = 17$ e si salva quest'ultima come migliore soluzione ammissibile nota. Il nodo 3 viene chiuso per ottimalità. Successivamente, valutando il nodo 5, risulta $z_5 = 17 \leq z$, quindi il nodo 5 viene chiuso. Rimangono aperti i nodi 2 e 4. Si può scegliere il nodo 4 percontinuare l'esplorazione. La soluzione del rilassamento del nodo 4 presenta il ciclo $(1,3,5,2,1)$ creato dagli archi $(2,5),(3,5),(1,3),(1,2)$.
L'arco $(1,2)$ è già fissato; per quanto riguarda i rimanenti archi, si ottengono i seguenti nodi.

- Nodo 6: $E_0 = \{(3,6),(2,5)\}, E_1 = \{(1,2),(2,6)\}$
- Nodo 7: $E_0 = \{(3,6),(3,5)\}, E_1 = \{(1,2),(2,6),(2,5)\}$

- Nodo 8: $E_0 = \{(3,6),(1,3)\}, E_1 = \{(1,2),(2,6),(2,5),(3,5)\}$

Il nodo 6 genera una soluzione ammissibile in quanto
$T_6 = \{(1,2),(1,3),(2,6),(3,5),(3,5),(4,6)\}$ è un circuito hamiltoniano
ma $z_6 = 19 \geq z$ quindi il nodo viene chiuso, mentre i nodi 7 e 8, senza necessità di risolvere il rilassamento, vengono chiusi per non-ammissibilita‘ in quanto il nodo 2 del grafo presenta grado 3 causato dagli archi in E_1.

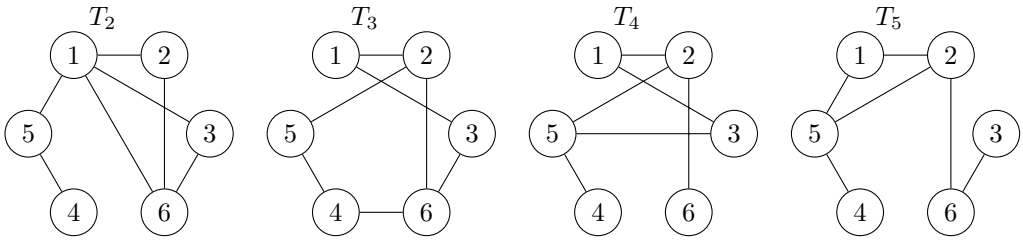


Rimane aperto il nodo 2 la soluzione del rilassamento del nodo 2 presenta il ciclo $(2,6,3,1,5,2)$ formato dagli archi $\{(2,6),(3,6),(1,3),(1,5),(2,5)\}$. Il braching porta alla creazione dei seguenti nodi figli.

- Nodo 9: $E_0 = \{(1,2),(2,6)\}, E_1 = \{\emptyset\}$
- Nodo 10: $E_0 = \{(1,2),(3,6)\}, E_1 = \{(2,6)\}$
- Nodo 11: $E_0 = \{(1,2),(1,3)\}, E_1 = \{(2,6),(3,6)\}$
- Nodo 11: $E_0 = \{(1,2),(1,5)\}, E_1 = \{(2,6),(3,6),(1,3)\}$
- Nodo 11: $E_0 = \{(1,2),(2,5)\}, E_1 = \{(1,2),(2,6),(1,3),(1,5)\}$

Valutandoli in sequenza, risulta:

- $T_9 = \{(1,3),(1,5),(2,5),(3,6),(4,5),(4,6)\}, c(T_9) = 18.$
- $T_{10} = \{(1,3),(1,5),(2,5),(2,6),(3,4),(4,5)\}, c(T_{10}) = 17.$
- $T_{11} = \{(1,4),(1,5),(2,5),(2,6),(3,5)\}, c(T_{11}) = 20.$
- $T_{12} = \{(1,3),(1,4),(2,5),(2,6),(3,6),(4,5)\}, c(T_{12}) = 18.$
- $T_{13} = \{(1,3),(1,5),(2,6),(3,5),(3,6),(4,6)\}, c(T_{12}) = 21.$



Tutti questi nodi vengono chiusi in quanto producono una soluzione di valore pari o superiorie a quella corrente. Non essendoci più nodi aperti, la soluzione corrispondente ottima, con ottimo $z^* = 17$.
Il numero di nodi generabili in un possibile albero di branch per un qualsiasi problema è decisamente elevato, esponenziale nelle dimensioni del problema. Con questo schema siamo riusciati a trovare il valore ottimo dell'istanza in esame creando solo con 13 nodi nell'albero di branch.

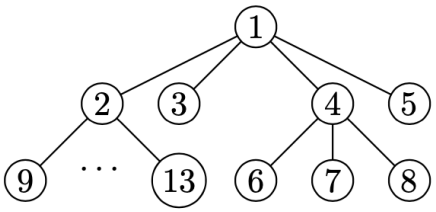


Figura 2: Branching Tree

Considerazioni. Negli schemi proposti si analizza il ciclo presente nell'1-albero ottenuto dal rilassamento di P, se questo è un ciclo hamiltoniano allora siamo davanti ad una possibile soluzione, altrimenti si prodece con un'aggiunta incrementale di vincoli nei sottoproblemi figli in modo che possano generare una soluzione in qualcuno dei sottoproblemi P_i discendenti. Per realizzare uno schema di branching quindi è importante capire cosa c'è che non va nella soluzione rilassata e come recuperare l'ammissibilità da tale soluzione.

5.2 Branching sul grado dei nodi

Un altro fattore che possa rendere non ammissible la nostra soluzione è il grado dei nodi, sappiamo che in un ciclo hamiltoniamo ogni nodo deve avere grado 2, mentre nella formulazione del problema dell'1-albero imponiamo che solo il nodo x deve avere grado 2 (con il vincolo (9)).
Data un 1-albero ottimo T^* si sceglie un vertice $v \in T$ con grado ≥ 3 e due lati a e b incidenti su v; il problema è suddiviso in tre sottoproblemi ottenuti vietando, a turno, il lato a, il lato b, oppure tutti gli altri lati incidenti su v in modo da rispettare i vincoli di grado ogni nodo a 2.

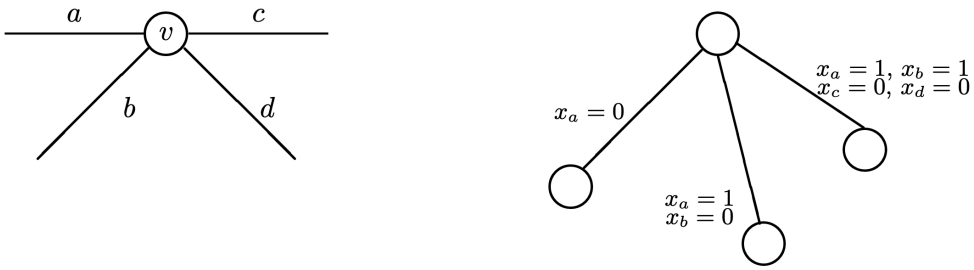


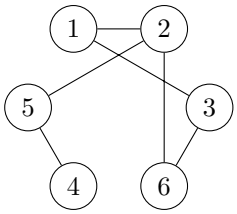
Figura 3: Schema branching su grado dei nodi

Si riprenda in considerazione l'istanza utilizzata nell'**Esempio 1** alla sezione precedente e la soluzione al suo rilassamento 1-albero mostrata in Figura 4.

Si osserva che questo 1-tree non corrisponde ad un tour ammissibile, in quanto sul nodo 2 incidono 3 archi, si procede quindi al branch.

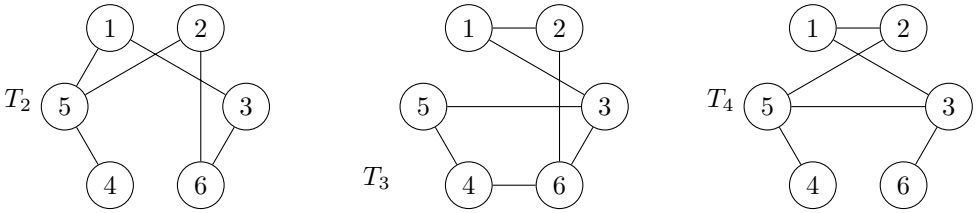
Il branch produce i seguenti nodi figli, con i rispettivi insiemi E_0, E_1.

- Nodo 2: $E_0 = \{(1,2)\}, E_1 = \emptyset$
- Nodo 3: $E_0 = \{(2,5)\}, E_1 = \{(1,2)\}$
- Nodo 4: $E_0 = \{(2,6)\}, E_1 = \{(1,2),(2,5)\}$



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \infty & & & & & \\ 4 & \infty & & & & \\ 3 & 6 & \infty & & & \\ 7 & 9 & 8 & \infty & & \\ 5 & 1 & 4 & 2 & \infty & \\ 9 & 2 & 3 & 4 & 7 & \infty \end{pmatrix} \end{array}$$

Figura 4: 1-albero T_1



Ai quali corrispondono i rispettivi 1-alberi.

- $T_2 = \{(1,3),(1,5),(2,5),(2,6),(4,5),(2,6)\}$
- $T_3 = \{(1,3),(1,2),(2,6),(3,5),(4,5),(4,6)\}$
- $T_4 = \{(1,2),(1,3),(2,5),(3,6),(5,4),(3,5)\}$

Si osserva che nessuno di questo 1-albero corrisponde ad un ciclo hamiltoniano, in quanto su ogni albero c'è sempre un nodo con grado superiore a 2, il branch quindi procedede iterativamente su T_2, T_3, T_4.

6 Conclusioni

In questo eleborato è stato discusso il problema del TSP e una procedura esatta di Branch and Bound per la sua risoluzione. La tecnica Branch and Bound può essere applicata a diversi problemi NP-Completi, ma non può essere utilizzata direttamente, prima deve esserci una profonda comprensione del problema, solo così è possibile definire un rilassamento efficiente e uno schema di branching ad hoc per il problema in esame. Lo schema di branching e il rilassamento sono fortemente correlati, in quanto solo dopo aver compreso i 'difetti' della soluzione prodotta dal rilassamento è possibile ideare uno schema di branch che generi dei sottoproblemi la cui soluzione sperabilemente non contenga i 'difetti' della soluzione rilassata.

Riferimenti bibliografici

[1] The TSP, Chapter 7
http://math.mit.edu/ goemans/18453S17/TSP-CookCPS.pdf

[2] L. De Giovanni, M. Di Summa, Unipd
https://www.math.unipd.it/ luigi/courses/metmodoc1314/m08.01.TSPexact.pdf

[3] Jens Clausen IMM
https://www2.imm.dtu.dk/courses/02713/TSP.Eng.Wolsey.pdf

[4] Michele Monaci, Dipartimento di Ingegneria, Università di Padova
https://www.infologis.biz/wp-content/uploads/2009/08/tsp03.pdf

[5] Appunti di ricerca operativa Unito
https://www.di.unito.it/ locatell/didattica/ro2/ro2.pdf
https://www.di.unito.it/ locatell/didattica/ro2/Branch-TSP-sl.pdf