Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Matematica

Metodi di Programmazione Dinamica per il Traveling Salesman Problem con vincoli di precedenza

Tesi di Laurea in Programmazione Matematica

Relatore:
Chiar.mo Prof.
ARISTIDE MINGOZZI

Presentata da:
ELEONORA FONTANA

VI Sessione
Anno Accademico 2017-18

Alla mia famiglia

Indice

Introduzione

Il Traveling Salesman Problem (TSP), ovvero il Problema del Commesso Viaggiatore, è un problema di ottimizzazione combinatoria che si può formulare nel seguente modo. Siano date n città e i costi per spostarsi da una città all'altra. Un salesman (agente di commercio), partendo da una delle città, deve compiere un tour in cui visita una ed una sola volta ciascuna città ritornando alla città di partenza. L'obiettivo è minimizzare il costo del tour. Aggiungendo al TSP i cosiddetti vincoli di precedenza, si parla di Traveling Salesman Problem with Precedence Constrains (TSPP), noto in letteratura anche come Sequential Ordering Problem (SOP). Questi vincoli impongono che determinate città possano essere visitate solamente dopo averne visitate delle altre, specificate nei dati del problema.

Non esistono algoritmi efficienti per la risoluzione del TSP, l'unico metodo di risoluzione consiste infatti nell'esaminare tutti i possibili cammini per poi scegliere quello migliore. Tuttavia, questa operazione ha una complessità esponenziale ed è quindi inattuabile per problemi reali in cui il valore di n è solitamente elevato. È stato dimostrato che TSP è un problema NP-hard. Da questo segue che anche il TSPP è un problema NP-hard, poichè il TSP non è altro che un caso particolare di TSPP in cui non ci sono precedenze da rispettare.

Il TSPP è un modello per molti problemi reali sia nella logistica dei trasporti sia nelle aziende manifatturiere, come i seguenti:

- determinare un percorso, in cui una o più raccolte devono precedere la consegna ad un cliente,
- determinare l'itinerario di un braccio meccanico per un sistema di immagazzinamento automatico,
- determinare uno scheduling in cui un determinato lavoro deve essere completetato prima che altri possano iniziare.

Nonostante il TSP sia uno dei problemi di ottimizzazione più trattati, non si trova molto in letteratura sul TSPP. La prima menzione di quest'ultimo, posto in una forma un po' diversa, è da attribuire ad Escudero et al.[10, 11], i quali descrivono un euristico da impiegare in un sistema di produzione e che ottiene buoni risultati pratici rispetto

al tempo di calcolo e alla qualità della soluzione. Ascheuer [1] e Ascheuer et al. [4, 3] usano un approccio *cutting-plane* per ottenere un lower bound al costo della soluzione ottima. Questi lower bound sono stati poi usati per verificare la qualità della soluzione ottenuta dall'euristico di Escudero. Questi autori confrontano tre diversi modelli e danno risultati computazionali su dati reali di IBM forniti da Escudero. Il modello presentato da Ascheuer et al. [5] porta a dei buoni bound e risulta essere migliore degli altri modelli da un punto di vista computazionale. Escudero et al. [12] rafforzano il modello introducendo nuove classi di disuguaglianze e descrivendo un approccio Lagrangiano di *relax-and-cut* con cui, in alcuni casi, ottengono migliori lower bound di quelli dati da Ascheuer et al [4, 3]. Nello stesso periodo viene sviluppato un modello simile da Balas et al. [26, 27]. Prendendo in considerazione il caso simmetrico, l'obiettivo della ricerca è descrivere un euristico che determini uno scheduling per elicotteri che devono raggiungere piattaforme petrolifere in un certo ordine, minimizzando la distanza totale percorsa. Inoltre, Gouveia et al. [15] combinano formulazioni basate su variabili di precedenza con formulazioni basate su sistemi di flusso, al fine di ottenere nuove disuguaglianze valide per il TSPP.

Contributi della tesi

In questa tesi vengono presentati i metodi di risoluzione del Traveling Salesman Problem con vincoli di Precedenza (TSPP) implementati da Ascheuer et al. [5] e Gouveia et al. [15]. I risultati ottenuti da questi due autori sono poi confrontati con quelli conseguiti mediante un metodo originale di programmazione matematica proposto in questa tesi.

I contributi originali per la risoluzione del TSPP proposti in questa tesi sono i seguenti.

- Una nuova formulazione matematica a numeri interi.
- Una formulazione mediante la Programmazione Dinamica (DP).
- Un metodo originale di rilassamento dello spazio degli stati della ricursione esatta di DP, denominato *kL-path*, che tiene parzialmente conto dei vincoli di precedenza. Tale metodo consente di ottenere un valido lower bound al TSPP, denominato $LB2$, che viene massimizzato mediante il metodo del subgradiente.
- Viene inoltre proposto un metodo di risoluzione del TSPP basato sulla ricursione di DP che usa il nuovo lower bound per eliminare stati che non possono condurre alla soluzione ottima e limita l'insieme degli stati esplorati. Tale metodo non garantisce di trovare la soluzione ottima del TSPP ma restituisce sempre un lower bound superiore a $LB2$.
- I nuovi metodi proposti sono stati valutati su un insieme di problemi test. I risultati di calcolo ottenuti mostrano l'efficienza dei nuovi metodi proposti rispetto a quelli ottenuti dai metodi noti in letteratura.

Capitolo 1

Formulazioni matematiche

1.1 Notazione

Esistono naturalmente diverse formulazioni possibili, le quali dipendono dalla scelta delle variabili decisionali e dal modo di esprimere i vincoli. In questo capitolo, dopo aver introdotto una notazione opportuna, saranno presentate le formulazioni matematiche proposte da Ascheuer et al. [5] e Gouveia et al. [15] e una terza formulazione su cui è basato il lower bound proposto in questa tesi.

Sia $G = (V, A)$ un grafo orientato, con $V = \{1, 2, ..., n\}$ insieme di n vertici ed A insieme di m archi. Ad ogni arco $(i, j) \in A$ è associato un costo $c_{ij} > 0$ e ad ogni vertice $i \in V$ è associato un insieme di relazioni di precedenza, definito dal sottoinsieme di vertici $P_i \subset V' = V \setminus \{1\}$. In seguito si assumerà che se $j \in P_i$ allora A non contiene l'arco (i, j) e $c_{ij} = \infty$. Si porrà inoltre $P_1 = \emptyset$ e $1 \notin P_i \ \forall i \in V$ e si supporrà che $P_j \subset P_i$ $\forall j \in P_i$, per ogni vertice $i \in V'$.

Con questa notazione, si può riformulare il TSPP come il problema di determinare un tour hamiltoniano di costo minimo nel grafo G che parta dal vertice 1, visiti ogni vertice $i \in V'$ dopo aver visitato tutti i vertici di P_i e ritorni al vertice 1.

Sia $\tilde{G} = (\tilde{V}, \tilde{A})$ il grafo delle precedenze associato ai sottoinsiemi P_i con $i \in V$, dove $\tilde{V} = \{i \in V;\ |P_i| > 0\}$ e $\tilde{A} = \left\{(i, j);\ i \in P_j, j \in \tilde{V}\right\}$. In seguito si assumerà che $\tilde{G}$ sia un grafo aciclico e che il problema abbia almeno una soluzione ammissibile.

Esempio. Si consideri l'insieme di vertici $V = \{1, ..., 12\}$ riportati in figura, in cui il costo c_{ij} è dato dala distanza euclidea tra il vertice i e il vertice j. Nella figura 1 le frecce indicano le relazioni di precedenza che devono essere rispettate.

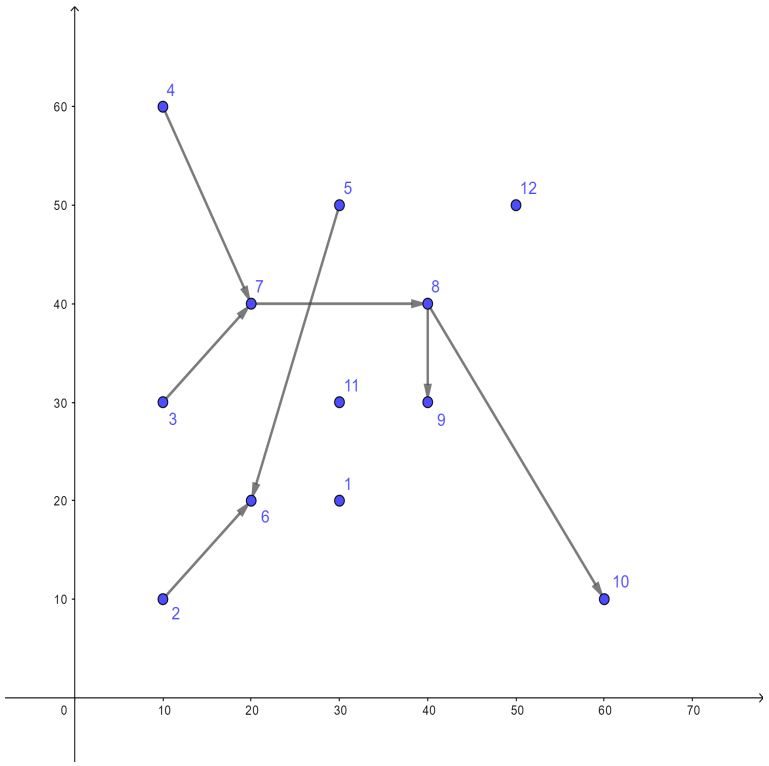


Fig. 1: Grafo $\tilde{G}$ delle precedenze

In questo caso quindi si avrà:

- $P_1 = \emptyset$,
- $P_2 = \emptyset$,
- $P_3 = \emptyset$,
- $P_4 = \emptyset$,
- $P_5 = \emptyset$,
- $P_6 = \{2, 5\}$,
- $P_7 = \{3, 4\}$,
- $P_8 = \{7\}$,

- $P_9 = \{8\}$,
- $P_{10} = \{8\}$,
- $P_{11} = \emptyset$,
- $P_{12} = \emptyset$.

1.2 Formulazione di Ascheuer et al. [5]

Si introducano le variabili decisionali x_{ij} con $(i,j) \in A$, definite come segue:

$$x_{ij} = \begin{cases} 1 & \text{se l'arco } (i,j) \text{ è in soluzione} \\ 0 & \text{altrimenti.} \end{cases}$$

La formulazione matematica (P1) del TSPP riportata da Ascheuer et al. [5] è la seguente.

$$z(\mathrm{P1}) = \min \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{1.2.1}$$

$$\text{s. t.} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{1.2.2}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \tag{1.2.3}$$

$$\sum_{i,j\in W} x_{ij} \leq |W| - 1, \qquad \forall\, W \subset V,\, |W| \geq 2 \tag{1.2.4}$$

$$\sum_{k\in W} x_{jk} + \sum_{l,k\in W} x_{lk} + \sum_{l\in W} x_{li} \leq |W|, \qquad \forall (i,j) \in \tilde{A},\, \forall\, W \subseteq V' \setminus \{i,j\},\, |W| \neq \emptyset \tag{1.2.5}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{1.2.6}$$

Nel dettaglio, i vincoli di questa formulazione hanno i seguenti significati:

- (A.1.9) e (1.2.3) impongono che ciascun vertice sia visitato una ed una sola volta.
- (1.2.4) assicurano l'eliminazione di sub-tour. Infatti se esistesse un sub-tour $(i_1,\ldots,i_r, i_{r+1} = i_1)$ associato ad un sottoinsieme di vertici S, si avrebbe

$$\sum_{j,k\in S} x_{jk} = r + 1.$$

Scegliendo $W = S$ si avrebbe quindi una contraddizione.

- (1.2.5) impongono che le precedenze siano rispettate. Se infatti ci fosse una precedenza $(i,j) \in \tilde{A}$ non rispettata, allora si avrebbe una soluzione del tipo (1,..., j, k,...,l,i,...,1) dove $l,k \in V' \setminus \{i,j\}$. Scegliendo W uguale all'insieme dei vertici che in soluzione sono tra j e i, (1.2.5) diverebbe $1 + |W| + 1 \leq |W|$ e non sarebbe ovviamente rispettata.

1.3 Formulazione di Gouveia et al. [15]

Le variabili decisionali x_{ij}, $(i,j) \in A$, definite nella sezione precedente, vengono utilizzate anche da Gouveia et al. [15] per proporre la seguente formulazione del TSPP. Si introducono inoltre le variabili decisionali v_i^j con $(i,j) \in A \setminus \{1\}$, definite da Gouveia e Pires [16] come:

$$v_i^j = \begin{cases} 1 & \text{se il vertice } i \text{ precede il vertice } j \text{ nel tour} \\ 0 & \text{altrimenti.} \end{cases}$$

La formulazione (P2) proposta di Gouveia et al. [15] è quindi la seguente.

$$z(\mathrm{P2}) = \min \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{1.3.1}$$

$$\text{s. t.} \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \tag{1.3.2}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \tag{1.3.3}$$

$$x_{ij} \leq v_i^j \qquad \forall (i,j) \in A,\ i,j \neq 1 \tag{1.3.4}$$

$$v_k^j + v_j^i + x_{ij} \leq v_k^i + 1 \qquad \forall (i,j,k) \text{ sequenza di tre vertici distinti in } V' \tag{1.3.5}$$

$$v_i^j + v_j^i = 1 \qquad \forall i,j \in V',\ i \neq j \tag{1.3.6}$$

$$v_i^j = 1 \qquad \forall (i,j) \in \tilde{A} \tag{1.3.7}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{1.3.8}$$

$$v_i^j \in \{0,1\}, \qquad \forall i,j \in V',\ i \neq j \tag{1.3.9}$$

Nel dettaglio, i vincoli di questa formulazione hanno i seguenti significati:

- (1.3.2) e (1.3.3) impongono che ciascun vertice sia visitato una ed una sola volta.
- (1.3.4), (1.3.5) e (1.3.6) assicurano l'eliminazione di sub-tour. Infatti se esistesse un sub-tour (i_1,..., i_{r-1}, i_r, $i_{r+1} = i_1$), scegliendo $i = i_r$, $j = i_{r+1}$ e $k = i_{k-1}$ la (1.3.5) diverrebbe
$$v_k^j + v_j^i + x_{ij} = 3 \leq v_k^i + 1 = 2$$
e si avrebbe ovviamente una contraddizione.

- (1.3.7) impongono che le precedenze siano rispettate.

1.4 Una terza formulazione

Si definisca $F_i = \left\{ j \in \tilde{V}; i \in P_j \right\}$ il sottoinsieme di vertici che devono essere preceduti da $i \in V$. Siano $np_i = |P_i|$ e $nf_i = |F_i|$, con $i \in V$. Si consideri $H = (i_1, i_2, ..., i_k, ..., i_n, i_{n+1})$ una soluzione del TSPP dove $i_1 = i_{n+1} = 1$. Diremo che l'arco (i_k, i_{k+1}) è in posizione k, $k = 1, ..., n$ (i.e. $(1, i_2)$ è in posizione $k = 1$ e l'arco $(i_n, 1)$ è in posizione n). Inoltre ogni vertice $i \in P_j$ occupa una posizione in H di indice inferiore a quella occupata dal vertice j. Dalla definizione di np_i e nf_i segue che ogni vertice $i \in V$ potrà occupare nel tour H solo le posizioni k tali che $np_i + 2 \leq k \leq n - nf_i$.

Siano V'_k e A_k rispettivamente il sottoinsieme dei vertici e degli archi che possono occupare la posizione $k = 1, ..., n$, definiti come segue

- $V'_1 = V'_{n+1} = \{1\}$
- $V'_k = \{i \in V';\ np_i + 2 \leq k \leq n - nf_i\}$ con $k = 2, ..., n$
- $A_1 = \{(1, j) \in A;\ j \in V'_2\}$
- $A_k = \{(i, j) \in A;\ i \in V'_k,\ j \in V'_{k+1}\}$ con $k = 2, ..., n$.

Si introducano le variabili decisionali x_{ijk} con $(i, j) \in A_k$ e $k = 1, ..., n$, definite come:

$$x_{ijk} = \begin{cases} 1 & \text{se l'arco } (i,j) \text{ è in posizione } \ k \\ 0 & \text{altrimenti} \end{cases}$$

Una formulazione (P3) del TSPP è la seguente:

$$z(\text{P3}) = \min \sum_{k=1}^{n} \sum_{(i,j) \in A_k} c_{ij}\, x_{ijk} \tag{1.4.1}$$

$$\text{s. t.} \sum_{j \in V'_2} x_{1j1} = 1 \tag{1.4.2}$$

$$\sum_{j \in V'_n} x_{j,1,n+1} = 1 \tag{1.4.3}$$

$$\sum_{j \in V'_{k+1}} x_{ijk} - \sum_{j \in V'_{k-1}} x_{j,i,k-1} = 0, \qquad \forall k = 2, ..., n, \forall i \in V'_k \tag{1.4.4}$$

$$\sum_{k=np_j+1}^{n-nf_j-1} \sum_{i \in V'_k} x_{ijk} = 1, \qquad \forall j \in V' \tag{1.4.5}$$

$$\sum_{k=np_i+2}^{n-nf_i} \sum_{j \in V'_{k+1}} x_{ijk} = 1, \qquad \forall i \in V' \tag{1.4.6}$$

$$\sum_{(i,j) \in A_k} x_{ijk} = 1, \qquad \forall k = 1, ..., n \tag{1.4.7}$$

$$\sum_{k=np_j+2}^{n-nf_j} \sum_{r \in V'_{k+1}} k\, x_{jrk} - \sum_{k=np_i+2}^{n-nf_i} \sum_{r \in V'_{k+1}} k\, x_{irk} \geq 1, \quad \forall (i,j) \in \tilde{A} \tag{1.4.8}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall (i,j) \in A_k,\ \forall k = 1, ..., n \tag{1.4.9}$$

Il significato dei vincoli è il seguente:

- (1.4.2) e (1.4.3) impongono che ogni soluzione ammissibile parta e termini nel vertice 1.
- (1.4.4) impongono che se il vertice i è in posizione $k > 1$ (i.e. $\sum_{j=2}^{n} x_{ijk} = 1$) allora deve essere in soluzione un arco (j, i) in posizione $k - 1$ (i.e. $\sum_{j=1}^{n} x_{j,i,k-1} = 1$). Questi sono detti vincoli di continuità.
- (1.4.5) e (1.4.6) impongono che ciascun vertice sia visitato una ed una sola volta.
- (1.4.7) impongono che in ciascuna posizione vi sia un solo arco. Si può dimostrare che i vincoli (1.4.7) sono ridondanti ovvero che ogni soluzione che soddisfa (1.4.2)-(1.4.6) soddisfa anche (1.4.7).
- (1.4.8) impongono che per ogni precedenza $(i, j) \in \tilde{A}$ il vertice i occupi nel tour hamiltoniano una posizione di indice inferiore a quella occupata dal vertice j.

Capitolo 2

Metodo di Ascheuer et al. [5]

Il metodo utilizzato da Ascheuer et al. [5] consiste in un algoritmo di tipo *branch & cut*. Una delle caratteriste principali degli algoritmi di questo tipo è il calcolo simultaneo di upper e lower bound dell'ottimo. Ascheuer et al. [5] ottiengono i primi mediante un euristico, mentre i secondi come il risultato di un algoritmo *cutting plane*. Nel caso in cui upper e lower bound coincidano, è dimostrata l'ottimalità della soluzione.

2.1 Calcolo dell'upper bound

Il calcolo di soluzioni ammissibili avviene in due punti dell'algoritmo proposto da Ascheuer et al. [5]: nella fase di inizializzazione e dopo ogni risoluzione di un problema di Programmazione Lineare (LP). Si osserva che, per alcuni problemi, è molto difficile trovare buone soluzioni mediante l'euristico iniziale. Soprattutto in questi casi, è necessario migliorare la soluzione sfruttando le informazioni sulla struttura di una soluzione ottimale, contenuta nella soluzione LP. Questa tecnica è nota come *LP-exploitation*.

Ascheuer et al. [5] implementano una serie di versioni modificate di euristici noti per il TSP, per poi eseguire un ampio confronto computazionale tra queste, concludendo che danno risultati piuttosto scarsi. Migliori soluzioni sono ottenute dall'euristico di *LP-exploitation*. Pertanto, non viene dedicato molto tempo alla costruzione di una soluzione iniziale e la si ricava mediante un euristico *nearest neighbor*. Si tratta di un algoritmo costruttivo, la cui fase di inizializzazione consiste nella scelta di un tour parzialmente ammissibile T, contenente solo un arco $(1, i)$. Per espandere la soluzione, si considera j, l'ultimo vertice nel tour, e si aggiunge un vertice k tale che tutti i predecessori di k siano già contenuti in T e tale che l'arco (j, k) abbia costo minimo tra tutti gli archi ammissibili (j, l). L'algoritmo viene ripetuto, inizializzando la soluzione con ognuno degli archi ammissibili $(1, i) \in A$. Successivamente, la sequenza migliore viene passata ad un euristico *3-opt* modificato, che rifiuta i cambiamenti non ammissibili della soluzione

migliore attuale. Il *3-opt* è un algoritmo di ricerca locale che consiste nel sostituire 3 archi in soluzione con altri 3, al fine di migliorare il costo del tour.

Dopo ogni risoluzione del problema LP, si cerca di sfruttare le informazioni contenute nella soluzione LP ottimale $\bar{x}$, per trovare una soluzione migliore. Innanzitutto, si controlla se l'attuale soluzione LP sia il vettore di incidenza di un tour ammissibile. Se questo è il caso, si procede con la determinazione del costo del tour. In caso contrario, si applica un euristico *greedy*, in cui gli archi vengono ordinati in base al loro valore nell'attuale soluzione LP. Gli archi corrispondenti alle variabili non attive sono aggiunti all'elenco con il valore 0. Questa lista viene scansionata e gli archi vengono scelti per far parte della sequenza se i percorsi parziali rimangono "ammissibili". Controllare se una certa selezione è ammissibile è più complicato che per l'ATSP (TSP con costi asimmetrici), perché non è solo necessario evitare i sub-tour, ma anche assicurarsi che le precedenze non siano violate. Questo euristico ottiene buoni risultati per le strutture di precedenza densa, cioè tali che $|\tilde{A}| > \alpha\,|V|$ con $\alpha \approx 1.5$.

Viceversa, per precedenze sparse (non dense) si possono ottenere risultati migliori con il seguente approccio, che tiene conto anche dei coefficienti di costo. Con questa tecnica si modificano i coefficienti di costo originali e si esegue l'euristico con la matrice dei costi modificata $C' = (c'_{ij})_{i,j\in\{1,\dots,n\}}$ dove $c'_{ij} := (1-\bar{x}_{ij})\,c_{ij}$. Per le variabili non attive x_{lk}, si assume $\bar{x}_{lk} = 0$. L'obiettivo è creare archi la cui variabile corrispondente abbia un valore elevato, attrattivo per gli algoritmi euristici. Dopo aver costruito la matrice dei costi modificata, si esegue un euristico costruttivo con C'. Per evitare che la stessa sequenza venga generata in diverse iterazioni successive, si modifica l'euristico che viene chiamato. Il tour ammissibile costruito è il punto di partenza per le successive chiamate agli euristici di miglioramento. Poiché l'euristico di miglioramento può richiedere molto tempo, si evita di chiamarlo più di una volta con la stessa sequenza di input, utilizzando una tabella hash. Si noti che sequenze diverse possono avere lo stesso valore e pertanto si potrebbe perdere una soluzione che porti a una sequenza migliore. Nonostante ciò, questa strategia porta ad una drastica riduzione del tempo di calcolo necessario per questa fase. Inoltre, si evita di chiamare l'euristico di miglioramento se il valore c^i della sequenza di input è "troppo lontano" dal valore c^b della migliore soluzione più recente. Nell' implementazione di Ascheuer et al. [5] si esegue la parte di miglioramento solo se $\frac{c^i}{c^b} \leq 2.5$. Quando possibile, viene chiamato un euristico di ottimizzazione locale. Quando si trova una soluzione migliore, si aggiornano la migliore soluzione corrente e l'upper bound globale. Le variabili non attive, il cui arco corrispondente è in questa soluzione, vengono aggiunte al digrafo sparso e al LP.

2.2 Procedure di separazione

Il nucleo di qualsiasi algoritmo del *cutting plane* è la generazione di disuguaglianze valide violate che vengono aggiunte come piani di taglio all'attuale LP. Le classi di

disuguaglianze considerate da Ascheuer et al. [5] sono le seguenti:

- disuguaglianze "pure" per il TSP asimmetrico (D_k-inequalities, SD-inequalities),
- versione rafforzata delle disuguaglianze valide per il TSP (2-matching inequalities, T_k-inequalities e predecessor-successor inequalities),
- classe di disuguaglianze introdotte appositamente per il TSPP.

Di seguito sono presentate le disuguaglianze considerate da Ascheuer et al. [5].

***D_k*-inequalities**. Introdotte da Grötschel e Padberg [18, 19], a partire dalla seguente disuguaglianze per il ciclo $C = \{(i_1, i_2), ..., (i_k, i_1)\}$

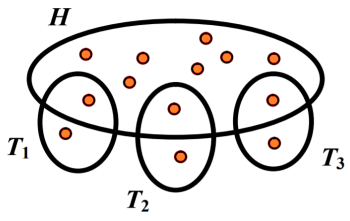
$$\sum_{(i,j)\in C} x_{ij} \leq k - 1,$$

si dividono in diverse classi di disuguaglianze, due delle quali chiamate D_k^- e D_k^+ e riportate di seguito:

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=3}^{k} x_{i_1 i_j} + \sum_{j=4}^{k}\sum_{h=3}^{j-1} x_{i_j i_h} \leq k - 1 \tag{2.2.1}$$

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=2}^{k-1} x_{i_j i_1} + \sum_{j=3}^{k-1}\sum_{h=2}^{j-1} x_{i_j i_h} \leq k - 1 \tag{2.2.2}$$

SD-inequalities. Queste disuguaglianze, introdotte da Balas e Fischetti [6], sono riprese da Ascheuer et al. [5]. Siano dati $H \subset V$, un numero dispari di *denti* disgiunti $T_1, ..., T_t$ tali che $|T_i \cap H| = 1$ e $|T_i \setminus H| = 1$, come nell'esempio in figura. Con dente di H ci si riferisce ad un sottoinsieme di A contenente gli archi aventi un estremo in H.



Siano S e D insiemi di vertici tali che $(S \cup D) \subset V \setminus (H \cup T_1 \cup ... \cup T_t)$ e tali che $|S| + |D| = t$ sia dispari, allora vale

$$\sum_{\substack{i\in S\cup H \\ j\in D\cup H}} x_{ij} + \sum_{i=1}^{t}\sum_{k,j\in T_i} x_{kj} \leq |H| + \frac{|S| + |D| + t - 1}{2} \tag{2.2.3}$$

Predecessor/successor inequalities. Balas et al. [7] hanno introdotto classi di disuguaglianze che possono essere viste come un rafforzamento delle disuguaglianze di eliminazione dei sub-tour. Siano $S \subseteq V'$ e $\bar{S} = V \setminus S$. Si definiscano gli insiemi $P(S) = \{j \in P_i;\, i \in S\}$ e $F(S) = \{j \in F_i;\, i \in S\}$. Sono verificate le seguenti disuguaglianze, dette rispettivamente *predecessor-inequality* e *successor-inequality*.

$$\sum_{\substack{i \in S \setminus P(S) \\ j \in \bar{S} \setminus P(S)}} x_{ij} \geq 1 \tag{2.2.4}$$

$$\sum_{\substack{i \in \bar{S} \setminus F(S) \\ j \in S \setminus F(S)}} x_{ij} \geq 1 \tag{2.2.5}$$

Siano $j, k \in V'$ tali che $P_j \neq \emptyset$ e $F_k \neq \emptyset$ e sia $S \subset V$ tale che $j, k \in S$. Allora le *predecessor/successor-inequality* in forma debole sono date da

$$\sum_{\substack{p \in S \setminus P_j \\ q \in \bar{S} \setminus P_j}} x_{pq} \geq 1 \tag{2.2.6}$$

$$\sum_{\substack{p \in \bar{S} \setminus F_k \\ q \in S \setminus F_k}} x_{pq} \geq 1 \tag{2.2.7}$$

Siano $X, Y \subseteq V'$ tali che $i \in P_j$ per ogni coppia $i \in X$, $j \in Y$. Sia $W = \{1\} \cup P(X) \cup F(Y)$, allora per ogni $S \subset V$ tale che $X \subset S$ e $Y \subset \bar{S}$ vale:

$$\sum_{\substack{i \in S \setminus W \\ j \in \bar{S} \setminus W}} x_{ij} \geq 1, \tag{2.2.8}$$

detta *predecessor-successor inequality*. La sua forma debole è ottenuta scegliendo $X = \{i\}$ e $Y = \{j\}$ con $(i, j) \in \tilde{A}$ e ponendo $W_{ij} = \{1\} \cup P_i \cup F_j$. In questo caso si ha

$$\sum_{\substack{p \in S \setminus W_{ij} \\ q \in \bar{S} \setminus W_{ij}}} x_{pq} \geq 1. \tag{2.2.9}$$

Precedence cycle breaking inequality (pcb-inequality). Siano $S_1, ..., S_m \subseteq V'$, con $m \geq 2$, insiemi di vertici disgiunti e tali che $P(S_i) \cap S_{i+1} \neq \emptyset$ dove $S_{m+1} = S_1$.

$$\sum_{i=1}^{m} \sum_{j,k \in S_i} x_{jk} \leq \sum_{i=1}^{m} |S_i| - m - 1. \tag{2.2.10}$$

Un caso particolare è quello della *simple pcb-inequality*, in cui $m = 2$ e $|S_2| = 1$. Può essere vista come una versione rafforzata delle disuguaglianze di eliminazione dei sub-tour ed è formulata nel modo seguente:

$$\sum_{j,k \in S_1} x_{jk} \leq |S_1| - 2. \tag{2.2.11}$$

T_k-inequalities rafforzate. Sia $W \subset V$ tale che $2 \leq |W| = k \leq n-2$. Siano $w \in W$ e $p, q \in V \setminus W$. Le T_k-inequalities introdotte da Grötschel e Padberg [18, 19]

$$\sum_{i,j \in W} x_{ij} + x_{pw} + x_{pq} + x_{wq} \leq k$$

possono essere rafforzate prendendo in considerazione i vincoli di precedenza, come mostrato da Ascheuer [2]. Definendo $\bar{W} = V \setminus W$ $\tilde{W} = W \setminus \{w\}$, si hanno

$$\sum_{i,j \in W} x_{ij} + x_{pw} + x_{pq} + x_{wq} + \sum_{\substack{i \in \bar{W} \cap F(\tilde{W}) \\ j \in W}} x_{ij} + \sum_{\substack{i \in \bar{W} \setminus F(\tilde{W}) \\ j \in W \cap F(\tilde{W})}} x_{ij} \leq k \tag{2.2.12}$$

$$\sum_{i,j \in W} x_{ij} + x_{pw} + x_{pq} + x_{wq} + \sum_{\substack{i \in W \\ j \in \bar{W} \cap P(\tilde{W})}} x_{ij} + \sum_{\substack{i \in W \cap P(\tilde{W}) \\ j \in \bar{W} \setminus P(\tilde{W})}} x_{ij} \leq k. \tag{2.2.13}$$

Inoltre, in alcuni casi, sono valide anche le seguenti disuguaglianze, con $i, k \in \tilde{W}$ e $j \notin W \cup \{p, q\}$

$$\sum_{l,m \in W} x_{lm} + x_{pw} + x_{pq} + x_{wq} \leq k - 1, \text{ se } (i,j), (j,k) \in \tilde{A} \tag{2.2.14}$$

$$\sum_{l,m \in W} x_{lm} + x_{pw} + x_{pq} + x_{wq} + \sum_{l \in \tilde{W}} x_{pl} + x_{wp} \leq k - 1 \text{ se } (i,q), (q,w) \in \tilde{A} \tag{2.2.15}$$

$$\sum_{l,m \in W} x_{lm} + x_{pw} + x_{pq} + x_{wq} + \sum_{l \in \tilde{W}} x_{lq} + x_{qw} \leq k - 1 \text{ se } (w,p), (p,k) \in \tilde{A}. \tag{2.2.16}$$

2-matching inequalities. Queste disuguaglianze sono ampiamente utilizzate nelle implementazioni di algoritmi di tipo *branch&cut* per il TSP, poiché esistono procedure di separazione euristiche precise ed efficienti. Come le T_k-inequalities, anche le 2-matching inequalities possono essere rafforzate quando vengono coinvolte le relazioni di precedenza. Siano $H, T_1, ..., T_k \subset V$ con $k \geq 1$ e tali che

$$|H \cap T_i| = 1 \qquad \forall\, i = 1, ..., k$$
$$|T_i \setminus H| = 1 \qquad \forall\, 1 \leq i < j \leq k$$
$$T_i \cap T_j = \emptyset \qquad \forall\, i = 1, ..., k$$
$$k \geq 3 \text{ e dispari oppure } k = 1 \text{ e } |H| \geq 4.$$

Siano $T = \cup_{i=1}^{k} T_i$ e $S = H \cup T$. Siano $\{s_i\} = H \cap T_i$ e $\{t_i\} = T_i \setminus H$ e si definisca

$$x_{2M} = \sum_{p,q \in H} x_{pq} + \sum_{i=1}^{k} \sum_{p,q \in T_i} x_{pq}.$$

Le 2-matching inequalities sono date da

$$x_{2M} \leq |H| + \frac{k-1}{2} \tag{2.2.17}$$

Come mostrato da Ascheuer [2], le seguenti disuguglianze sono una versione rafforzata delle 2-matching inequalities:

$$x_{2M} + \sum_{i=1}^{k} \sum_{j \in \bar{S} \cap F_{t_i}} x_{j s_i} + \sum_{i=1}^{k} \sum_{j \in \bar{S} \cap P_{t_i}} x_{s_i j} \leq |H| + \frac{k-1}{2} \tag{2.2.18}$$

$$x_{2M} + \sum_{i=1}^{k} \sum_{j \in (H \setminus T) \cap F_{s_i}} x_{j t_i} + \sum_{i=1}^{k} \sum_{j \in (H \setminus T) \cap P_{s_i}} x_{t_i j} \leq |H| + \frac{k-1}{2} \tag{2.2.19}$$

$$x_{2M} + \sum_{i=1}^{k} \sum_{\substack{j \in H \setminus s_i \\ k \in t_i \cap F(T_i)}} x_{jk} \leq |H| + \frac{k-1}{2} \tag{2.2.20}$$

$$x_{2M} + \sum_{i=1}^{k} \sum_{\substack{j=1 \\ i \neq j}}^{k} \sum_{k \in s_i \cap P(T_i)} x_{k t_j} \leq |H| + \frac{k-1}{2} \tag{2.2.21}$$

$$x_{2M} + \sum_{i=1}^{k} \sum_{\substack{j \in t_i \cap P(T_i) \\ k \in H}} x_{jk} \leq |H| + \frac{k-1}{2} \tag{2.2.22}$$

$$x_{2M} + \sum_{i=1}^{k} \sum_{\substack{j=1 \\ i \neq j}}^{k} \sum_{k \in s_i \cap F(T_i)} x_{t_j k} \leq |H| + \frac{k-1}{2} \tag{2.2.23}$$

Inoltre se esistono m triplette di vertici (u_i, v_i, w_i) con $m = \frac{k+1}{2}$ e $i = 1, ..., m$ tali che

$$\begin{aligned}
&\{u_i, w_i\} \in H \cap T, \\
&v_i \in \bar{H} \setminus T, \\
&(u_i, v_i), (v_i, w_i) \in \tilde{A} && \forall i = 1, ..., m, \\
&v_i \neq v_j && \forall i, j = 1, ..., m,\ i \neq j, \\
&w_i = u_{i+1} && \forall i = 1, ..., m-1,
\end{aligned}$$

allora vale che

$$x_{2M} \leq |H| + \frac{k-1}{2} - 1. \tag{2.2.24}$$

Durante l'esecuzione dell'algoritmo viene utilizzato un pool per memorizzare quali disuguaglianze valide sono attive e quali inattive. Una disguglianza è detta attiva se è memorizzata sia nel pool che nella matrice dei vincoli del corrente LP. Il pool è inizialmente vuoto. Successivamente, ogni taglio generato viene aggiunto alla matrice dei vincoli e al pool. Quando una disuguaglianza non è più limitante, viene eliminata dalla matrice dei vincoli, diventando inattiva. Le disuguaglianze nel pool possono essere utilizzate per rigenerare gli LP da zero o per verificare se uno qualsiasi dei tagli generati in una precedente iterazione dell'algoritmo viene violato dall'effettiva soluzione LP. Nel caso in cui il pool diventi troppo grande, si rimuovono le disuguaglianze che non sono state riattivate per un lungo periodo.

Le procedure di separazione per le disuguaglianze sopra descritte sono richiamate nell'ordine gerarchico seguente.

1. Ricerca di disuguaglianze del pool violate.
2. Separazione esatta delle disuguaglianze di eliminazione di sub-tour, come descritto da Padberg e Rinaldi [23].
3. Separazione esatta delle D_k-inequalities.
4. Separazione euristica delle SD-inequalities.
5. Separazione euristica delle predecessor inequalities.
6. Procedura di separazione euristica per la riduzione delle pcb-inequalities.
7. Separazione euristica delle predecessor-successor inequalities.
8. Separazione euristica delle successor inequalities.
9. Separazione euristica delle 2-matching inequalities rafforzate, descritta da Padberg e Rinaldi [24] e basata su un algoritmo proposto da Padberg e Rao [25].
10. Separazione euristica delle T_k-inequalities rafforzate.

Si noti che se uno dei passaggi ha esito positivo e viene quindi rilevata una disuguaglianza violata, non vengono effettuate altre procedure di separazione.

Gli step 3-4 sono eseguiti mediante l'implementazione in FORTRAN fornita da Fischetti e Toth [13] Gli step 5-8 sono eseguiti mediante la procedura descritta da Balas et al. [7]. Lo step 10 viene invece svolto come descritto di seguito.

In una prima fase, si determinano gli insiemi S tali che $|S| \geq 2$ e $\sum_{i,j \in S} x_{ij} = |S| - 1$. Ciò viene fatto numerando i nodi $i, j \in V$ tali che $x_{ij} + x_{ji} = 1$, riducendoli a un singolo

nodo e applicando iterativamente questa procedura di enumerazione al digrafo ridotto risultante. Ogni riduzione eseguita viene memorizzata in un digrafo ristretto $D_s = (V, A_s)$, in cui $(i, j) \in A_s$ se i e j sono stati ridotti a j. L'outdegree di ciascun vertice è al massimo 1, mentre più di un arco può entrare in un determinato vertice. Dato $i \in V$, si può facilmente determinare un insieme S_k contenente i e tale che $\sum_{i,j \in S_k} x_{ij} = |S_k| - 1$. Nella seconda fase, vengono enumerate triplette di nodi che potrebbero violare le disuguaglianze. Affinchè il tempo di calcolo resti moderato, Ascheuer et al. [5] considerano solo i nodi p, w e q tali che $x_{pw} + x_{pq} + x_{wq}$ sia compreso tra determinati valori, fissati a 1.33 e 1.75 in seguito a test computazionali. Inoltre, Ascheuer et al. [5] generano solo n di queste triplette. Infine, si controlla se una combinazione delle triplette (p, w, q) e S_k viola una delle T_k-inequalities rafforzate.

2.3 Metodo per imporre le variabili a 0 o 1

In diverse parti dell'algoritmo si vogliono fissare alcune variabili a 0 o 1, usando criteri di costo ridotto e criteri basati su implicazioni logiche. I primi consentono di correggere variabili attive non-base al loro valore corrente. Per essere più precisi, supponiamo di avere un limite inferiore globale glb, un limite superiore globale gub e una variabile x_{ij} non-base con un costo ridotto associato r_{ij}. Nel caso in cui $x_{ij} = 0$ e $[glb + r_{ij}] \geq gub$ allora si può fissare la variabile x_{ij} a zero. Analogamente $x_{ij} = 1$ e $[glb - r_{ij}] \geq gub$ implicano che x_{ij} può essere fissata a 1.

I test computazionali mostrano che fissare variabili, conseguentemente a implicazioni logiche, è importante per un'implementazione efficiente, perché consente di ridurre drasticamente la dimensione del problema. Quando una variabile x_{ij} è fissata a 1, è possibile correggere un certo numero di altre variabili. Queste variabili includono quelle corrispondenti all'arco inverso (j, i), a tutti gli altri archi con coda in i, ma anche variabili x_{kl} per le quali sappiamo che non possono essere presenti entrambi gli archi (i, j) e (k, l) in una soluzione ammissibile senza violare una relazione di precedenza. Si ottengono quindi le seguenti implicazioni:

$$
\begin{aligned}
\text{se } x_{ij} = 1 \;\Rightarrow\quad & x_{ji} = 0 && \\
& x_{ik} = 0 && \forall\, k \in V \setminus \{j\} \\
& x_{kj} = 0 && \forall\, k \in V \setminus \{i\} \\
& \sum_{k \in P_i} x_{jk} = 0 && \\
& \sum_{k \in F_j} x_{ki} = 0 && \\
& \sum_{\substack{k \in P_i \\ l \in F_j}} x_{kl} = 0 &&
\end{aligned}
$$

$$\sum_{\substack{k \in P_j \\ l \in F_i}} x_{kl} = 0$$

Dopo aver fissato le suddette variabili a zero, può capitare che ci siano dei vertici $i \in V$ tali che esista un solo vertice k per cui l'arco (k, i) (risp. (i, k)) sia in A . Poiché è necessario ci sia almeno un arco entrante in i (risp. uscente da i), la variabile corrispondente può essere fissata a 1. Inoltre, se x_{ij} è fissato a 1, aggiorniamo il digrafo delle precedenze $\tilde{G}$. Da un lato, sappiamo che i precede j e possiamo aggiungere questa relazione di precedenza al digrafo (se non ancora presente). D'altra parte, il vertice i (risp. j) eredita tutti i predecessori e successori del vertice j (risp. i). Successivamente, la chiusura transitiva viene aggiornata e le variabili possono essere fissate a zero. Se dopo questo aggiornamento si trova un nuovo vertice fisso, il problema si può scomporre in due sottoproblemi da gestire separatamente.

2.4 Ulteriori dettagli implementativi

In una prima fase di pre-processamento, vengono eliminati tutti gli archi che non possono essere in nessuna soluzione ammissibile, ovvero:

- gli archi (i, k) tali che (i, j) e (j, k) siano in $\tilde{A}$,
- gli archi $(1, j)$ tali che $|P_j| > 0$,
- gli archi $(i, 1)$ tali che $|F_i| > 0$.

Viene inoltre controllato se ci sono dei vertici fissi, cioè degli $i \in V$ tali che $|P_i| + |F_i| = n - 2$. In caso affermativo, è possibile spezzare il problema in due sotto-problemi e lavorare su essi separatamente.

Tutti gli archi non fissi vengono memorizzati esplicitamente nell'insieme degli archi ammissibili. Poiché solo un piccolo sottoinsieme delle variabili sarà diverso da zero in una soluzione ottimale, non viene usato l'intero insieme di variabili, ma un suo sottoinsieme "'promettente"', che si spera contenga una soluzione ottimale. Sono scelte le variabili del digrafo *k-next neighbor*, chiamato anche digrafo sparso $D_s = (V, A_s)$. Dopo un test su diversi valori possibili di k, è stato scelto $k = 4$. Si esegue un euristico iniziale per ottenere una soluzione ammissibile e gli archi di quel tour sono aggiunti al digrafo sparso. Solo le variabili corrispondenti agli archi del digrafo sparso vengono considerate nel LP iniziale. Il valore dell'upper bound globale viene inizializzato con il valore di questo tour ammissibile.

Si imposta il LP iniziale con le variabili nel digrafo sparso D_s, i vincoli di grado e i vincoli banali $x_{ij} \geq 0$.

Se si riscontra non-ammissibilità e se si osserva un fenomeno di riduzione, prima del *branching* dovrà essere eseguita una fase di determinazione del costo. Si ricorda che le

variabili corrispondenti al digrafo *k-next neighbor* sono quelle considerate nell'iniziale LP. La tecnica del digrafo sparso mantiene la dimensione del LP moderata ma rende necessario valutare le variabili non attive, che non sono considerate nell'effettivo LP. Lo scopo è quello di verificare se la soluzione LP è valida per il digrafo completo, cioè se tutte le variabili non attive vengono eliminate correttamente. Poiché questo può richiedere molto tempo, il calcolo del costo viene eseguito in modo gerarchico. Inizialmente si considerano le variabili in un cosiddetto digrafo di riserva $D_r = (V, A_r)$, prima di calcolare il costo dell'insieme completo di variabili. Il D_r è scelto come digrafo *m-next neighbor* per $m > k$. Gli archi già contenuti in A_s non sono considerati in A_r. Se il grafico delle riserve ha successo, il prezzo completo è omesso. Alcuni esperimenti computazionali hanno dimostrato che nell'attuale implementazione la scelta di m e k ha solo un'influenza minore sul tempo di computazione complessivo. Sono stati testati i valori per $k \in \{2, 3, 4, 5, 6, 7\}$ e $m \in \{5, 6, 7, 8, 9, 10\}$ ottenendo più o meno gli stessi tempi complessivi di calcolo. Si è deciso di scegliere $k = 4$ e $m = 7$ come valori predefiniti. Inoltre, la fase di determinazione del prezzo viene eseguita ogni $princing_{freq}$ iterazioni, dove il valore di default di $princing_{freq}$ è 10. Questo assicura che le variabili necessarie in una soluzione ottimale inseriscano presto il LP.

Se durante le ultime k iterazioni della fase del *cutting plane* non si ottiene un miglioramento significativo (almeno del $p\%$), viene eseguita una fase di determinazione del prezzo. Se questo non ha successo, cioè non vengono aggiunte variabili, si abbandona la fase di *cutting plane* e si passa al *branch*. Sono scelti i valori predefiniti $k = 5$ e $p = 2.5 \cdot 10^{-5}$.

Nella fase di *branch* viene scelta una variabile frazionaria e vengono generati due nuovi sottoproblemi impostando il valore della variabile a 0 (risp. a 1). La strategia di *branch* scelta è quella supportata dal framework *branch & cut*: si sceglie la variabile con il valore più vicino a 0.5; nel caso in cui esistano più variabili di questo tipo, viene scelta quella con il coefficiente di costo più alto.

Il modo in cui i sottoproblemi vengono elaborati è molto importante per un'implementazione efficiente. Nell'implementazione di Ascheuer et al. [5] si testano tre diverse strategie: depth-first-search (DFS), breadth-first-search (BRFS) e best-firstsearch (BEFS). I risultati computazionali su alcuni problemi di *benchmark* hanno dimostrato che DFS non è una buona strategia, perché il rischio di passare troppo tempo in un ramo inutile dell'albero è troppo alto. BRFS supera leggermente BEFS perché tende a dare risultati più stabili. Pertanto, Ascheuer et al. [5] usano BRFS come strategia predefinita.

2.5 Risultati computazionali

L'algoritmo di *branch&cut* descritto nel capitolo 3 è stato implementato da Ascheuer et al. [5] in C su un SUN UltraSPARC Model 140 con 128 MB di memoria e con SUN OS 5.6. Sono stati utilizzati una versione preliminare del framework *branch&cut*

ABACUS, CPLEX 5.0 per risolvere gli LP e l'implementazione in FORTRAN delle *SD* e D_k *inequalities* fornita da Fischetti e Toth [13]. Il codice è testato su diverse classi di problemi: dati reali, dati generati randomicamente e una combinazione di entrambi, tra cui problemi contenuti nella TSPLIB. Inoltre per ogni problema viene impostato un tempo di calcolo massimo pari a $5 \cdot \lceil \frac{n}{100} \rceil$ ore di tempo CPU.

In particolare, Ascheuer et al. [5] affrontano i seguenti problemi.

- Problemi con dati reali: `ESC07-ESC98` presi da IBM e `rbg019a-rbg378a` riguardanti l'itinerario di un braccio meccanico per un sistema di immagazzinamento automatico.

- Problemi generati randomicamente: `prob.*`. Tra questi, il `prob.7` è uno dei più difficili e perciò sono stati costruiti altri problemi `prob.7.k` in cui si considerano solo i primi k vertici del `prob.7`.

- Problemi ATSP puri (caso particolare di TSPP in cui $\tilde{A} = \emptyset$): problemi presi da TSPLIB (da 17 a 100 vertici) e alcuni derivati dall'itinerario di un braccio meccanico per un sistema di immagazzinamento automatico (`rbg*.0`).

- Problemi ATSP con precedenze random: alle istanze contenute nella TSPLIB sono aggiunte delle relazioni di precedenza. Inizialmente viene posto $\tilde{A} = \emptyset$ e sono definite due liste l_1 e l_2 di k interi random nell'intervallo $[1, n]$. La lista l_1 (rispettivamente l_2) corrisponde alle code (risp. teste) dei possibili archi $(i, j) \in \tilde{A}$. A partire dal primo elemento della lista, viene aggiunto ad $\tilde{A}$ l'arco $(l_1[i], l_2[i])$ se $l_1[i] \neq l_2[i]$, $(l_1[i], l_2[i]) \notin \tilde{A}$ e $\tilde{A} \cup (l_1[i], l_2[i])$ rimane aciclico. Questa procedura è seguita per $k = \frac{n}{4}$, $\frac{n}{2}$, n, $2n$. Per questi problemi, affrontati anche da Gouveia et al. [15], sono riportati i risultati in una tabella.

La maggior parte dei problemi di piccole e medie dimensioni, ovvero quelli con dati reali, sono risolti all'ottimo con un tempo di calcolo ragionevole. Tuttavia, ci sono molti altri problemi per cui il gap tra il risultato ottenuto e l'ottimo è parecchio elevato, anche dopo diverse ore di tempo computazionale. Questi problemi sono quasi tutti tra le istanze della TSPLIB con precedenze random.

Per tutte le istanze con dati reali, si trovano buoni upper e lower bound anche nella radice del *branch&cut* tree. Per ognuna di queste, eccetto che per `ESC47`, il gap tra i bound è compreso nel range del 4%. Sia l'upper che il lower bound differiscono dello stesso ordine di grandezza rispetto all'ottimo.

Un comportamento simile può essere osservato per le istanze riepilogate nella tabella sottostante. Tuttavia, il gap finale per queste istanze è maggiore. Inoltre, il divario prima del *branch* è maggiore rispetto alle altre classi di istanze. Di conseguenza, la maggior parte di questi problemi non può essere risolta con ottimalità entro il limite di tempo specificato, sebbene siano di dimensioni più piccole rispetto a molte delle istanze con dati

reali. Le diverse routine di separazione disponibili e un numero elevato di tagli non sono comunque sufficienti. Ciò è forse dovuto al fatto che le disuguaglianze generate sono solo disuguaglianze valide, ma non è noto se siano definitive, cioè non sono abbastanza forti da un punto di vista computazionale.

Per quanto riguarda il TSP, la dimensione dell'istanza, in genere denotata dal numero di vertici, non è l'unico indicatore della sua difficoltà. Si noti inoltre che solo in pochissimi casi l'euristico iniziale fornisce buone soluzioni ammissibili. Per la maggior parte delle istanze considerate, le soluzioni migliori sono trovate dall'euristico *LP-exploiting*. I risultati dell'euristico iniziale peggiorano all'aumentare del numero di precedenze coinvolte. Ciò indica che l'euristico di partenza non gestisce correttamente le relazioni di precedenza. Tutte le istanze ATSP sono risolte all'ottimo in un breve tempo di calcolo. Questi risultati indicano che le istanze con vincoli di precedenze sono più difficili delle istanze ATSP puri della stessa dimensione.

Nella tabella 1 sono presentati i risultati ottenuti da Ascheuer et al. [5] per i problemi ATSP con precedenze random, rispettando la seguente notazione.

- Problema: nome del problema.
- n: numero di vertici.
- $|\tilde{A}|$: numero di relazioni di precedenza (senza le precedenze derivate dalla transitività).
- BC-root lower bound: lower bound alla radice LP.
- BC-tree: informazioni sul *branch&cut* tree.
  - LB finale: lower bound *glb* globale trovato.
  - UB finale: upper bound *gub* globale trovato.
  - gap finale: qualità della soluzione ammissibile finale, calcolata come $\frac{gub-glb}{glb} \cdot 100$ (se il problema è risolto all'ottimo, questo campo è "-").
  - time : tempo CPU necessario per risolvere il problema (formato min:sec). Se il problema non è risolto all'ottimo in un determinato tempo, viene segnato "$*$".

In conclusione, gli esperimenti computazionali mostrano che l'algoritmo *branch&cut* di Ascheuer et al. [5] riesce a risolvere all'ottimo i problemi `p43.1` e `ry48p.1`. Il gap tra *gub* e *glb* a volte è molto alto (circa 26% per `p43.4`), nonostate la sua media sia del 6.61%.

Tabella 1: Ascheuer et al. [5]

			BC-root	**BC-tree**			
Problema	**n**	$\|\tilde{A}\|$	**LB**	**LB finale**	**UB finale**	**gap finale**	**time**
p43.1	44	9	28063	28140	28140	-	06:21
p43.2	44	20	28109	28319	28480	0.57	*
p43.3	44	37	28160	28553	28835	0.99	*
p43.4	44	50	55884	65497	83005	26.73	*
ry48p.1	49	11	14969	15805	15805	-	208:03
ry48p.2	49	23	15117	15587	16666	6.92	*
ry48p.3	49	42	17248	17813	19894	11.68	*
ry48p.4	49	58	27165	29616	31446	6.18	*
ft53.3	54	48	9115	9253	10262	10.90	*
ft53.4	54	53	13871	14124	14425	2.13	*

Capitolo 3

Metodo di Gouveia et al. [15]

3.1 Introduzione e Modello M1

L'approccio utilizzato da Gouveia et al. [15] è quello di scegliere cinque diversi modelli M1,M2,...,M5 per il TSPP ed effettuare su ognuno di essi un rilassamento lineare per ottenere un lower bound ed eventualmente l'ottimo. Il primo modello M1 non è altro che la formulazione (P2) introdotta nella sezione 1.3. Ogni modello successivo M_k ($k = 2, ..., 5$) si ottiene dal precedente M_{k-1} aggiungendo a quest'ultimo nuove valide disguguaglianze.

3.2 Modello M2

Si consideri la seguente generalizzazione delle disuguaglianze disaggregate di Desrochers e Laporte [9]

$$v_k^j + \sum_{p,q \in S} x_{pq} \leq v_k^i + |S| - 1 \quad \forall i, j, k \in V',\ \{i, j\} \subseteq S \subseteq V \setminus \{1, k\}. \tag{3.2.1}$$

Questa può essere riscritta nella cut form:

$$\sum_{\substack{p \in S' \\ q \in S}} x_{pq} \geq v_i^k + v_k^j \quad \forall\, i, j, k \in V';\ \forall (S, S') \text{ partizione di } V;\ 1, k \in S',\ i, j \in S. \tag{3.2.2}$$

Le (3.2.2) diventano particolarmente interessanti quando $v_i^k = v_k^j = 1$. Dato che i e j sono in S, mentre 1 e k non lo sono, questi vincoli stabiliscono che ogni taglio diretto che separa i vertici $\{1, k\}$ dai vertici $\{i, j\}$, contiene almeno due archi, in quanto le variabili x_{ij} sono limitate da 1. Pertanto, dal teorema di Menger per i grafi diretti [22] si conclude che esistono almeno due percorsi disgiunti per archi dall'insieme $\{1, k\}$

all'insieme $\{i, j\}$. Inoltre le variabili v_i^k e v_k^j forniscono ulteriori informazioni su questi due percorsi. Quando le due variabili sono uguali a 1, cioè il tour visita prima il vertice i, quindi il vertice k e infine il vertici j, si può concludere che il percorso corrispondente alla parte del tour che va dal vertice 1 al vertice i e il percorso corrispondente alla parte del tour che va dal vertice k al vertice j, devono essere disgiunti per archi. Questi due percorsi corrispondono ai due percorsi disgiunti definiti dalle disuguaglianze (3.2.2).

Osservazione. Se $v_i^k = 1$ per due vertici dati i e k, allora

1. esiste un cammino dal vertice 1 al vertice i che non passa dal vertice k,
2. esiste un cammino dal vertice i al vertice k che non passa dal vertice 1,
3. esiste un cammino dal vertice k al vertice 1 che non passa dal vertice i.

Questo risultato può essere tradotto in termini di disuguaglianze di taglio. Per ogni coppia di vertici $(i, k) \in V'$, le seguenti disuguaglianze sono valide per il TSPP:

$$\sum_{\substack{p \in S' \\ q \in S}} x_{pq} \geq v_i^k \quad \forall (S, S') \text{ partizione di } V \setminus \{k\},\ 1 \in S',\ i \in S, \tag{3.2.3}$$

$$\sum_{\substack{p \in S' \\ q \in S}} x_{pq} \geq v_i^k \quad \forall (S, S') \text{ partizione di } V \setminus \{1\},\ i \in S',\ k \in S, \tag{3.2.4}$$

$$\sum_{\substack{p \in S' \\ q \in S}} x_{pq} \geq v_i^k \quad \forall (S, S') \text{ partizione di } V \setminus \{i\},\ k \in S',\ 1 \in S. \tag{3.2.5}$$

Aggiungendo a M1 le disuguaglianze (3.2.2)-(3.2.5) si ottiene il modello M2.

3.3 Modello M3

Gouveia e Pesneau [14] introducono molti insiemi di disuguaglianze di taglio. In particolare, Gouveia et al. [15] considerano i tagli di *k-path* che generalizzano i tagli semplici (3.2.3)-(3.2.5) della sezione precedente. Sia $(i_1, ..., i_{k+1})$ una sequenza di $k+1 \leq n-2$ vertici distinti di V'. Si supponga che $i_0 = i_{k+2} = 1$. Sia $r \in \{0, ..., k+1\}$. Le disuguaglianze di taglio di *k-path* sono date da

$$\begin{aligned} &\sum_{\substack{p \in S' \\ q \in S}} x_{pq} \geq v_{i_1}^{i_2} + ... + v_{i_k}^{i_{k+1}} - k + 1, \\ &\forall (S', S) \text{ partizione di } V \setminus (\{i_0, ..., i_{r-1}\} \cup \{i_{r+2}, ..., i_{k+1}\}), \\ &i_r \in S',\ i_{r+1} \in S. \end{aligned} \tag{3.3.1}$$

Di seguito descriviamo il caso particolare delle disuguaglianze di tagli di *k-path* in cui $k = 2$. Sia (i, j, k) una sequenza di tre vertici distinti di V'. I quattro insiemi di disuguaglianze 2-path sono i seguenti:

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S, S') \text{ partizione di } V \setminus \{j, k\},\ 1 \in S',\ i \in S, \tag{3.3.2}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S, S') \text{ partizione di } V \setminus \{k, 1\},\ i \in S',\ j \in S, \tag{3.3.3}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S, S') \text{ partizione di } V \setminus \{1, i\},\ j \in S',\ k \in S, \tag{3.3.4}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S, S') \text{ partizione di } V \setminus \{i, j\},\ k \in S',\ 1 \in S. \tag{3.3.5}$$

Nel dettaglio, queste disuguaglianze hanno i seguenti significati.

- (3.3.2) affermano che quando i precede j e j precede k, deve esistere un percorso da 1 a i che non attraversa j e k;
- (3.3.3) affermano che quando i precede j e j precede k, deve esistere un percorso da i a j che non attraversa 1 e k;
- (3.3.4) affermano che quando i precede j e j precede k, deve esistere un percorso da j a k che non attraversa 1 e i;
- (3.3.5) affermano che quando i precede j e j precede k, deve esistere un percorso da k a 1 che non attraversa i e j.

Aggiungendo a M2 le disuguaglianze (3.3.2)-(3.3.5) si ottiene il modello M3.

3.4 Modello M4

È possibile riscrivere le disuguaglianze (3.2.3)-(3.2.5) sotto forma problemi di flusso, come descritto da Gouveia e Pesneau [14].

Si considerino le seguenti variabili decisionali

$$f_{pq}^{i,k} = \begin{cases} 1 & \text{se l'arco (p,q) è nel cammino da 1 a } i \text{ non passante per } k\ , \\ 0 & \text{altrimenti.} \end{cases}$$

Per ogni coppia di vertici $i, k \in V'$, con $i \neq k$, si ha allora il seguente sistema:

$$\begin{aligned}
&\sum_{\substack{q,\,(1,q)\in A\\ q\neq 1,k}} f_{1q}^{i,k} = v_i^k \\
&\sum_{\substack{p,\,(p,i)\in A\\ p\neq i,k}} f_{pi}^{i,k} = v_i^k \\
&\sum_{\substack{q,\,(q,p)\in A\\ q\neq i,k,p}} f_{qp}^{i,k} = \sum_{\substack{q,\,(p,q)\in A\\ q\neq 1,k,p}} f_{pq}^{i,k}, \qquad \forall p \in V \setminus \{1,i,k\} \\
&f_{pq}^{i,k} \leq x_{pq}, \qquad \forall (p,q) \in A,\ p,q \neq k \\
&f_{pq}^{i,k} \in \{0,1\}, \qquad \forall (p,q) \in A,\ p,q \neq k
\end{aligned} \tag{3.4.1}$$

Analogamente, si possono definire le variabili

$$g_{pq}^{i,k} = \begin{cases} 1 & \text{se l'arco (p,q) è nel cammino da } i \text{ a } k \text{ non passante per 1 ,} \\ 0 & \text{altrimenti.} \end{cases}$$

Per ogni coppia di vertici $i, k \in V'$, con $i \neq k$, si ha allora il seguente sistema:

$$\begin{aligned}
&\sum_{\substack{q,\,(i,q)\in A\\ q\neq 1,i}} g_{iq}^{i,k} = v_i^k \\
&\sum_{\substack{p,\,(p,k)\in A\\ p\neq 1,k}} g_{pk}^{i,k} = v_i^k \\
&\sum_{\substack{q,\,(q,p)\in A\\ q\neq 1,k,p}} g_{qp}^{i,k} = \sum_{\substack{q,\,(p,q)\in A\\ q\neq 1,i,p}} g_{pq}^{i,k}, \qquad \forall p \in V \setminus \{1,i,k\} \\
&g_{pq}^{i,k} \leq x_{pq}, \qquad \forall (p,q) \in A,\ p,q \neq 1 \\
&g_{pq}^{i,k} \in \{0,1\}, \qquad \forall (p,q) \in A,\ p,q \neq 1
\end{aligned} \tag{3.4.2}$$

Infine si considerino le variabili

$$h_{pq}^{i,k} = \begin{cases} 1 & \text{se l'arco (p,q) è nel cammino da } k \text{ a 1 non passante per } i \text{ ,} \\ 0 & \text{altrimenti.} \end{cases}$$

Per ogni coppia di vertici $i, k \in V'$, con $i \neq k$, si ha allora il seguente sistema:

$$\begin{aligned}
&\sum_{\substack{q,\,(k,q)\in A\\ q\neq i,k}} h^{i,k}_{kq} = v^k_i \\
&\sum_{\substack{p,\,(p,1)\in A\\ p\neq 1,i}} h^{i,k}_{p1} = v^k_i \\
&\sum_{\substack{q,\,(q,p)\in A\\ q\neq 1,i,p}} h^{i,k}_{qp} = \sum_{\substack{q,\,(p,q)\in A\\ q\neq i,k,p}} h^{i,k}_{pq}, \qquad && \forall p \in V \setminus \{1,i,k\} \\
&h^{i,k}_{pq} \leq x_{pq}, && \forall (p,q) \in A,\ p,q \neq i \\
&h^{i,k}_{pq} \in \{0,1\}, && \forall (p,q) \in A,\ p,q \neq i
\end{aligned} \tag{3.4.3}$$

Combinando i sistemi (3.4.2) e (3.4.3), si ottiene:

$$g^{i,k}_{pq} + h^{k,j}_{pq} \leq x_{pq} \quad \forall (p,q) \in A,\ \forall (i,k,j) \text{ sequenza di vertici distinti di } V'. \tag{3.4.4}$$

Una conseguenza sono le seguenti disuguaglianze, che possono essere viste come le inverse delle (3.2.2):

$$\sum_{\substack{q\in S'\\ p\in S}} x_{pq} \geq v^k_i + v^i_k \quad \forall i,j,k \in V',\ \forall (S,S') \text{ partizione di } V,\ i,j \in S',\ 1,k \in S \tag{3.4.5}$$

Le disuguaglianze appena riportate possono essere ottenute dalle (3.2.2) applicando adeguatamente i vincoli (1.3.2) o (1.3.3) e sono quindi ridondanti. Tuttavia, come vedremo nella sezione 3.6, la combinazione dei sistemi (3.4.2) e (3.4.3) fornirà un modo per ottenere delle disuguaglianze rafforzate.

Analogamente a quanto visto sopra, Gouveia e Pesneau [14] applicano una procedura di separazione alle disuguaglianze (3.3.2)-(3.3.5). Al fine di ottenere il Modello M4, la routine deve essere eseguita solamente sulle (3.3.3) e (3.3.4).

Si introducano le seguenti variabili

$$b^{i,j,k}_{pq} = \begin{cases} 1 & \text{se l'arco } (p,q) \in A \text{ è nel cammino da } i \text{ a } j \text{ non passante per } k \text{ e } 1\ , \\ 0 & \text{altrimenti.} \end{cases}$$

Per ogni tripletta di vertici distinti $i, j, k \in V'$ si ha il seguente sistema:

$$\begin{aligned}
&\sum_{\substack{q,\,(i,q)\in A\\ q\neq i,1,k}} b_{iq}^{i,j,k} \geq v_i^j + v_j^k - 1 \\
&\sum_{\substack{p,\,(p,j)\in A\\ p\neq j,1,k}} b_{pj}^{i,j,k} \geq v_i^j + v_j^k - 1 \\
&\sum_{\substack{q,\,(q,p)\in A\\ q\neq j,k,1,p}} b_{qp}^{i,j,k} = \sum_{\substack{q,\,(p,q)\in A\\ q\neq i,1,k,p}} b_{pq}^{i,j,k}, \qquad \forall p \in V \setminus \{1,i,k\} \\
&b_{pq}^{i,j,k} \leq x_{pq}, \qquad \forall (p,q) \in A,\ p,q \notin \{1,k\} \\
&b_{pq}^{i,j,k} \in \{0,1\}, \qquad \forall (p,q) \in A,\ p,q \notin \{1,k\}
\end{aligned} \tag{3.4.6}$$

Allo stesso modo, si definiscano

$$c_{pq}^{i,j,k} = \begin{cases} 1 & \text{se l'arco } (p,q) \in A \text{ è nel cammino da } j \text{ a } k \text{ non passante per 1 e } i\ , \\ 0 & \text{altrimenti.} \end{cases}$$

Per ogni tripletta di vertici distinti $i, j, k \in V'$ si ha il seguente sistema:

$$\begin{aligned}
&\sum_{\substack{q,\,(j,q)\in A\\ q\neq j,1,i}} c_{jq}^{i,j,k} \geq v_i^j + v_j^k - 1 \\
&\sum_{\substack{p,\,(p,k)\in A\\ p\neq k,1,i}} c_{pk}^{i,j,k} \geq v_i^j + v_j^k - 1 \\
&\sum_{\substack{q,\,(q,p)\in A\\ q\neq k,1,i,p}} c_{qp}^{i,j,k} = \sum_{\substack{q,\,(p,q)\in A\\ q\neq j,1,i,p}} c_{pq}^{i,j,k}, \qquad \forall p \in V \setminus \{1,i,k\} \\
&c_{pq}^{i,j,k} \leq x_{pq}, \qquad \forall (p,q) \in A,\ p,q \notin \{1,i\} \\
&c_{pq}^{i,j,k} \in \{0,1\}, \qquad \forall (p,q) \in A,\ p,q \notin \{1,i\}
\end{aligned} \tag{3.4.7}$$

Combinando i sistemi (3.4.2) e (3.4.7), si ottiene:

$$g_{pq}^{i,j} + c_{pq}^{j,r,s} \leq x_{pq} \quad \forall (p,q) \in A,\ \forall (i,j,r,s) \text{ sequenza di vertici distinti di } V'. \tag{3.4.8}$$

Questa disuguaglianza è valida, in quanto si possono verificare solamente i seguenti casi:

1. $g_{pq}^{i,j} = 1$, $c_{pq}^{j,r,s} = 0$ e quindi $x_{pq} = 1$,
2. $g_{pq}^{i,j} = 0$, $c_{pq}^{j,r,s} = 1$ e quindi $x_{pq} = 1$,

3. $g_{pq}^{i,j} = 0$, $c_{pq}^{j,r,s} = 0$ e $x_{pq} = 0$,

4. $g_{pq}^{i,j} = 0$, $c_{pq}^{j,r,s} = 0$ e $x_{pq} = 1$.

A partire dalla (3.4.8) e considerando i sistemi (3.4.2) e (3.4.7), si ricavano le seguenti disuguaglianze:

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S', S) \text{ partizione di } V',\ i, r \in S',\ j, s \in S. \tag{3.4.9}$$

Seguendo lo stesso ragionamento usato per ottenere la (3.4.8), è possibile scrivere le seguenti disuguaglianze a partire rispettivamente dai sistemi (3.4.3)-(3.4.6) e (3.4.1)-(3.4.7):

$$h_{pq}^{r,s} + b_{pq}^{i,j,r} \leq x_{pq} \quad \forall (p, q) \in A,\ \forall (i, j, r, s) \text{ sequenza di vertici distinti di } V', \tag{3.4.10}$$

$$f_{pq}^{i,j} + c_{pq}^{j,r,s} \leq x_{pq} \quad \forall (p, q) \in A,\ \forall (i, j, r, s) \text{ sequenza di vertici distinti di } V'. \tag{3.4.11}$$

Con un ultimo passaggio, analogo a quello utilizzato per ricavare la (3.4.9), risultano valide le seguenti disuguaglianze:

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S', S) \text{ partizione di } V \setminus \{r\},\ i, s \in S',\ j, 1 \in S, \tag{3.4.12}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S', S) \text{ partizione di } V \setminus \{j\},\ 1, r \in S',\ i, s \in S. \tag{3.4.13}$$

Il Modello M4 è quindi ottenuto aggiungendo al Modello M3 le diuguaglianze (3.4.9), (3.4.12) e (3.4.13).

3.5 Modello M5

L'ultimo modello presentato da Gouveia et al. [15] è dato dal Modello M4 con l'aggiunta di due disuguaglianze. La prima è ottenuta in modo analogo a quanto precedentemente illustrato per la (3.4.9). Infatti, da una prima combinazione dei sistemi (3.4.6) e (3.4.7) si ha

$$b_{pq}^{i,j,k} + c_{pq}^{k,r,s} \leq x_{pq} \quad \forall (p, q) \in A,\ \forall (i, j, k, r, s) \text{ sequenza di vertici distinti di } V'. \tag{3.5.1}$$

Nuovamente, combinando (3.5.1) con con i sistemi (3.4.6) e (3.4.7) si ha

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k + v_k^r + v_r^s - 2 \quad \forall (S', S) \text{ partizione di } V\backslash\{1,k\},\ i,r \in S',\ j,s \in S. \quad (3.5.2)$$

Gouveia et al. [15] presentano inoltre il seguente risultato.

Proposizione 3.5.1.

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k + v_k^r + v_r^s - 1 \quad \forall (S', S) \textit{ partizione di } V,\ 1,j,r \in S',\ i,k,s \in S. \quad (3.5.3)$$

Dimostrazione. La disuguaglianza è verificata in quanto i casi che possono verificarsi sono i seguenti.

1. Se $v_i^j = v_j^k = v_k^r = v_r^s = 1$ allora i tre cammini da 1 a i, da j a k e da r a s sono disgiunti.

2. Se solo una delle variabili v è pari a 1, la disuguaglianza diventa banale in quanto il membro di destra è nullo.

3. Se esattamente due delle variabili v sono uguali a 1, la disuguaglianza diventa

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq 1,$$

che è presentata da Gouveia et al. [15] come uno degli usuali modi di imporre che non ci siano sub-tour.

4. Rimane il caso in cui esattamente tre delle variabili v sono uguali a 1. Supponiamo che v_i^j o v_j^k (rispettivamente v_k^r o v_r^s) sia nulla, allora $v_k^r = v_r^s = 1$ (risp. $v_i^j = v_j^k = 1$). Quindi i cammini da 1 a k e da r a s (risp. da 1 a i e da j a k) sono disgiunti.

Il Modello M5 è dato quindi dal modello M4 insieme alle disuguaglianze (3.5.2) e (3.5.3).

3.6 Disuguaglianze rafforzate

Le disuguaglianze introdotte nelle sezioni precedenti e che definiscono i cinque modelli, possono essere rafforzate se si considerano le relazioni di precedenza imposte dal grafo $\tilde{G}$. Questo, come sottolineato da Gouveia e Ruthmair [17] e Letchford e Salazar-González [21], implica miglioramenti significativi per i bound LP.

Proposizione 3.6.1. *Le disuguaglianze (3.2.3)-(3.2.5) possono essere rafforzate nel modo seguente:*

$$\sum_{\substack{q\in S'\\ p\in S}} x_{pq} \geq v_i^k \quad \forall (S,S') \text{ partizione di } V\setminus(\{k\}\cup N_{\tilde{A}}^+(i)\cup N_{\tilde{A}}^+(k)), \quad 1\in S',\ i\in S \tag{3.6.1}$$

$$\sum_{\substack{p\in S'\\ q\in S}} \geq v_i^k \quad \forall (S,S') \text{ partizione di } V\setminus(\{1\}\cup N_{\tilde{A}}^-(i)\cup N_{\tilde{A}}^+(k)), \quad i\in S',\ k\in S \tag{3.6.2}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^k \quad \forall (S,S') \text{ partizione di } V\setminus(\{i\}\cup N_{\tilde{A}}^-(i)\cup N_{\tilde{A}}^-(k)), \quad k\in S',\ 1\in S \tag{3.6.3}$$

dove $N_{\tilde{A}}^+(i)=\{q\in V',\ (i,q)\in\tilde{A}\}$ *e* $N_{\tilde{A}}^-(i)=\{p\in V',\ (p,i)\in\tilde{A}\}$.

Dimostrazione. Gouveia et al. [15] presentano la dimostrazione solo per (3.6.2), in quanto per le altre due disuguaglianze il procedimento è analogo.

Siano i e j due vertici distinti di V'. Se $v_i^j=1$ allora esiste un percorso da i a j che non passa attraverso il vertice 1. Si supponga esista un vertice q tale che $(j,q)\in\tilde{A}$, cioè che j debba precedere q nel tour. Di conseguenza, q non può appartenere al percorso da i a j e può essere rimosso dagli insiemi che definiscono il taglio (3.2.4). Allo stesso modo, ogni vertice p tale che $(p,i)\in\tilde{A}$ può essere rimosso dal taglio poiché p deve precedere i nel tour. □

Un ragionamento simile può essere usato per dimostrare il seguente risultato.

Proposizione 3.6.2. *Le disuguaglianze (3.3.2)-(3.3.5) possono essere rafforzate nel modo seguente:*

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S,S') \text{ part. di } V\setminus(\{j,k\}\cup N_{\tilde{A}}^+(i)\cup N_{\tilde{A}}^+(j)\cup N_{\tilde{A}}^+(k)), \quad 1\in S',\ i\in S \tag{3.6.4}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S,S') \text{ part. di } V\setminus(\{1,k\}\cup N_{\tilde{A}}^-(i)\cup N_{\tilde{A}}^+(j)\cup N_{\tilde{A}}^+(k)), \quad i\in S',\ j\in S \tag{3.6.5}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S,S') \textit{ part. di } V \setminus (\{1,i\} \cup N_{\tilde{A}}^-(i) \cup N_{\tilde{A}}^-(j) \cup N_{\tilde{A}}^+(k)), \quad j \in S',\, k \in S \tag{3.6.6}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k - 1 \quad \forall (S,S') \textit{ part. di } V \setminus (\{i,j\} \cup N_{\tilde{A}}^-(i) \cup N_{\tilde{A}}^-(j) \cup N_{\tilde{A}}^-(k)), \quad k \in S',\, 1 \in S \tag{3.6.7}$$

con (i,j,k) una sequenza di vertici distinti.

Per quanto riguarda le altre disuglianze, il procedimento seguito è meno intuitivo del precedente, ma facilmente descrivibile. Si consideri ad esempio (3.2.2), che si ricava combinando i sistemi (3.4.1) per le precedenze (i,k) e (3.4.2) per le precedenze (k,j). Questi due modelli di flusso corrispondono all'ammissibilità per il problema associato alla separazione dei tagli (3.2.3) per (i,k) e dei tagli (3.2.4) per (k,j).

Invece di considerare i sistemi di flusso associati alle disuguaglianze di taglio originali (3.2.3) e (3.2.4), si scelgono ora i sistemi di flusso associati alle versioni rafforzate appena presentate (3.6.1) e (3.6.2). Il sistema di flusso corrispondente alle disuguaglianze (3.6.1) per la precedenza (i,k) è definito sul grafo in cui vengono rimossi i vertici in $\{k\} \cup N_{\tilde{A}}^+(i) \cup N_{\tilde{A}}^+(k)$. Allo stesso modo il sistema di flusso corrispondente alle disuguaglianze (3.6.2) per la precedenza (k,j) è definito sul grafo in cui vengono rimossi i vertici in $\{1\} \cup N_{\tilde{A}}^-(i) \cup N_{\tilde{A}}^+(k)$. Si supponga ora che $v_i^k = v_k^j = 1$. Ciò significa che i percorsi da 1 a i e da k a j devono essere disgiunti. Un vertice che è stato rimosso in entrambi i precedenti sistemi di flusso non può appartenere a nessuno di questi due percorsi e, quindi, qualsiasi arco incidente a tale vertice può essere rimosso dalla disuguaglianza. Di conseguenza, il taglio (S',S) che definisce le disuguaglianze (3.2.2) può essere definito sul grafo in cui i vertici di $(\{k\} \cup N_{\tilde{A}}^+(i) \cup N_{\tilde{A}}^+(k)) \cap (\{1\} \cup N_{\tilde{A}}^-(i) \cup N_{\tilde{A}}^+(k))$ vengono rimossi. Questo significa che le disuguaglianze rafforzate possono essere ottenute direttamente dalla combinazione dei due flussi associati ai tagli rafforzati (3.6.2) e (3.6.1). Per semplificare, sia $N_{\tilde{A}}^f(i,k)$ l'insieme di vertici da rimuovere dal cut-set in disuguaglianze (3.6.1), cioè $N_{\tilde{A}}^f(i,k) = \{k\} \cup N_{\tilde{A}}^+(i) \cup N_{\tilde{A}}^+(k)$. Allo stesso modo, si definiscano $N_{\tilde{A}}^g(i,k)$ e $N_{\tilde{A}}^h(i,k)$ per disuguaglianze (3.6.2) e (3.6.3), rispettivamente.

Proposizione 3.6.3. *Per ogni sequenza (i,j,k) di vertici distinti in V', si ha*

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^k + v_k^j \quad \forall (S,S') \textit{ part. di } V \setminus (N_{\tilde{A}}^f(i,k) \cup N_{\tilde{A}}^g(k,j)), \quad 1, k \in S',\, i,j \in S \tag{3.6.8}$$

Nella sezione 3.4 abbiamo osservato che le disuguaglianze (3.4.5) possono essere ottenute combinando i sistemi (3.4.2) e (3.4.3) e possono essere mostrati come equivalenti, sotto i vincoli di assegnazione, alle disuguaglianze derivate dalla combinazione di f con g. Tuttavia, per il TSPP, la regola data sopra con i flussi g e h porta a disuguaglianze che, in generale, non equivalgono alle disuguaglianze derivate dalla combinazione di f con g, e quindi Gouveia et al. [15] considerano i due insiemi nei risultati computazionali.

Questo procedimento in cui si intersecano i sottoinsiemi di vertici associati a ciascun flusso utilizzato per derivare una nuova disuguaglianza, è valido per ottenere versioni più forti delle disuguaglianze (3.4.9), (3.4.12), (3.4.13) e (3.5.2). Siano $N^a_{\tilde{A}}(i,j,k)$, $N^b_{\tilde{A}}(i,j,k)$, $N^c_{\tilde{A}}(i,j,k)$ e $N^d_{\tilde{A}}(i,j,k)$ gli insiemi di vertici da rimuovere dal cut-set nelle disuguaglianze (3.3.2)-(3.3.5) rispettivamente.

Proposizione 3.6.4. *Per ogni sequenza (i,j,r,s) di vertici distinti in V', si ha*

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S,S') \text{ part. di } V\setminus (N^g_{\tilde{A}}(i,j)\cup N^c_{\tilde{A}}(j,r,s)), \quad i,r\in S',\ j,s\in S \tag{3.6.9}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S,S') \text{ part. di } V\setminus (N^h_{\tilde{A}}(r,s)\cup N^b_{\tilde{A}}(i,j,r)), \quad i,s\in S',\ j,1\in S \tag{3.6.10}$$

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^r + v_r^s - 1 \quad \forall (S,S') \text{ part. di } V\setminus (N^f_{\tilde{A}}(i,j)\cup N^c_{\tilde{A}}(j,r,s)), \quad 1,r\in S',\ i,s\in S \tag{3.6.11}$$

Per ogni sequenza (i,j,k,r,s) di vertici distinti in V', si ha

$$\sum_{\substack{p\in S'\\ q\in S}} x_{pq} \geq v_i^j + v_j^k + v_k^r + v_r^s - 2 \quad \forall (S,S') \text{ part. di } V\setminus (N^b_{\tilde{A}}(i,j,k)\cup N^c_{\tilde{A}}(k,r,s)), \quad i,r\in S',\ j,s\in S \tag{3.6.12}$$

Per quanto riguarda le disuguaglianze (3.5.3) non è chiaro come trovare un insieme corrispondente di vertici da eliminare.

3.7 Risultati computazionali

Lo studio computazionale di Gouveia et al. [15] è eseguito su un Intel (R) Core (TM) i7-4790 a 3.60 GHz, con 8 GB RAM e utilizzando la tecnologia Concerto di CPLEX 12.6.1 per C++. In questo studio sono utilizzate alcune istanze prese dalla libreria TSPLIB: `p43.1-4` e `ry48p.1-4`. Sono stati inoltre generati in modo randomico quattro insiemi aggiuntivi di istanze, con 25 o 35 vertici, e costi simmetrici random o euclidei, basati sul metodo utilizzato per generare le istanze di Letchford e Salazar-González [21].

In questa sezione saranno presentati i risultati computazionali di Gouveia et al. [15] solamente per i problemi `p43.1-4` e `ry48p.1-4`, in quanto affrontati anche da Ascheuer et al. [5]. Nella tabella 2 viene usata la seguente notazione.

- Problema: nome del problema;
- n: numero di vertici;
- $|\tilde{A}|$: numero di vincoli di precedenza;
- OPT: ottimo oppure gli upper e lower bound migliori conosciuti (forniti da Gouveia e Ruthmair [17]);
- M1: modello base dato da (1.3.2), (1.3.3), (1.3.8) + (1.3.4), (1.3.9), (1.3.6) + (1.3.5);
- M2: M1 + (3.2.2) + (3.2.3)-(3.2.5);
- M3: M2 + (3.3.2)-(3.3.5);
- M4: M3 + (3.4.9) + (3.4.12) + (3.4.13);
- M5: M4 + (3.5.2) + (3.5.3), per il quale Gouveia et al. [15] non presentano risultati in quanto la separazione delle disuguaglianze coinvolte ha richiesto troppo tempo;
- LPB: bound ottenuti con il rilassamento lineare del modello corrispondente;
- t: tempo (in secondi) necessario per risolvere il rilassamento lineare del modello corrispondente.

I risultati presentati in tabella 2 sono ottenuti senza utilizzare le disuguaglianze rafforzate descritte nella sezione 3.6.

Tabella 2: Gouveia et al. [15] senza rimozione dei vertici

				M1		M2		M3		M4	
Problema	**n**	$\|\tilde{A}\|$	**OPT**	**t**	**LPB**	**t**	**LPB**	**t**	**LPB**	**t**	**LPB**
p43.1	44	9	28140	14	920	4673	28138.3	10030	28138.7	11889	28140
p43.2	44	20	28480	12	1064	6603	28397	18657	28401	86001	28425.7
p43.3	44	37	28835	23	1449.11	3127	28643.2	6173	28653	45384	28707
p43.4	44	50	83005	1	56000.8	71	82874.6	172	82876.9	7888	82919.8
ry48p.1	49	11	15805	51	13889.6	3213	15331.6	5417	15344.7	60460	15420.2
ry48p.2	49	23	[16074,16666]	68	14060.6	2263	15711.9	3936	15725.5	61439	15864.9
ry48p.3	49	42	[19490,19894]	98	15907.6	773	17718.2	1836	17742.5	28658	18256.2
ry48p.4	49	58	31446	6	25124.8	171	27397.6	401	27499.9	10982	28377.2

La tabella 2 mostra che nelle istanze `p43` il modello M4 migliora sostanzialmente i bound del rilassamento LP rispetto ai modelli precedenti. Tuttavia, questo richiede tempi notevolmente più lunghi. Si osservi che per l'istanza `p43.1`, il rilassamento LP associato al modello M4 coincide con il valore ottimale della soluzione intera. Per quanto riguarda le istanze `ry48p`, le conclusioni sono abbastanza simili. Ancora una volta il modello M4 è in grado di migliorare considerevolmente i bound del rilassamento LP, sebbene siano ancora lontani dai corrispondenti valori interi ottimali.

Gouveia et al. [15] affrontano anche uno studio simile in cui implementa l'eliminazione dei vertici come descritto nella sezione 3.6. Questi risultati sono descritti nella tabella 3, in cui le voci corrispondono agli stessi modelli descritti in precedenza. Tuttavia, si osserva che il modello M2 include anche le disuguaglianze (3.4.5) poiché esse ora differiscono dalle normali disuguaglianze (3.6.8) in termini di vertici da rimuovere, come spiegato nella sezione 3.6.

Tabella 3: Gouveia et al. [15] con rimozione dei vertici

				M1		M2		M3		M4	
Problema	**n**	$\|\tilde{A}\|$	**OPT**	**t**	**LPB**	**t**	**LPB**	**t**	**LPB**	**t**	**LPB**
p43.1	44	9	28140	14	920	5379	28138.3	12215	28138.8	13959	28140
p43.2	44	20	28480	12	1064	10848	28429.5	29686	28432	125780	28443.4
p43.3	44	37	28835	23	1449.11	5708	28733.4	9926	28740.7	36861	28785.7
p43.4	44	50	83005	1	56000.8	270	82991.7	585	82993.6	1642	83005
ry48p.1	49	11	15805	42	13889.6	3155	15386.1	5507	15403.8	103330	15477.4
ry48p.2	49	23	[16074,16666]	83	14060.6	2502	15806	4275	15844.7	93498	15938.1
ry48p.3	49	42	[19490,19894]	84	15907.6	1463	18352.1	4373	18506.7	39339	18704.9
ry48p.4	49	58	31446	3	25124.8	453	30465	912	30599.7	15651	30919.8

Nella tabella 3, la notazione è la stessa usata per la tabella 2, ma questa volta i modelli sono dati dalle disuguaglianze rafforzate come segue:

- M1: modello base come prima;
- M2: M1 + (3.4.5) + (3.6.8) + (3.6.2)-(3.6.3);
- M3: M2 + (3.6.4)-(3.6.7);

- M4: M3 + (3.6.9) + (3.6.10) + (3.6.11);
- M5: M4 + (3.6.12) + (3.5.3), per il quale Gouveia et al. [15] non presentano risultati in quanto la separazione delle disuguaglianze coinvolte ha richiesto troppo tempo;

Entrambe le tabelle mostrano che i migliori lower bound si ottengono con il modello M4. Il confronto con i risultati ottenuti da Ascheuer et al. [5] mostra che i lower bound ottenuti mediante M4 sono migliori di quelli ottenuti da Ascheuer et al. [5]. Tuttavia, Ascheuer et al. [5] risolvono il problema `ry48p.1` in 208 minuti, mentre con M4 si impiegano 103330 secondi (circa 28 ore) per il solo calcolo di un lower bound.

Capitolo 4

Metodi di programmazione dinamica

4.1 Ricursione di programmazione dinamica per il TSPP

Il TSPP può essere risolto estendendo a questo problema il metodo della Programmazione Dinamica (DP) introdotto da Held e Karp [20] per risolvere il TSP.

Sia $f(S,j)$ il costo minimo di un cammino che parte dal vertice $1 \in S$, visita una ed una sola volta tutti i vertici in $S \subseteq V$, rispettando i vincoli di precedenza, e termina nel vertice $j \in S$. Le funzioni $f(S,j)$ possono essere calcolate mediante la seguente ricursione di DP:

$$\begin{aligned} f(S,j) &= \min_{i \in S \setminus \{j\}} \{f(S \setminus \{j\}, i) + c_{ij}\} \\ \forall j \in V'_{|S|}, \;\; &\forall S \subseteq V' \setminus F_j \text{ tale che } P_j \subseteq S. \end{aligned} \tag{4.1.1}$$

dove i sottoinsiemi V'_k, $k = 1, ..., n$, sono definiti nella sezione 1.4.

Le funzioni $f(S,j)$ sono inizializzate come

$$\begin{aligned} f(\{1\}, 1) &= 0, \\ f(\{1,j\}, j) &= c_{1j}, \; \forall j \in V'_2. \end{aligned} \tag{4.1.2}$$

Il costo della soluzione ottima del TSPP è dato da

$$z(P) = \min_{i \in V'_n} \{f(V', i) + c_{i1}\}. \tag{4.1.3}$$

Tuttavia, risolvere il TSPP mediante la ricursione (4.1.1) può risultare proibitivo poiché il numero delle variabili di stato (S,j) è dell'ordine di $\mathcal{O}(2^n\, n)$. Ciò significa che per $n = 20$ si ha $\mathcal{O}(2^n\, n) \approx 21,000,000$ e che per n=30 si ha $\mathcal{O}(2^n\, n) \approx 21,000,000.000$.

4.2 Lower bound LB basato sulla state-space relaxation

Quando il numero delle variabili di stato rende computazionalmente proibitivo risolvere all'ottimo un problema, in questo caso il TSPP, mediante la DP si può ricorrere al metodo della *state space relaxation* come descritto da Christofides et al. [8]. Questo consente di rilassare, ovvero ridurre, lo spazio delle variabili di stato di una ricursione di DP, in modo che la soluzione della ricursione nello spazio rilassato produca un valido lower bound all'ottimo del problema originale. Tale lower bound può poi essere impiegato in un metodo *branch & bound* per risolvere all'ottimo il problema oppure nella ricursione esatta di DP, come la (4.1.1), per eliminare gli stati che non possono condurre alla soluzione ottima.

Sia $g(\cdot)$ una funzione di mapping dal dominio di (S, j) a qualche altro spazio $(g(S), j)$ di dimensioni ridotte, nel quale la ricursione (4.1.1) diviene:

$$\begin{aligned} &f(g(S), j) = \min_{i \in (S \setminus \{j\})} \{f(g(S \setminus \{j\}), i) + c_{ij}\}, \\ &\forall j \in V'_{|S|},\ \forall g(S) \subseteq g(V' \setminus F_j) \ \text{ tale che } g(P_j) \subseteq g(S \setminus \{j\}). \end{aligned} \tag{4.2.1}$$

L'inizializzazione (4.1.2) può essere riscritta come

$$\begin{aligned} f(g(\{1\}), 1) &= 0 \\ f(g(\{1, j\}), j) &= c_{1j},\ \forall j \in V'_2. \end{aligned} \tag{4.2.2}$$

Nell'operazione di minimo della (4.2.1) si può sostituire $i \in (S \setminus \{j\})$ con $i \in V'_{|S|-1}$, in quanto è ovvio che $(S \setminus \{j\}) \subseteq V'_{|S|-1}$. Ciò è valido poiché siamo interessati ad un lower bound e non a risolvere all'ottimo il problema.

Si consideri il mapping $g(S) = |S| = k$. Per un dato k intero e un dato vertice j, il sottoinsieme di stati $\{(S, j); S \subseteq V', |S| = k\}$ corrisponde nello spazio ridotto all'unico stato $\{(k, j)\}$. La ricursione (4.2.1) diviene quindi

$$f(k, j) = \min_{i \in V'_{k-1}} \{f(k-1, i) + c_{ij}\},\ k = 2, ..., n,\ j \in V'_k, \tag{4.2.3}$$

con l'inizializzazione

$$\begin{aligned} f(1, 1) &= 0 \\ f(2, j) &= c_{1j},\ \forall j \in V'_2. \end{aligned} \tag{4.2.4}$$

Il costo della soluzione ottima del TSPP è pertanto dato da

$$LB = \min_{i \in V'_n} \{f(n, i) + c_{i1}\}. \tag{4.2.5}$$

Si indichi con $tour = (i_1 = 1, i_2, ..., i_n, i_{n+1} = 1)$ il cammino di costo LB, il quale sarà calcolato mediante l'algoritmo `path_back`, descritto in seguito. Sia (RP3) il problema ottenuto da (P3) ignorando i vincoli (1.4.5), (1.4.6) e (1.4.8). È evidente che l'ottimo $z(RP3)$ sia un valido lower bound alla soluzione ottima $z(P3)$ del TSPP. È facile osservare inoltre che il tour corrispondente al costo LB è anche la soluzione ottima di (RP3): infatti è sufficiente porre $x_{i_{k-1}i_k k} = 1$, $k = 2, ..., n+1$, e $x_{ijk} = 0$ altrimenti. Ne segue quindi che $LB = z(RP3)$.

La funzione $f(k, j)$ rappresenta il costo del cammino di costo minimo di cardinalità k, non necessariamente elementare, dal vertice 1 al vertice j. Per questo motivo il cammino corrispondente a $f(k, j)$ è detto *k-path* e la funzione di mapping $g(S) = |S|$ è denominata *k-path relaxation*. È facile osservare che nella ricursione rilassata i vincoli di precedenza vengono a loro volta rilassati e parzialmente considerati solo nei sottoinsiemi V'_k. Ne segue che il *k-path* di costo $f(k, j)$ non solo può visitare un vertice più di una volta ma può violare i vincoli di precedenza. Inoltre possono formarsi loop di 2 vertici consecutivi del tipo (i, j, i). Quest'ultimo inconveniente può essere eliminato facilmente senza accrescere la complessità computazionale della risoluzione come segue.

Sia $\pi(k, j)$ il vertice che precede j nel *k-path* di costo $f(k, j)$ e sia $h(k, j)$ il costo del *k-path* di costo minimo tale che il vertice $\gamma(k, j)$ che precede j sia diverso da $\pi(k, j)$. La ricursione di DP per calcolare le funzione $f(k, j)$ e $h(k, j)$ è la seguente. Per una data cardinalità k ed un dato vertice j si definisca $\varphi(i, j)$ come il costo del *k-path* di costo minimo dove i è il vertice che precede j. Si avrà

$$\varphi(i,j) = \begin{cases} f(k-1,i) + c_{ij}, \text{ se } \pi(k-1,i) \neq j, \\ h(k-1,i) + c_{ij}, \text{ se } \pi(k-1,i) = j. \end{cases} \tag{4.2.6}$$

Pertanto si ha

$$f(k,j) = \min_{i \in V'_{k-1} \setminus F_j} \{\varphi(i,j)\} \tag{4.2.7}$$

$$\pi(k,j) = i^* \tag{4.2.8}$$

dove i^* è il vertice che produce il minimo nella (4.2.7),

$$h(k,j) = \min_{\substack{i \in V'_{k-1} \setminus F_j \\ i \neq i^*}} \{\varphi(i,j)\} \tag{4.2.9}$$

$$\gamma(k,j) = i' \tag{4.2.10}$$

dove i' è il vertice che produce il minimo nella (4.2.9).

4.3 Algoritmo per il calcolo di LB

In questa sezione sono descritti gli algoritmi `dp_forw` e `path_back` per il calcolo, rispettivamente, delle funzioni $f(k,j)$ e $h(k,j)$ e del lower bound LB dato dalla (4.2.5) e del cammino *tour* di costo LB.

Algoritmo `dp_forw`: calcola le funzioni $f(k,j)$ e $h(k,j)$ e il lower bound LB.

Algorithm 1 `dp_forw`

1) Inizializzazione

$f(1,1) = 0$
$\pi(1,1) = 0$
$h(1,1) = \infty$
$\gamma(1,1) = 0$
for $j = 2, ..., n$ **do**
  $h(2,j) = \infty$
  $\gamma(2,j) = 0$
  if $np_j = 0$ **then**
    $f(2,j) = c_{1j}$
    $\pi(2,j) = 1$
  else
    $f(2,j) = \infty$
    $\pi(2,j) = 0$
  end if
end for
for $k = 3, ..., n$ **do**
  for $j = 2, ..., n$ **do**
    $f(k,j) = h(k,j) = \infty$
    $\pi(k,j) = \gamma(k,j) = 0$
  end for
end for

2) Ricursione

for $k = 3, ..., n$ **do**
  for $j \in V'_k$ **do**
    for $i \in V'_{k-1} \setminus F_j$ **do**
      if $\pi(k-1,i) \neq j$ **then**
        $rr = f(k-1,i) + c_{ij}$

else
$rr = h(k-1,i) + c_{ij}$
end if
if $rr < f(k,j)$ **then**
$h(k,j) = f(k,j)$
$\gamma(k,j) = \pi(k,j)$
$f(k,j) = rr$
$\pi(k,j) = i$
else
if $rr < h(k,j)$ **then**
$h(k,j) = rr$
$\gamma(k,j) = i$
end if
end if
end for
end for
end for

3) Calcolo Lower Bound LB

$LB = \infty$
$i_{best} = 0$
for $i = 2, ..., n$ **do**
if $nf_i = 0$ **then**
$rr = f(n,i) + c_{i1}$
if $rr < LB$ **then**
$LB = rr$
$i_{best} = i$
end if
end if
end for
return

Algoritmo `path_back`. Questo algoritmo viene utilizzato dopo l'esecuzione di `dp_forw` per ricostruire il cammino *tour* di costo LB, dove i_{best} è l'ultimo vertice visitato, trovato da `dp_forw`.

Algorithm 2 `path_back`

1) Inizializzazione

$tour(n+1) = 1$
$tour(1) = 1$
$i = i_{best}$
$last = 1$

2) Backtracking

for $k = n, n-1, ..., 2$ **do**
  $tour(k) = i$
  $j = \pi(k, i)$
  if $j = last$ **then**
    $j = \gamma(k, i)$
  end if
  $last = i$
  $i = j$
end for
return

4.4 Funzioni inverse

Si indichi con $f^{-1}(k, j)$ il costo del cammino di costo minimo, di cardinalità k e non necessariamente elementare per andare dal vertice j al vertice 1. Sia $\pi^{-1}(k, j)$ il vertice successore di j in tale cammino. Inoltre si definisca come $h^{-1}(k, j)$ il costo del cammino, di costo minimo di cardinalità k, non necessariamente elementare, che parte dal vertice j e termina nel vertice 1 e tale che il successore $\gamma^{-1}(k, j)$ di j in tale cammino sia diverso da $\pi^{-1}(k, j)$. Le funzioni $f^{-1}(k, j)$, $h^{-1}(k, j)$, $\pi^{-1}(k, j)$ e $\gamma^{-1}(k, j)$ possono essere calcolate con gli algoritm `dp_forw` e `path_back`, descritti in precedenza, con i seguenti accorgimenti:

- è necessario sostituire la matrice dei costi con la sua trasposta;
- le precedenze devono essere invertite, ovvero la precedenza i *precede* j diviene j *precede* i.

In seguito `dp_back` indicherà l'algoritmo per il calcolo delle funzioni $f^{-1}(k, j)$ e $h^{-1}(k, j)$.

4.5 Metodo del subgradiente per migliorare il lower bound LB

Il valore del lower bound LB può essere migliorato, ovvero massimizzato, poiché, come osservato in precedenza, il tour di costo LB può risultare non-elementare. In

questo caso è possibile assegnare una penalità $u_i \in \mathbb{R}$ ad ogni vertice che non è stato visitato esattamente una volta dal tour di costo LB e ricalcolare, mediante l'algoritmo `dp_forw`, le funzioni $f(k,j)$ e $h(k,j)$ usando i costi modificati $c'_{ij} = c_{ij} - \frac{1}{2}u_i - \frac{1}{2}u_j$. Questo metodo di penalizzazione equivale al problema rilassato (RP3') che deriva da (P3) rilassando in modo lagrangiano i vincoli (1.4.5) e (1.4.6) ed ignorando i vincoli di precedenza (1.4.8). Sia z il costo del tour ottimo calcolato mediante l'espressione (4.2.5). Si può dimostrare che $LB' = z + \sum_{i \in V'} u_i$ è un valido lower bound qualunque siano i valori assegnati alle penalità u_i, $i \in V'$. Un metodo efficiente per calcolare le penalità u_i, $i \in V'$, è fornito dal metodo del subgradiente, descritto in Appendice. Si tratta di una procedura iterativa in cui le penalità u_i vengono modificate ad ogni iterazione in base al tour corrispondente al lower bound LB' ottenuto nella corrente iterazione. Sia d_i il grado del vertice i nel tour di costo LB', ovvero il numero di volte in cui il vertice i viene visitato. Si avrà quindi $d_i = 0$ se il vertice i non viene visitato e $d_i > 1$ se il vertice i è visitato più di una volta. Sia z_{UB} un upper bound al costo ottimo del TSPP. Le penalità u_i vengono modificate nel modo seguente

$$u_i = u_i - step(d_i - 1),\ i \in V', \tag{4.5.1}$$

dove *step* viene calcolato come

$$step = \alpha \frac{z_{UB} - LB}{\sqrt{\sum_{i \in V'} (d_i - 1)^2}}. \tag{4.5.2}$$

Lo scalare α rappresenta la step size e viene modificato iterativamente in base al valore di LB' come descritto nel seguente algoritmo.

Siano max_{iter} il numero massimo di iterazioni ammesse e $ndown_{max}$ un parametro per il controllo del valore di α, il quale viene inizializzato come $\alpha = 2.0$. Se per $ndown_{max}$ iterazioni consecutive il lower bound non cresce rispetto al miglior bound ottenuto, allora il valore di α viene decrementato con $\alpha = 0.75\alpha$.

Algorithm 3 Algoritmo del subgradiente

1) Inizializzazione

$u_i = 0\ \forall i \in V'$
$c'_{ij} = c_{ij}\ \forall (i,j) \in A$
$\alpha = 2.0$
$LB = \infty$
$ndown = 0$
$iter = 1$

2) Calcolo di LB'

Calcola le funzioni $f(k,j)$ ed $h(k,j)$ e LB usando i costi c'_{ij} mediante l'algoritmo `dp_forw`.
Siano $LB' = \min_{i \in V'_n}\{f(n,i) + c_{i1}\} + \sum_{i \in V'} u_i$ e i_{best} l'indice del vertice che produce LB'.
Sia tour() il cammino di costo LB' prodotto dall'algoritmo `path_back`.
if $LB' > LB$ **then**
  $LB = LB'$
  $u^* = u$
end if
if $iter = max_{iter}$ **then**
  stop
else
  $iter = iter + 1$
end if

3) Update delle penalità

if $LB' \leq LB$ **then**
  $ndown = ndown + 1$
  if $ndown = ndown_{max}$ **then**
    $\alpha = 0.75\,\alpha$
    $ndown = 0$
  end if
end if
Calcola lo $step = \alpha \frac{z_{UB} - LB}{\sqrt{\sum_{i \in V'}(d_i - 1)^2}}$ e aggiorna le penalità $u_i = u_i - step(d_i - 1),\ i \in V'$.
Ritorna allo step 2.

4.6 Lower bound $LB2$ basato sul rilassamento kL-path

In questa tesi viene proposto un nuovo rilassamento dello spazio degli stati che tiene conto dei vincoli di precedenza in modo parziale ma più efficiente di come tali vincoli vengono considerati dalla *k-path relaxation*. Il lower bound $LB2$ che si ottiene da questo rilassamento domina il lower bound LB, descritto in precedenza, ottenuto mediante la *k-path relaxation* (i.e. $LB2 \geq LB$). Il nuovo rilassamento si basa sulle seguenti osservazioni.

Sia $L = (i_0 = 1, i_1, ..., i_r, i_{r+1}...., i_h, i_{h+1} = n+1)$ la sequenza dei vertici del cammino con costo maggiore dal vertice 1 al vertice $n+1$ nel grafo delle precedenze $\tilde{G}$.

Esempio. Riprendiamo il problema definito nell'esempio della sezione 1.1. In questo caso il longest path L nel grafo delle precedenze è dato da $(1, 4, 7, 8, 10, 1)$ con un costo pari a 154. In figura L è evidenziato dalle frecce non tratteggiate.

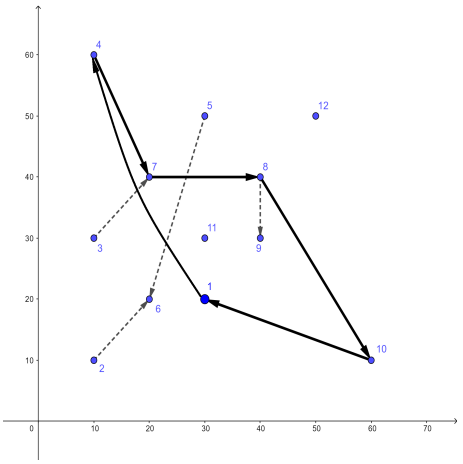


Fig. 2: Longest path L

È ovvio che in ogni soluzione ammissibile del TSPP il vertice i_r debba precedere il vertice i_{r+1} di L, per ogni $r = 1, ..., h$. Sia $\bar{V}_r$ il sottoinsieme di vertici che devono o possono essere visitati fra i vertici i_{r-1} e i_r , $r = 2, ..., h+1$. È evidente che nel cammino da i_{r-1} a i_r non possa essere visitato alcun vertice $j \in P_{i_{r-1}} \cup F_{i_r}$, quindi si ha:

$$\bar{V}_r = \{j \in V' \setminus \{i_{r-1}, i_r\};\ j \notin P_{i_{r-1}} \cup F_{i_r}\} \cup \{i_r\} \quad \forall\, r = 2, ..., h+1. \tag{4.6.1}$$

Si definisca un cammino *kL(r)-path* come un *k-path* che parte dal vertice 1 e termina nel vertice $i \in \bar{V}_r$ dopo aver visitato una ed una sola volta i vertici $\{i_1, ..., i_{r-1}\}$. Inoltre ogni vertice i_{s-1} è visitato prima del vertice i_s e nel cammino da i_{s-1} a i_s vengono visitati solo vertici di $\bar{V}_s$, $s = 2, ..., r-1$.

Si definisca con $f_r(k, i)$, $i \in \bar{V}_r$, il costo del *kL(r)-path* di costo minimo, di cardinalità k e che parte dal vertice 1 ed arriva nel vertice i dopo aver visitato una ed una sola volta i primi $r-1$ vertici $\{i_1, ..., i_{r-1}\}$ del longest path L. Si osservi che il vertice $i_0 = 1$ occupa la posizione 1 di ogni *kL-path* e che ogni vertice i_r può occupare le posizioni $k = np_{i_r} + 2, ..., n - nf_{i_r}$. La ricursione di programmazione dinamica per calcolare un *kL-tour* è la seguente.

Si inizializzi $f_0(1, 1) = 0$ e $f_0(1, j) = \infty$, $\forall\, j \in \bar{V}_1$.

Per $r = 1, ..., h$ si calcoli

$$f_r(k, i_{r-1}) = f_{r-1}(k, i_{r-1}) \text{ e } f_r(k, j) = \infty, \quad \forall\, j \in \bar{V}_r, \quad \forall\, k = np_{i_r} + 2, ..., n - nf_{i_r}$$

e quindi

$$f_r(k, j) = \min_{i \in \bar{V}_r}\{f_r(k-1, i) + c_{ij}\}, \quad \forall j \in \bar{V}_r,\ k = np_{i_r} + 2, ..., n - nf_{i_r}. \tag{4.6.2}$$

Il lower bound $LB2$ corrisponde al costo del *kL-tour* di costo minimo ed è dato da

$$LB2 = \min_{i \in \bar{V}_{h+1}} \{f_h(n,i) + c_{i1}\}. \tag{4.6.3}$$

Il *kL-tour* di costo $LB2$ si ottiene mediante le usuali operazioni di back-tracking, simili a quelle descritte nell'algoritmo `path_back`.

Esempio. Si riprenda il problema definito nell'esempio della sezione 1.1. In questo caso il longest path nel grafo delle precedenze è $(1, 4, 7, 8, 10, 1)$ con un costo pari a 154. Dopo la prima iterazione del rilassamento Lagrangiano con penalità nulle, si ottiene il lower bound 206.0 relativo al tour $(1, 11, 5, 4, 7, 5, 8, 9, 11, 10, 9, 11, 1)$ disegnato in rosso nella figura 3.

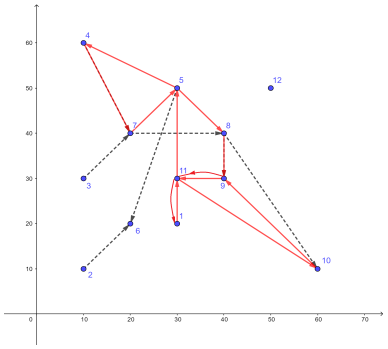


Fig. 3: Tour dopo la prima iterazione del Lagrangiano con penalità nulle

Come si vede, ci sono diversi vincoli della formulazione (P3) che sono violati da questo tour. Infatti, nonostante quattro vincoli di precedenza siano rispettati, le precedenze $(3, 7)$, $(2, 6)$ e $(5, 6)$ sono violate. Inoltre, definendo il grado del vertice i come il numero di volte che i compare nel tour, si hanno diversi vertici con grado D_i diverso da 1 e che quindi non rispettano i vincoli (1.4.5) e (1.4.6). In particolare, $d_2 = d_3 = d_6 = d_{12} = 0$, $d_5 = d_9 = 2$ e $d_{11} = 3$.

4.7 Miglioramento del lower bound LB2

Il *kL-tour* prodotto dalla ricursione (4.6.3) non è necessariamente elementare, così come avveniva per il *k-tour* descritto in precedenza. Inoltre un *kL-tour* può contenere loop di due vertici consecutivi. Ne segue che il lower bound $LB2$ può essere migliorato come segue:

a) Sia $\pi_r(k, j)$ il vertice che precede j nel cammino di costo $f_r(k, j)$. Sia $g_r(k, j)$ il costo *kL(r)-path* di costo minimo, di cardinalità k, che parte dal vertice 1 e termina nel vertice i dopo aver visitato una ed una sola volta i primi $r-1$ vertici $(i_0, ..., i_{r-1})$

del longest path L e tale che il vertice che precede j sia diverso da $\pi_r(k,j)$. È evidente che si può definire un algoritmo simile a `dp_forw` per calcolare$f_r(k,j)$ e $g_r(k,j)$ evitando loop di 2 vertici consecutivi.

b) $LB2$ può essere migliorato usando esattamente lo stesso metodo del subgradiente impiegato per migliorare LB.

Similmente a quanto descritto per il rilassamento *k-path* si può definire la funzione $f_r^{-1}(k,j)$ che rappresenta il costo del cammino ottimo, di cardinalità k, dal vertice j al vertice 1 e che visita una ed una sola volta i vertici $\{i_r, i_{r+1}, ..., i_{h+1}\}$ del longest path L. Analogamente può essere definita la funzione $g_r^{-1}(k,j)$. Tali funzioni si possono calcolare mediante le stesse ricursioni usate per calcolare $f_r(k,j)$ e $g_r(k,j)$, con lo stesso artificio descritto in precedenza, ovvero:

- sostituendo la matrice dei costi c_{ij} con la sua trasposta;
- invertendo il verso delle precedenze (la precedenza i *precede* j diviene j *precede* i).

Esempio. Si consideri nuovamente il problema definito dalla figura 1 nell'esempio della sezione 1.1. Nella seguente figura è disegnato in rosso il tour trovato dopo la 101-esima iterazione del lagrangiano: $(1,2,3,4,7,5,8,12,10,9,11,6,1)$. Il costo di questo tour è 239,000 ed è pari all'ottimo poichè il tour è elementare e rispetta tutti i vincoli di precedenza.

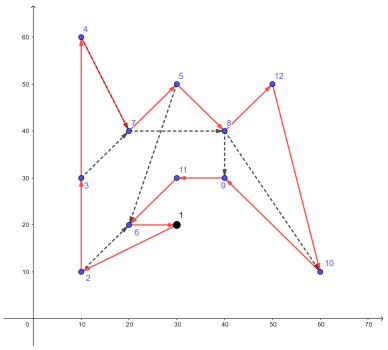


Fig. 4: Tour corrispondente al costo ottimo

4.8 Algoritmo esatto di programmazione dinamica DP-FW per risolvere il TSPP

In questa sezione viene descritto un algoritmo, chiamato DP-FW, per trovare la soluzione ottima del TSPP, basato sulla ricursione di programmazione dinamica (4.1.1). Il

metodo che viene proposto fa uso di funzioni di bounding per eliminare gli stati che non possono condurre ad alcuna soluzione ottima.

Sia $b^{-1}(S,i)$ un lower bound al costo di ogni soluzione ottima che contiene lo stato (S,i). Sia z_{UB} un valido upper bound al costo della soluzione ottima del TSPP. È evidente che se $f(S,i) + b^{-1}(S,i) \geq z_{UB}$, allora lo stato (S,i) non può condurre ad alcuna soluzione ottima di costo minore di z_{UB}. In seguito viene spiegato come $b^{-1}(S,i)$ possa essere calcolato usando i rilassamenti *k-path* e *kL-path*. Si ricordi che l'insieme degli stati è partizionabile in n sottinsiemi come $S_1 = \{(\{1\},1)\}$, $S_k = \{(S,j),\ S \subseteq V',\ |S| = k,\ j \in S, P_j \subseteq S\}\ k = 2,...,n$.

L'algoritmo `DP-FW` può essere schematizzato mediante i seguenti step:

1. Inizializzazione di $S_1 = \{(\{1\},1)\}$, $f(\{1\},1) = 0$ e $S_k = \emptyset$, $\forall k = 1,...,n$.
2. Ripetere per $k = 1,...,n-1$ i seguenti step 3 e 4.
3. Espandere ogni stato $(S,i) \in S_k$ come segue:
4. Per ogni vertice $j \in V' \setminus S$ tale che $P_j \subseteq S$ si consideri lo stato $S' = S \cup (\{j\},j)$, ottenuto aggiungendo a (S,i) il vertice j. Il costo di (S',j) è $h = f(S,i) + c_{ij}$. Si hanno i seguenti casi.
   - Caso 1: $h + b^{-1}(S',j) \geq z_{UB}$. Allora (S',j) viene rigettato.
   - Caso 2: $h + b^{-1}(S',j) < z_{UB}$ e $(S',j) \in S_{k+1}$. Si aggiunge (S',j) a S_{k+1} e si pone $f(S',j) = h$.
   - Caso 3: $h + b^{-1}(S',j) < z_{UB}$ e $(S',j) \in S_{k+1}$ ma $f(S',j) > h$. Si aggiorna $f(S',j) = h$.
5. Calcolo del costo ottimo
$$z = \min_{i \in V'} \{f(V',i) + c_{i1}\}.$$

4.9 Funzione di bounding

Il valore della funzione $b^{-1}(S,j)$ per un dato stato (S,j) dipende dal tipo di rilassamento utilizzato.

a) Rilassamento *k-path*.

$$b^{-1}(S,j) = \begin{cases} f^{-1}(n - |S| + 1, j) & \text{se } \pi^{-1}(n - |S| + j) \notin S, \\ h^{-1}(n - |S| + 1, j) & \text{altrimenti.} \end{cases}$$

b) Rilassamento *kL-path*.
Sia i_r l'ultimo vertice del longest path $L = (i_0, i_1, ..., 1_r, i_{r+1}, ..., i_h, i_{h+1})$ visitato dallo stato (S, j), ovvero $\{i_0, i_1, ..., 1_r\} \subseteq S$ mentre i vertici $i_{r+1}, ..., i_h$ non sono visitati da (S, j).

$$b^{-1}(S,j) = \begin{cases} f_r^{-1}(n - |S| + 1, j) & \text{se } \pi_r^{-1}(n - |S| + j) \notin S, \\ h_r^{-1}(n - |S| + 1, j) & \text{altrimenti.} \end{cases}$$

4.10 Algoritmo esatto/euristico EHDP per risolvere il TSPP

L'algoritmo di programmazione dinamica DP-FW descritto in precedenza può essere impiegato solo per problemi con al più 20 o 25 vertici. Ciò è dovuto al numero esponenziale di stati che vengono generati.

In questa sezione è descritto un algoritmo, chiamato EHDP, che può essere usato per problemi con più di 25 vertici. EHDP non garantisce la soluzione ottima, ma produce un lower bound $\theta_{min} \geq LB2$ e una soluzione ammissibile di costo z' non necessariamente ottima. EHDP deriva dall'algoritmo DP-FW come viene descritto in seguito.

Si associ a ciascuno stato (S, j) la label $\mathcal{L}(S, j) = f(S, j) + b^{-1}(S, j)$ che rappresenta un lower bound al costo di ogni soluzione ottima contenente lo stato (S, j). Sia $\mathcal{R}_k \subseteq S_k$ il sottinsieme di Δ stati di S_k che hanno la label $\mathcal{L}(S, j)$ di valore più piccolo. Si consideri una versione dell'algoritmo DP-FW in cui allo step 3, invece di espandere tutti gli stati S_k, si espandono solo gli stati di $\mathcal{R}_k$. Lo step 3 di DP-FW viene sostituito dal seguente:

3'. Sia $\mathcal{U} = \{(S, j) \in S_k : \mathcal{L}(S, j) < z_{UB}\}$. Si hanno i seguenti tre casi.

- Caso 1: $|\mathcal{U}| \leq \Delta$. Si pongono $\mathcal{R}_k = S_k$ e $\theta_k = \infty$.
- Caso 2: $|\mathcal{U}| > \Delta$. Si inseriscono in $\mathcal{R}_k$ i Δ stati di $\mathcal{U}$ che hanno la label $\mathcal{L}()$ di valore più piccolo. Si pone $\theta_k = \min_{(S,j) \in \mathcal{U} \setminus S_k}\{\mathcal{L}(S, j)\}$.
- Caso 3: $|\mathcal{U}| = 0$. DP-FW termina senza aver trovato alcuna soluzione di costo inferiore a z_{UB}. Si espande ogni stato $(S, i) \in \mathcal{R}_k$ come segue.

Sia z' il costo della soluzione trovata da EHDP allo step 5 e sia $\theta_{min} = \min_{k=1,...,n}\{\theta_k\}$.È evidente che se $z' \leq \theta_{min}$ allora z' è il costo della soluzione ottima mentre se $z' > \theta_{min}$ allora z' potrebbe non essere il costo ottimo, ma θ_{min} è un valido lower bound al costo ottimo.

4.11 Risultati preliminari di calcolo

In questa sezione sono descritti i risultati di calcolo ottenuti dal lower bound $LB2$ e dall'algoritmo `EHDP` su un sottoinsieme dei problemi test usati sia da Ascheuer et al. [5] che da Gouveia et al. [15]. Gli algoritmi descritti in questa tesi sono stati implementati in Fortran 77 usando il compilatore Fortran della Salford. Il metodo di Ascheuer et al. [5] è stato codificato in C e lo studio computazionale è stato eseguito su una SUN Ultra-SPARC Model 140, mentre lo studio computazionale di Gouveia et al. [15] è stato effettuato su un Intel(R) Core(TM) i7-4790 @3.60GHz con 8 GB RAM usando un single thread. Lo studio computazionale di questa tesi è stato effettuato su una macchina avente le stesse caratteristiche di quella di Gouveia et al. [15]. Pertanto, i tempi di calcolo conseguiti nella tesi sono perfettamente paragonabili con quelli di Gouveia et al. [15], mentre risulta più complesso il confronto con i tempi calcolo di Ascheuer et al. [5], in quanto non si è riusciti a trovare tabelle di confronto fra le prestazioni dei due diversi elaboratori. Tuttavia, è ragionevole supporre che il processore della SUN Ultra-SPARC usata da Ascheuer et al. [5] sia molto più lento del processore usato per questa tesi.

Per questa tesi sono state usate alcune delle istanze del Sequential Ordering Problem presenti nella libreria TSPLIB, in particolare le istanze con costi asimmetrici `p43.1-4` e `ry48p.1-4` usate da Gouveia et al. [15] e da Ascheuer et al. [5]. Inoltre sono state usate le due istanze istanze `ft53.3` e `ft53.4` usate da Ascheuer et al. [5], ma non da Gouveia et al. [15]. I risultati di Ascheuer et al. [5] sono stati ottenuti su una SUN Spark station 140 usando un time limit di circa 5 ore, mentre i risultati di Gouveia et al. [15] sono stati ottenuti su un laptop avente lo stesso processore di quello usato per questa tesi. Nel calcolo del lower bound $LB2$ sono state effettuate 400 iterazioni di ascent Lagrangiano e nell'algoritmo `EHDP` si è posto $\Delta = 400,000$. Il confronto dei risultati ottenuti con quelli conseguiti da Ascheuer et al. [5] è riportato nella tabella 4, mentre il confronto con Gouveia et al. [15] è riportato nelle tabelle 5 e 6. In grassetto sono evidenziati i bound migliori per ogni problema. Sono inoltre presentati il valor medio ed il massimo per i tempi di calcolo e il gap finale ottenuto.

La tabella 4 mostra che il lower bound $LB2$ proposto in questa tesi domina il lower bound di Ascheuer et al. [5] al vertice radice del tree search su 4 problemi, mentre il lower bound ottenuto da `EHDP` domina su 5 problemi quello di Ascheuer et al. [5] alla fine del tree search o al raggiungimento del time limit. Si noti che i lower bound proposti in questa tesi richiedono un tempo di calcolo irrilevante rispetto ai tempi di calcolo di Ascheuer et al. [5], qualunque sia il rapporto fra la CPU usata in questa tesi e quella della Sun Spark station usata da Ascheuer et al. [5]. In particolare `EHDP` dimostra l'ottimalità della soluzione prodotta per i problemi `ft53.4`, `p43.4` e `ry48p.4` in 1-2 secondi per problema mentre il metodo esatto di Ascheuer et al. [5] fallisce con un time limit di 5 ore per problema.

La tabella 5 mostra che il lower bound di Gouveia et al. [15], ottenuto senza la rimozione dei nodi, domina $LB2$ su tutti i problemi ad eccezione di `ry48p.4` e il lower

Tabella 4: Confronto dei lower bound di Ascheuer et al. [5] con $LB2$ e il lower bound prodotto da EHDP

			Ascheuer et al. [5]					Questa Tesi				
Prob.	**n**	$\|\tilde{A}\|$	**Upper bound**	**Bound Root**	**Bound finale**	**Time (min.)**	**Gap finale**	**Bound LB2**	**Bound EHDP**	**Upper bound**	**Time (sec.)**	**Gap finale**
ft53.3	54	48	10262	9115	9253	-	10.90%	**9326**	**9675**	10262	46.3	6.07%
ft53.4	54	63	14425	13871	14124	-	2.13%	**13930**	**14425**	14425	2.1	0.00%
p43.1	44	9	28140	**28063**	**28140**	06:21	0.00%	27894	27969	28140	37,7	0.61%
p43.2	44	20	28480	**28109**	**28319**	-	0.57%	28023	28174	28480	21.6	1.09%
p43.3	44	37	28835	**28160**	**28553**	-	0.99%	28062	28392	28835	56.8	1.56%
p43.4	44	50	83005	55884	65497	-	26.73%	**82801**	**83005**	83005	1.3	0.00%
ry48p.1	49	11	15805	**14969**	**15805**	208:03	0.00%	14888	15306	15805	83	3.26%
ry48p.2	49	23	16666	**15117**	15587	-	6.92%	15055	**15894**	16666	84.1	4.86%
ry48p.3	49	42	19894	**17248**	17813	-	11.68%	16474	**17994**	19894	84.5	10.56%
ry48p.4	49	58	31446	27165	29616	-	6.18%	**30383**	**31446**	31446	1.7	0.00%
media							6.61%					2.80%
max							26.73%					10.56%

Tabella 5: Confronto del lower bound di Gouveia et al. [15] senza rimozione dei vertici con $LB2$ e il lower bound prodotto da EHDP

			Gouveia et al. [15]				Questa Tesi				
Problema	**n**	$\|\tilde{A}\|$	**Upper bound**	**Bound finale**	**Time (sec.)**	**Gap finale**	**Bound LB2**	**Bound EHDP**	**Upper bound**	**Time (sec.)**	**Gap finale**
p43.1	44	9	28140	**28140**	11889	0.00%	27894	27969	28140	37.70	0.61%
p43.2	44	20	28480	**28426**	86001	0.19%	28023	28174	28480	21.60	1.09%
p43.3	44	37	28835	**28707**	45384	0.45%	28062	28392	28835	56.80	1.56%
p43.4	44	50	83005	82920	7888	0.10%	82801	**83005**	83005	1.30	0.00%
ry48p.1	49	11	15805	**15420**	60460	2.50%	14870	15357	15805	83.00	2.92%
ry48p.2	49	23	16666	15865	61439	5.05%	15055	**15894**	16666	84.10	4.86%
ry48p.3	49	42	19894	**18256**	28658	8.97%	16474	17994	19894	84.50	10.56%
ry48p.4	49	58	31446	28377	10982	10.81%	30383	**31446**	31446	1.70	0.00%
media					39088	3.51%				46	2.70%
max					86001	10.81%				84.50	10.56%

bound di EHDP su 5 degli 8 problemi considerati. Il gap medio finale ottenuto da EHDP è del 2.70% ed è quindi migliore di quello di Gouveia et al. [15], pari al 3.51%. Inoltre EHDP richiede un tempo medio di 46 secondi, mentre l'algoritmo di Gouveia et al. [15] richiede un tempo medio di 39088 secondi.

Mediante la rimozione dei nodi, come si evince dalla tabella 6, Gouveia et al. [15] migliorano il lower bound finale. Ll lower bound di Gouveia et al. [15] domina $LB2$ su tutti i problemi e il lower bound di EHDP su 6 degli 8 problemi. Tuttavia EHDP domina Gouveia et al. [15] nel problema ry48p.4. Infatti EHDP produce per questo problema in meno di 2 secondi un lower bound pari al costo della soluzione ottima mentre il metodo di Gouveia et al. [15] fallisce con tempi di calcolo di 15651 secondi. Si noti che sia EHDP che Gouveia et al. risolvono all'ottimo p43.4. Tuttavia EHDP impiega 1.3 secondi, mentre Gouveia et al. impiegano 1642 secondi. Nonostante il gap medio finale ottenuto da Gouveia et al. [15] con la rimozione dei nodi sia migliore di quello ottenuto da EHDP, il tempo di calcolo di Gouveia et al. [15] è in media più di 1000 volte superiore rispetto

Tabella 6: Confronto del lower bound di Gouveia et al. [15] con rimozione dei vertici con $LB2$ e il lower bound prodotto da EHDP

			Gouveia et al. [15]				Questa Tesi				
Problema	**n**	$\|\tilde{A}\|$	**Upper bound**	**Bound finale**	**Time (sec.)**	**Gap finale**	**Bound LB2**	**Bound EHDP**	**Upper bound**	**Time (sec.)**	**Gap finale**
p43.1	44	9	28140	**28140**	13959	0.00%	27894	27969	28140	37.70	0.61%
p43.2	44	20	28480	**28443**	125780	0.13%	28023	28174	28480	21.60	1.09%
p43.3	44	37	28835	**28756**	36861	0.27%	28062	28392	28835	56.80	1.56%
p43.4	44	50	83005	**83005**	1642	0.00%	82801	**83005**	83005	1.30	0.00%
ry48p.1	49	11	15805	**15477**	103330	2.12%	14870	15357	15805	83.00	2.92%
ry48p.2	49	23	16666	**15938**	93498	4.57%	15055	15894	16666	84.10	4.86%
ry48p.3	49	42	19894	**18705**	39339	6.36%	16474	17994	19894	84.50	10.56%
ry48p.4	49	58	31446	30920	15651	1.70%	30383	**31446**	31446	1.70	0.00%
media					53758	1.89%				46	2.70%
max					125780	6.36%				84.50	10.56%

a quello richiesto da EHDP.

4.12 Conclusioni

In questa tesi è stato descritto un nuovo metodo esatto di programmazione dinamica (DP) per risolvere il Traveling Salesman Problem con vincoli di precedenza (TSPP). È stato proposto un nuovo lower bound $LB2$ che si basa su un nuovo metodo di rilassamento dello spazio degli stati della ricursione esatta di DP, denominato *kL-path*, che tiene parzialmente conto dei vincoli di precedenza. Il bound che ne deriva è stato massimizzato usando il metodo del subgradiente. Inoltre è stato sviluppato un metodo di risoluzione del TSPP, denominato EHDP, basato sulla ricursione esatta di DP che usa il lower bound $LB2$ per eliminare stati che non conducono ad alcuna soluzione di costo inferiore a un dato upper bound. Tale metodo non garantisce la soluzione ottima ma restituisce sempre un lower bound θ_{min} superiore a $LB2$. I risultati di calcolo su un insieme di 10 problemi test noti in letteratura mostrano che i lower bound $LB2$ e θ_{min} dominano in media quelli di Ascheuer et al. [5] e sono confrontabili con quelli ottenuti da Gouveia et al. [15]. Tuttavia i tempi di calcolo necessari per calcolare $LB2$ e EHDP sono decisamente inferiori ai tempi richiesti da Ascheuer et al. [5] e Gouveia et al. [15]. In media il calcolo del lower bound di Gouveia et al. [15] richiede circa 50,000 secondi, mentre il calcolo dei lower bound proposti in questa tesi richiede in media 46 secondi.

Considerando gli elevatissimi tempi di calcolo richiesti dal lower bound di Gouveia et al. [15] risulta molto complesso (forse impossibile) realizzare un algoritmo esatto di tipo *branch and bound* per il TSPP che utilizzi tali lower bounds. Infatti un tale metodo esatto richiederebbe di calcolare il lower bound per diverse centinaia di nodi dell'albero decisionale, rendendo così impensabile di poter risolvere all'ottimo un'istanza di TSPP del tipo considerate in tabella 5 e 6 in tempi accettabili.

Appendice A

Cenni teorici

A.1 Rilassamento Lagrangiano

Si consideri il problema (P) di programmazione a numeri interi:

$$z(P) = \min c\,x \tag{A.1.1}$$
$$\text{s. t. } A\,x \geq b \tag{A.1.2}$$
$$B\,x \geq d \tag{A.1.3}$$
$$x \in (0,1)^n. \tag{A.1.4}$$

dove A è una matrice $m \times n$, B è una matrice $m_1 \times n$, b è un vettore di dimensione m, d è un vettore di dimensione m_1 e c e x sono vettori di dimensione n.

Il valore ottimo $z(LP)$ del rilassamento lineare (LP) del problema (P) fornisce un valido lower bound, ovvero $z(LP) \leq z(P)$. Il problema (LP) è ottenuto da (P) sostituendo (A.1.4) con $0 \leq x \leq 1$:

$$z(LP) = \min c\,x \tag{A.1.5}$$
$$\text{s. t. } A\,x \geq b \tag{A.1.6}$$
$$B\,x \geq d \tag{A.1.7}$$
$$0 \leq x \leq 1. \tag{A.1.8}$$

Tuttavia, in molti casi è proibitivo risolvere (LP) a causa di troppe variabili e/o vincoli oppure il valore $z(LP)$ è troppo distante da $z(P)$ e non è quindi utilizzabile in un algoritmo *branch & bound*.

Viene così definito il problema (RL_u), ottenuto da (P) rimuovendo i vincoli (A.1.2) e sottraendo dalla funzione obiettivo il termine $u(A\,x - b)$, dove $u \geq 0$ è il vettore dei

moltiplicatori lagrangiani:

$$L(u) = \min c\,x - u(A\,x - b) \tag{A.1.9}$$
$$\text{s. t. } B\,x \geq d \tag{A.1.10}$$
$$x \in (0,1)^n. \tag{A.1.11}$$

La funzione $L(u)$ è detta funzione Lagrangiana.

Teorema A.1.1 (Dualità Lagrangiana debole)**.**
Il valore ottimo $z(P)$ del problema (P) è maggiore o uguale al valore ottimo $L(u)$ del problema (RL_u) per ogni scelta di $u \geq 0$.

Dimostrazione. Se x^* è la soluzione ottima di (P), allora x^* è anche una soluzione ammissibile per (RL_u) $\forall\, u \geq 0$.

Si ha quindi $c\,x^* - u(A\,x^* - b) \geq L(u)$. Inoltre $u(A\,x^* - b) \geq 0$, da cui $z(P) = c\,x^* \geq L(u)$. □

Definizione. Il problema (D_L)

$$z(D_L) = \max_{u \geq 0} L(u)$$

è detto Lagrangiano Duale di (P).

Corollario A.1.2.
L'ottimo $z(D_L)$ del Lagrangiano Duale di (P) è un valido lower bound a $z(P)$, ovvero $z(D_L) \leq z(P)$.

Nel caso in cui $z(D_L) < z(P)$, si dice che esiste un *duality gap* fra il problema (P) e il problema (D_L).

Si supponga che l'ottimo di (D_L) sia ottenuto risolvendo $L(\bar{u})$ per un dato $\bar{u} \geq 0$, ovvero $z(D_L) = L(\bar{u})$. Sia $\bar{x}$ la soluzione ottima di (RL_u), cioè $z(D_L) = L(\bar{u}) = c\,\bar{x} - \bar{u}\,(A\,\bar{x} - b)$. Si consideri il caso in cui $\bar{x}$ sia anche l'ottimo di (P), ovvero $z(P) = c\,\bar{x}$. È evidente che se $\bar{u}\,(A\,\bar{x} - b) > 0$ allora $z(D_L) < z(P)$.

Teorema A.1.3 (Dualità Lagrangiana forte)**.**
Sia $\bar{x}$ la soluzione ottima di $L(\bar{u})$ per un dato $\bar{u} \geq 0$. Se $\bar{x}$ e $\bar{u}$ soddisfano le seguenti condizioni

(i) $A\,\bar{x} \geq b$

(ii) $\bar{u}\,(A\,\bar{x} - b) = 0$

allora $\bar{x}$ è la soluzione ottima di (P) ed inoltre $z(D_L) = L(\bar{u}) = z(P)$.

Dimostrazione. Poichè $\bar{x}$ soddisfa (i) allora è una soluzione ammissibile di (P) e quindi

$$c\,\bar{x} \geq z(P). \tag{A.1.12}$$

Per il Teorema di Dualità Lagrangiana debole e (ii), si ha

$$z(P) \geq L(\bar{u}) = c\,\bar{x} - \bar{u}\,(A\,\bar{x} - b) = c\,\bar{x}. \tag{A.1.13}$$

Combinando (A.1.12) e (A.1.13) si ottiene $c\,\bar{x} \geq z(P) \geq c\,\bar{x}$, ovvero $z(P) = c\,\bar{x}$ e quindi $\bar{x}$ è soluzione ottima di (P).

Per come è definito il problema (D_L) si ha che

$$z(D_L) \geq L(\bar{u}) \tag{A.1.14}$$
$$z(P) \geq z(D_L). \tag{A.1.15}$$

Inoltre per quanto dimostrato, vale

$$z(P) = L(\bar{u}) = c\,\bar{x}. \tag{A.1.16}$$

Da (A.1.14), (A.1.15) e (A.1.16) si ha $z(D_L) = z(P)$. □

Al fine di stabilire una relazione tra (D_L) ed il rilassamento lineare (LP) di (P) è utile riformulare (D_L) come un problema di programmazione lineare.

Definizione. Siano $X = \{x;\ Bx \geq d,\ x \in (0,1)\}$ e $conv(X)$ l'inviluppo convesso di tutti i punti di X.

Si osservi che l'ottimo del problema lagrangiano (RL_u) corrisponde ad un punto estremo di $conv(X)$.

Teorema A.1.4. *Il problema Lagrangiano Duale (D_L) corrisponde al seguente problema di programmazione lineare*

$$\begin{aligned} z(D_L) = \min\, & c\,x \\ s.\ t.\ \ & A\,x \geq b \\ & x \in conv(X) \end{aligned}$$

Dimostrazione. Si ricordi che

$$z(D_L) = \max_{u \geq 0} L(u)$$

e, per come è stato definito X, il problema (RL_u) diviene

$$L(u) = \min_{x \in X} c\,x - u\,(A\,x - b)$$

. Quindi il problema (D_L) può essere riscritto come

$$z(D_L) = \max_{u \geq 0} \left(\min_{x \in X} c\,x - u\,(A\,x - b) \right)$$

o anche

$$z(D_L) = \max_{u \geq 0} \left(\min_{x \in conv(X)} c\,x - u\,(A\,x - b) \right)$$

poichè $L(u)$ raggiunge l'ottimo in un punto estremo di $conv(X)$.

Siano x^i, $i = 1, ..., t$, i punti estremi di $conv(X)$. Il problema (D_L) può essere riscritto come

$$z(D_L) = \max_{u \geq 0} \left(\min_{1 \leq i \leq t} c\,x^i - u\,(A\,x^i - b) \right).$$

Quest'ultimo problema può essere riformulato mediante la programmazione lineare come segue:

$$\begin{aligned} z(D_L) = \max\, & v \\ \text{s. t. } & v \leq cx^i - u(Ax^i - b), \quad i = 1, ..., t \\ & v \text{ qualsiasi} \\ & u \geq 0. \end{aligned}$$

Il duale (DD_L) di questo problema è il seguente

$$\begin{aligned} z(D_L) = \min & \sum_{i=1}^{t} \lambda_i (cx^i) \\ \text{s. t. } & \sum_{i=1}^{t} \lambda_i = 1 \\ & \sum_{i=1}^{t} \lambda_i (Ax^i - b) \geq 0 \\ & \lambda_i \geq 0, \quad i = 1, ..., t. \end{aligned}$$

Si noti che $\sum_{i=1}^{t} \lambda_i (cx^i) = c \left(\sum_{i=1}^{t} \lambda_i x^i \right)$ e inoltre

$$\sum_{i=1}^{t} \lambda_i (Ax^i - b) = A \left(\sum_{i=1}^{t} \lambda_i x^i \right) - b \left(\sum_{i=1}^{t} \lambda_i \right)$$

. Quindi (DD_L) può essere riscritto come

$$z(D_L) = \min c \sum_{i=1}^{t} \lambda_i x^i \tag{A.1.17}$$

$$\text{s. t.} \quad \sum_{i=1}^{t} \lambda_i = 1 \tag{A.1.18}$$

$$A \sum_{i=1}^{t} \lambda_i x^i \geq b \sum_{i=1}^{t} \lambda_i \tag{A.1.19}$$

$$\lambda_i \geq 0, \quad i = 1, ..., t. \tag{A.1.20}$$

Si osservi che per ogni t-upla $\lambda_1, ..., \lambda_t$ che soddisfa i vincoli (A.1.18) e (A.1.20), il punto $x = \sum_{i=1}^{t} \lambda_i x^i$ appartiene a $conv(X)$. Quindi il problema (DD_L) può essere riscritto come

$$\begin{aligned} z(D_L) = \min\, & c\, x \\ \text{s. t.}\ & A\, x \geq b \\ & x \in conv(X) \end{aligned}$$

□

Teorema A.1.5.

$$z(D_L) \geq z(LP)$$

Dimostrazione. Sia $\bar{X} = \{x;\ Bx \geq d,\ 0 \leq x \leq 1\}$. Il problema (LP) può essere riscritto come

$$\begin{aligned} z(LP) = \min\, & c\, x \\ \text{s. t.}\ & A\, x \geq b \\ & x \in \bar{X}. \end{aligned}$$

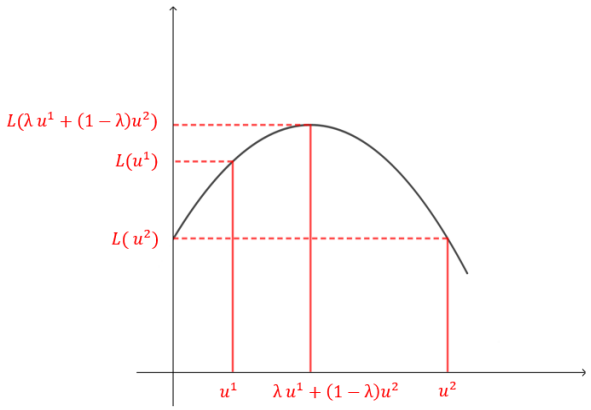
Per come è stato definito $\bar{X}$, è facile osservare che $conv(X) \subset \bar{X}$. Da questo e dal *Teorema A.1.4* segue quindi che $z(D_L) \geq z(LP)$. □

Teorema A.1.6.
La funzione Lagrangiana $L(u)$ è concava, ovvero $L(\lambda\, u^1 + (1-\lambda)u^2) \geq \lambda\, L(u^1) + (1-\lambda)\, L(u^2)$, $\lambda \in [0,1]$.

Dimostrazione. Siano $u^1,\ u^2 \geq 0$ e $u^0 = \lambda\, u^1 + (1-\lambda)u^2$ con $\lambda \in [0,1]$. Sia x^0 la soluzione ottima di (RL_{u^0}), ovvero $L(u^0) = \min c\, x^0 - u^0(A\, x^0 - b)$. Ovviamente x^0 è soluzione ammissibile di (RL_{u^1}) e (RL_{u^2}), quindi

$$L(u^1) \leq \min c\, x^0 - u^1(A\, x^0 - b) \tag{A.1.21}$$

$$L(u^2) \leq \min c\, x^0 - u^2(A\, x^0 - b). \tag{A.1.22}$$



Moltiplicando (A.1.21) per λ, (A.1.22) per $(1-\lambda)$ e sommando si ha:

$$\lambda\, L(u^1) + (1-\lambda)\, L(u^2) \leq \min c\, x^0 - (\lambda\, u^1 + (1-\lambda)\, u^2)\,(A\, x^0 - b) = L(u^0).$$

□

A.2 Metodo del subgradiente

Definizione. Un vettore y è detto subgradiente di $L(u)$ in $\bar{u}$ se $L(u) \leq L(\bar{u}) + y(u - \bar{u})$.

Come calcolare y? Sia $\bar{x}$ tale che

$$L(\bar{u}) \leq \min c\, \bar{x} - \bar{u}(A\, \bar{x} - b). \tag{A.2.1}$$

Per ogni $u \geq 0$ si ha

$$L(u) \leq \min c\, \bar{x} - u(A\, \bar{x} - b). \tag{A.2.2}$$

Sottraendo dalla (A.2.2) la (A.2.1), si ottiene

$$L(u) - L(\bar{u}) \leq -(A\, \bar{x} - b)\,(u - \bar{u})$$

ma anche

$$L(u) \leq L(\bar{u}) - (A\, \bar{x} - b)\,(u - \bar{u}).$$

Ne segue che $y = -(A\, \bar{x} - b)$ è un subgradiente di $L(u)$ in $\bar{u}$.

Il metodo del subgradiente è un metodo iterativo per risolvere il Lagrangiano Duale (D_L). Questo metodo genera una sequenza finita di punti $(u^1, ..., u^k)$ e quindi calcola

$$z(D_L) = \max_{u \in \{u^1, ..., u^k\}} L(u).$$

Di seguito è spiegato come si genera u^r in funzione di u^{r-1}. Sia x^{r-1} tale che $L(u^{r-1}) = c\,x^{r-1} - u^{r-1}(A\,x^{r-1} - b)$. Come dimostrato, si ha $L(u^r) \leq L(u^{r-1}) - (A\,x^{r-1} - b)\,(u^r - u^{r-1})$. Se si vuole che $L(u^r)$ possa essere maggiore di $L(u^{r-1})$, è necessario che

$$-(A\,x^{r-1} - b)\,(u^r - u^{r-1}) > 0. \tag{A.2.3}$$

Si noti che una scelta di u^r che soddisfi la suddetta condizione non è sufficiente per garantire $L(u^r) > L(u^{r-1})$. Di seguito è descritto un modo per definire u^r affinchè (A.2.3) sia verificata. Poichè A ha m righe, si avrà $u = (u_1, ..., u_m)$. Indicando con $a^i x \geq b_i$ la i-esima disequazione di $Ax \geq b$, la condizione (A.2.3) può essere riscritta come

$$-\sum_{i=1}^{m}(a^i\,x^{r-1} - b_i)\,(u_i^r - u_i^{r-1}) > 0. \tag{A.2.4}$$

Per soddisfare (A.2.4) è sufficiente determinare ogni u_i^r in modo che

$$-(a^i\,x^{r-1} - b_i)\,(u_i^r - u_i^{r-1}) > 0 \quad \forall\, i = 1, ..., m$$

Perciò:

- Se $(a^i\,x^{r-1} < b_i)$ allora x^{r-1} viola il vincolo i-esimo. Si definisce $u_i^r > u_i^{r-1}$.
- Se $(a^i\,x^{r-1} > b_i)$ allora x^{r-1} soddisfa il vincolo i-esimo. Si definisce $u_i^r < u_i^{r-1}$, imponendo $u_i^r \geq 0$.
- Se $(a^i\,x^{r-1} = b_i)$ allora x^{r-1} satura il vincolo i-esimo. Si definisce $u_i^r = u_i^{r-1}$.

Il metodo del subgradiente può essere schematizzato nei seguenti step.

1. Si inizializzano $u^1 = 0$, $r = 1$ e $LB = -\infty$.
2. Si risolve

$$\begin{aligned} L(u^r) = \min\, & c\,x - u^r(A\,x - b) \\ \text{s. t. } & B\,x \geq d \\ & x \in (0,1) \end{aligned}$$

   Sia x^r la soluzione ottima. Se $L(u^r) > LB$ allora si pone $LB = L(u^r)$ e $u^* = u^r$. Se $Ax^r \geq b$ e $u^r(A\,x - b) = 0$ allora x^r è la soluzione ottima di (P): STOP.

3. Si definiscano i moltiplicatori u^{r+1}:

$$u_i^{r+1} = \max\left\{0,\, u_i^r - \alpha\frac{z_{UB} - L(u^r)}{\sum_{i=1}^m \tilde{y}_i^2}\tilde{y}_i\right\},\ \ \forall\, i$$

   dove $\tilde{y}_i = a^i\,x^r - b_i$ e α è una costante tale che $0 < \alpha \leq 2$. Si pone $r \leftarrow r + 1$ e si torna allo step 2.

Di seguito sono riportate alcune osservazioni su quanto appena descritto:

- il metodo potrebbe non arrestarsi ed è quindi necessario imporre un numero massimo di iterazioni;
- è oppurtuno diminuire il valore di α, dimezzandolo se per Δ iterazioni consecutive $L(u) \leq LB$;
- i valori di α e Δ devono essere determinati sperimentalmente: tipicamente si ha $\alpha = 2$ e $\Delta = 30$.

Si consideri ora il caso in cui i vincoli del problema (P) siano misti:

$$\begin{aligned} z(P) = \min\ & c\,x \\ \text{s. t. } & A_1\,x \geq b_1 \\ & A_2\,x = b_2 \\ & A_3\,x \leq b_3 \\ & B\,x \geq d \\ & x \in \{0,1\}^n. \end{aligned}$$

In questo caso, aggiungendo le penalità $u^1 \geq 0$, $u^2 \in \mathbb{R}^{m_2}$ e $u^3 \leq 0$, il rilassamento lagrangiano è dato da

$$\begin{aligned} L(u) = \min\ & c\,x - u^1(A_1\,x - b_1) - u^2(A_2\,x - b_2) - u^3(A_3\,x - b_3) \\ \text{s. t. } & B\,x \geq d \\ & x \in \{0,1\}^n. \end{aligned}$$

Questo può essere riscritto come

$$\begin{aligned} L(u) = \min\ & (c - u^1A_1 - u^2A_2 - u^3A_3)x + u^1b_1 + u^2b_2 + u^3b_3 \\ \text{s. t. } & B\,x \geq d \\ & x \in \{0,1\}. \end{aligned}$$

Ad una generica iterazione del metodo del subgradiente, data $\bar{x}$ la soluzione ottima di $L(u)$, si procede nel modo seguente.

Si calcolino $y^1 = A_1\,\bar{x} - b_1$, $y^2 = A_2\,\bar{x} - b_2$ e $y^3 = A_3\,\bar{x} - b_3$. Si pongono $y = (y^1, y^2, y^3)$ e $t = \alpha \frac{z_{UB} - L(u)}{\sum_{i=1}^{m} y_i^2}$

$$\begin{aligned} & u_i^1 \leftarrow \max\left\{0, u_i^1 - t\,y_i^1\right\}, \quad i = 1, ..., m_1 \\ & u_i^2 \leftarrow u_i^2 - t\,y_i^2, \quad i = 1, ..., m_2 \\ & u_i^3 \leftarrow \min\left\{0, u_i^3 - t\,y_i^3\right\}, \quad i = 1, ..., m_3. \end{aligned}$$

Bibliografia

[1] N. Ascheuer, *Ein Schnittebenenverfahren für ein Reihenfolgeproblem in der flexiblen Fertigung*, Master's thesis, Universität Augsburg, Germany, 1989.

[2] N. Ascheuer, *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems*, PhD thesis Tech. Univ. Berlin, Avail. at URL http://www.zib.de/ZIBbib/Publications/, 1995.

[3] N. Ascheuer, L. Escudero, M. Grötschel, M. Stoer, *A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing)*, SIAM Journal on Optimization, 3, 25-42, 1993.

[4] N. Ascheuer, L. Escudero, M. Grötschel, M. Stoer *On identifying in polynomial time violated sub-tour elimination and precedence forcing constraints for the sequential ordering problem*, In Kannan, R. e Pulleyblank, W.R., editors, Integer Programming and Combinatorial Optimization, pages 19-28. University of Waterloo, Waterloo, Ontario, 1990.

[5] N. Ascheuer, M. Jünger, G. Reinelt, *A branch & cut algorithm for the asymmetric traveling salesman problem with precedence Constraints*, Computational Optimization and Applications, 17, 61-84, 2000.

[6] E. Balas e M. Fischetti, *A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets*, Mathematical Programming, 58, 325-352, 1993.

[7] E. Balas, M. Fischetti, W. R. Pulleyblank, *The precedence constrained asymmetric traveling salesman polytope*, Mathematical Programming, 68, 241-265, 1995.

[8] N. Christofides, A. Mingozzi, P. Toth, *State-Space Relaxation Procedures for the Computation of bound to Routing Problems*, Networks, 11, 145-164, 1981.

[9] M. Desrochers, G. Laporte, *Improvements to the Miller-Tucker-Zemlin sub-tour elimination constraints*, Operations Research Letters, 10, 27-36, 1991.

[10] L.F. Escudero, *An inexact algorithm for the sequential ordering problem*, European Journal of Operational Research, 37, 236-253, 1988.

[11] L.F. Escudero, *On the implementation of an algorithm for improving a solution to the sequential ordering problem*, Trabajos de Investigacion-Operativa, 3, 117-140, 1988.

[12] L.F. Escudero, M. Guignard, K. Malik, *A Lagrangean relaxandcut approach for the sequential ordering problem with precedence constraints*, Annals of Operations Research, 50, 219-237, 1994.

[13] M. Fischetti, P. Toth. *A polyhedral approach to the asymmetric traveling salesman problem*, Management Science, 43(11), 1520-1536, 1997.

[14] L. Gouveia, P. Pesneau, *On extended formulations for the precedence constrainted asymmetric traveling salesman problem*, Networks, 48(2), 77-89, 2006.

[15] L. Gouveia, P. Pesneau, M. Ruthmair, D. Santos, *Combining and projecting flow models for the (Precedence Constrained) Asymmetric Traveling Salesman Problem*, Networks, Wiley, In press, 1-19, 2017.

[16] L. Gouveia, J. M. Pires, *The asymmetric traveling salesman problem and a reformulation of the MillerTuckerZemlin constraints*, European Journal of Operational Research, 112, 134-146, 1999.

[17] L. Gouveia, M. Ruthmair, *Load-dependent and precedence-based models for pickup and delivery problems*, Computers & Operations Research, 63, 56-71, 2015.

[18] M. Grötschel, *Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme*, Hain, Meisenheim am Glan, 1977.

[19] M. Grötschel e M. Padberg, *Polyhedrak theory*, In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, The Traveling Salesman Problem. John Wiley & Sons, 1985.

[20] M. Held e R. M. Karp, *A dynamic programming approach to sequencing problems*, Journal for the Society for Industrial and Applied Mathematics 1, 10, 1962.

[21] A. N. Letchford, J.J. Salazar-González, *Stronger multicommodity flow formulations of the capacitated sequential ordering problem*, European Journal of Operational Research, 251(1), 74-84, 2016.

[22] K. Menger, *Zur allgemeinen kurventheorie*, Fundamanta Mathematicae, 10, 96-115, 1927.

[23] M. Padberg e G. Rinaldi, *An efficient algorithm for the minimum capacity cut problem*, Mathematical Programming, 47, 19-36, 1990.

[24] M. Padberg e G. Rinaldi, *Facet identification for the symmetric traveling salesman polytope*, Mathematical Programming, 47, 219-257, 1990.

[25] M. Padberg e M.R. Rao, *Odd minimum cut-sets and b-matchings*, Math. of OR, 7, 67-80, 1982.

[26] M. Timlin, *Precedence constrained routing. Master's thesis, Department of Combinatorics and Optimization*, Master's thesis, Department of Combinatorics and Optimization, University of Waterloo, 1989.

[27] M. Timlin, W.R. Pulleyblank, *Precedence constrained routing and helicopter scheduling: Heuristic design*, Interfaces, 22(3), 100-111, 1992.