UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea magistrale in
Scienze e Tecnologie dell'Informazione

Algoritmi di programmazione matematica per il Max Edge Weighted Clique problem with Multiple Choice constraints

RELATORE:
Prof. Alberto Ceselli

CORRELATORE:
Prof. Roberto Cordone

TESI DI LAUREA DI:
Yari Melzani
(703242)

Anno Accademico 2008/2009

Indice

Capitolo 1

Introduzione

La perplessità è l'inizio della conoscenza
– Kahlil Gibran

Molti problemi rilevanti in ambito sceintifico e tecnologico possono essere descritti e risolti con successo utilizzando modelli basati su grafi. In questa tesi viene affrontato il *Max Edge Weighted Clique problem with Multiple choice Contraints* (MEWCMC) che, dati un grafo pesato sia sui vertici che sui lati, e una partizione dell'insieme dei vertici, consiste nell'individuare una clique di peso massimo selezionando un vertice per ogni classe della partizione. Vediamo di seguito alcuni esempi applicativi, che potrebbero essere formulati come MEWCMC.

L'informatica si è sviluppata enormemente negli ultimi decenni, diventando fondamentale anche in ambiente lavorativo. L'evidente comodità ha permesso di redirigere l'archiviazione dei dati dal formato cartaceo al formato elettronico. In questo modo è possibile salvare informazioni anagrafiche dei clienti, le attività svolte e da svolgere, la gestione finanziaria e tanto altro ancora su supporti compatti e sicuri. Per fare ciò si ricorre molto spesso ai *DataBase*, ovvero archivi di dati correlati per argomento e strutturati in modo tale da consentire la gestione degli stessi dati (l'inserimento, la ricerca, la cancellazione ed il loro aggiornamento) da parte di applicazioni software.

La gestione di questi archivi viene effettuata da sistemi software progettati per consentire la creazione e manipolazione efficiente dei database chiamati *Database Management System* (DBMS). Grazie a questi potenti mezzi risulta molto più semplice effettuare delle ricerche e recuperare informazioni di ogni tipo e non solo, infatti opportune progettazioni permettono di ricavare informazioni utili che risultano "nascoste" tra i dati (ad esempio in alcuni casi potrebbe essere utile ricercare parole chiave classificando i risultati trovati secondo determinati criteri).

E' il caso degli strumenti di *Data Mining*, dall'inglese "estrazione dei dati", che permettono di effettuare statistiche e previsioni in modo efficiente sulla

base di grosse quantità di dati. A seconda della complessità dei calcoli ci si può avvalere di tecniche differenti di data mining:

- Estrazione, con tecniche analitiche all'avanguardia, di informazione implicita, nascosta, da dati già strutturati, per renderla disponibile e direttamente utilizzabile.

- Esplorazione ed analisi, eseguita in modo automatico o semiautomatico, su grandi quantità di dati allo scopo di scoprire pattern (schemi) significativi.

Gli strumenti di data mining hanno un utilizzo concreto e fondamentale in vari ambiti. Nella ricerca di mercato, ad esempio, i pattern possono mostrare informazioni utili sugli acquisti in specifiche aree geografiche, ai fini di creare promozioni mirate e quindi con maggior probabilità di successo. Ovviamente affinché l'informazione estratta sia significativa e utile è necessario che sia valida, precedentemente sconosciuta e comprensibile. Nella ricerca scientifica, ad esempio, il data mining può essere associato al concetto di *machine learning* e l'identificazione dei pattern può essere paragonata all'apprendimento di una relazione causale precedentemente ignota. In questo caso i pattern identificati possono essere a loro volta il punto di partenza per ipotizzare e quindi verificare nuove relazioni di tipo causale tra fenomeni.

Queste tecniche si avvalgono dunque di specifici algoritmi che devono garantire un'efficienza dei calcoli in qualsiasi situazione, anche in casi di grosse quantità di dati e di a innumerevoli richieste.

Tra gli algoritmi utilizzabili per l'estrazione dei dati troviamo gli algoritmi risolutivi per il Max Edge Weighted Clique Problem (MEWCP) proposti da Glover in [21]. Il Data Mining richiede per sua natura un gran numero di algoritmi che permettono di aggregare, strutturare ed estrarre i dati al fine di ricavare informazioni e conoscenza. In questo contesto è facile pensare ad un'esigenza reale da formulare come MEWCMC. Supponiamo di avere un insieme di dati V partizionato in classi $V_k \in K$ e di dover estrarre da questo m pattern, uno per classe, in relazione ad una certo valore di diversità che dovrà essere opportunamente definito. Supponendo che l'obiettivo consista nel massimizzare la diversità totale fra i pattern identificati, abbiamo ottenuto il MEWCMC.

Anche nell'ambito del progetto di reti di telecomunicazione sorge spesso l'esigenza di collocare i dispositivi che compongono i nodi della dorsale. Si consideri il progetto di una grande rete geografica, suddivisa in diverse aree dette cluster. Per garantire l'efficienza nella comunicazione di due qualsiasi nodi della rete è necessario definire una corretta topologia di rete. In una rete gerarchica è necessario posizionare per ogni cluster un nodo che si occupi di gestire il traffico verso nodi appartenenti a diversi cluster. L'obiettivo della progettazione consiste nell'eleggere per ogni area un nodo che sia connesso con gli altri nodi scelti in modo da formare una clique. L'obiettivo è aumentare

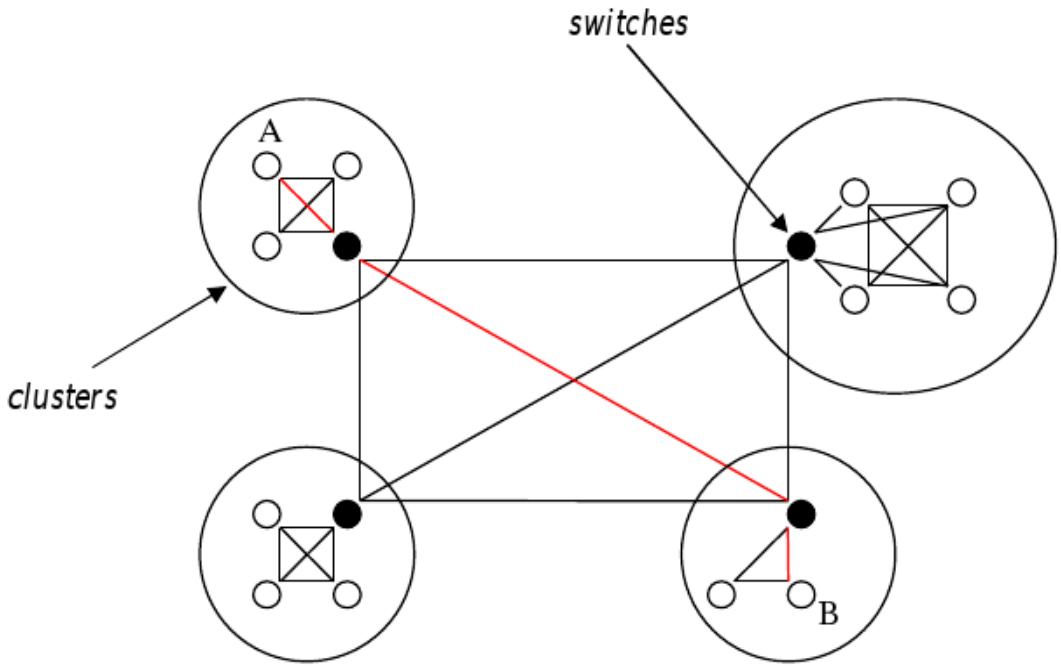

Figura 1.1: Esempio in cui viene mostrata la struttura della rete ed una possibile connessione (in rosso) tra i vertici A e B appartenenti a *cluster* diversi.

l'efficienza nella trasmissione ovvero massimizzare la banda passante e minimizzare il ritardo. La figura 1.1 mostra in modo molto rudimentale una rete composta da 4 cluster in ognuno dei quali viene eletto un nodo (pallino in nero) incaricato di fare da interfaccia verso i nodi delle altre aree.

Nell'ambito della biologia computazionale si riscontra frequentemente quello che è definito come *side-chain placement problem* [19]. Il problema consiste nel trovare la conformazione lineare di una proteina, in cui l'energia di legame tra gli amminoacidi sia minima e che, al contempo, assicuri un'adeguata stabilità strutturale (GMEC).

L'organizzazione spaziale degli atomi di una proteina viene detta conformazione. Questo termine si riferisce a qualsiasi stato strutturale che può, senza rompere nessun legame covalente, interconvertirsi in altro stato. Soltanto una, o poche, delle innumerevoli conformazioni teoricamente possibili di una proteina contenente centinaia di legami singoli è quella che tende a predominare nelle condizioni biologiche. Questa è di solito la conformazione termodinamicamente più stabile, quella che ha minore energia libera di Gibbs (G). Per una descrizione più approfondita dell'ambito biochimico vedere [25] e [22].

Un'ulteriore applicazione nasce nell'ambito della *management science* e riguarda l'interoperabilità delle imprese al fine di massimizzare l'efficienza. Si considerino N reparti di una grande impresa, ognuno dei quali ha la facoltà di scegliere, a dati costi, uno tra diversi possibili protocolli per interagire con gli altri reparti. I costi di interazione tra due uffici dipendono dai due protocolli

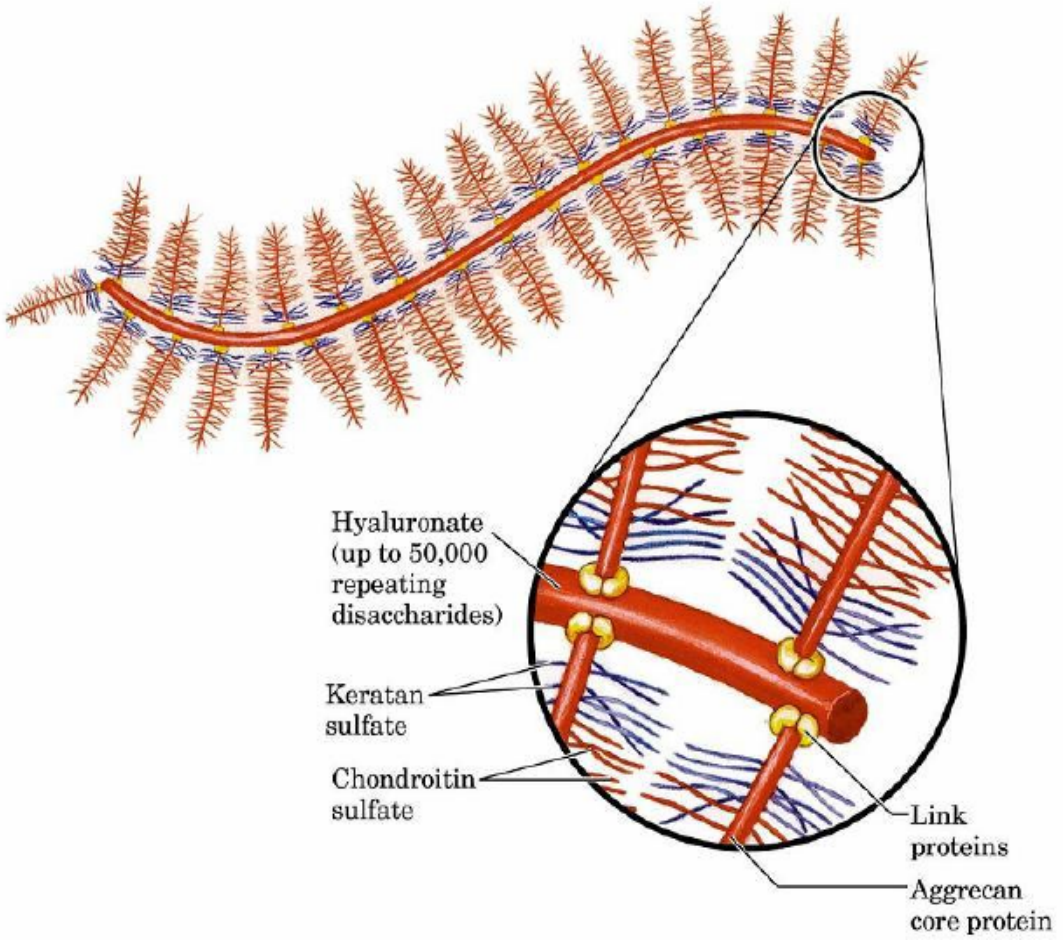

Figura 1.2: Aggregato proteoglicanico della matrice extracellulare.

scelti, che possono risultare più o meno compatibili tra loro. L'obiettivo di questo problema è minimizzare i costi complessivi cioè i costi di adozione del protocollo scelto per ogni ufficio più i costi di interazione per ogni coppia di uffici.

Tutti questi problemi hanno una struttura comune, riconducibile alla ricerca di una clique di peso massimo in un grafo opportuno e possono pertanto essere formulati come istanze di MEWCMC.

1.1 Analisi della letteratura

Il MEWCMC è un problema di ottimizzazione su grafo molto studiato nella versione senza i vincoli di Multiple Choice. La versione del problema così ottenuta è conosciuta come Max Edge Weighted Clique Problem (**MEWCP**) e consiste nel trovare una clique di peso massimo in un grafo pesato opportuno. Per questo sottoproblema sono stati proposti in letteratura sia algoritmi esatti che algoritmi euristici. Infatti dato che il problema è $NP - difficile$ ([13], si riduce al problema della *clique* di peso massimo), gli algoritmi trovano ampio spazio in scenari reali dove la dimensione delle istanze può rendere impraticabile l'utilizzo anche di un algoritmo esatto di enumerazione.

Il MEWCP è stato originariamente proposto da Glover nel 1977 ([15]) il quale presentò una formulazione lineare intera in grado di risolvere istanze di piccole dimensioni, con meno di 40 elementi, a causa del numero quadratico di variabili decisionali necessarie. Nel corso degli anni sono comparse numerose pubblicazioni ([31, 11, 30]) riguardanti gli aspetti teorici e computazionali del problema. Sungsoo e altri hanno proposto in [31] varie formulazioni e un algoritmo basato sui piani di taglio così come Sørensen e altri in [30]. Il problema è stato anche affrontato mediante un algoritmo di Branch and Bound basato sul rilassamento Lagrangeano da Hunting [20] e da Bosio nella tesi di laurea ([6]). Ghilardi ([14]) ha proposto per il MEWCP un algoritmo esatto parallelo con lo scopo di risolvere anche istanze di dimensioni altrimenti difficilmente attaccabili.

Per quanto riguarda gli algoritmi euristici, sono stati proposti numerosi lavori [28, 29, 1, 9, 24], alcuni dei quali si basano su algoritmi di Tabu search. Glover ha presentato un lavoro basato su questo tipo di euristica in [1], così come Aringhieri e altri in [24]. Dalla letteratura si evince che per risolvere il MEWCP, alcune euristiche funzionano meglio di altre; ad esempio, in una precedente pubblicazione ([24]), abbiamo mostrato che il Tabu Search risulta più efficiente rispetto ad algoritmi di tipo GRASP [28, 9] (Greedy Randomized Adaptive Search Procedure), ovvero strategie di ricerca basate sulla generazione di una soluzione ammissibile *Greedy Randomized* raffinata col contributo della ricerca locale.

1.2 Scopo del lavoro

Il MEWCMC è un problema di ottimizzazione su grafo molto generico e, come molti altri problemi simili, è $NP - difficile$. Attualmente non sono noti algoritmi in grado di risolvere in tempo polinomiale questa classe di problemi: per ottenere metodi di risoluzione efficaci, è necessario studiare il problema in modo approfondito. Nella tesi vengono proposti modelli matematici e algoritmi per la risoluzione esatta del problema, basati su tecniche di enumerazione implicita. In particolare è stato dapprima costruito un modello di programmazione lineare intera, un modello con funzione obiettivo quadratica e uno di programmazione semidefinita ottenuto tramite riformulazione del modello quadratico. Un'analisi sperimentale ha evidenziato che i solutori commerciali, utilizzando questi modelli, sono in grado di risolvere solo istanze di dimensioni ridotte.

Quindi, per superare le limitazioni dei solutori generici è stato sviluppato e implementato un algoritmo euristico di Tabu Search, in grado di determinare delle soluzioni di buona qualità, in tempi molto rapidi.

Infine è stato studiato e implementato un algoritmo ad-hoc, che sfrutta euristiche e tecniche di rilassamento per esplorare efficientemente l'albero di decisione.

Le prestazioni degli algoritmi implementati sono state valutate tramite un'estesa campagna sperimentale. Questa ha evidenziato che l'impiego delle tecniche di programmazione semidefinita sono particolarmente efficaci: permettono di ottenere bound duali stretti, riducendo così il tempo necessario all'enumerazione implicita.

I risultati ottenuti dall'algoritmo ad-hoc sono stati confrontati con quelli dal solutore commerciale *ILOG Cplex 11.2* a cui è stato fornito in ingresso il modello lineare intero rafforzato con dei vincoli di taglio. L'algoritmo ad-hoc risolve istanze di dimensioni molto maggiori rispetto a Cplex, comportandosi meglio sia in termini di tempo di calcolo che in termini di qualità dei bound. È stato inoltre possibile verificare che l'euristica di Tabu Search implementata, nella maggior parte delle istanze testate, riesce ad individuare la soluzione ottima in pochi secondi.

1.3 Struttura della tesi

La tesi è divisa in 7 parti. Nel capitolo 2 vengono proposte una serie di formulazioni matematiche che verranno poi utilizzate per ottenere un confronto con il solutore commerciale Cplex e ottenere delle tecniche di rilassamento, in grado di fornire una stima per eccesso del valore della soluzione ottima.

Nel capitolo 3 vengono illustrate alcune euristiche in grado di fornire rapidamente delle buone soluzioni ammissibili. In particolare viene presentato un algoritmo di Tabu Search e un insieme di tecniche che permettono di ricavare un bound primale sfruttando le informazioni contenute nelle soluzioni fornite dai rilassamenti proposti.

Nel capitolo 4 vengono descritte le tecniche di rilassamento prese in esame ed utilizzate nel seguito per la progettazione di un algoritmo esatto. Successivamente vengono presentati due algoritmi di bounding basati su tecniche di rilassamento del problema originario: il primo bound è basato sulla struttura combinatoria del problema (4.1), mentre il secondo si basa sulla riformulazione di MEWCMC come problema di programmazione semidefinita (4.2).

Nel capitolo 5, abbiamo presentato degli algoritmi per il calcolo dei bound e abbiamo mostrato come sono stati inseriti all'interno di una procedura di Branch and Bound appositamente studiata, in grado di garantire l'ottimalità della soluzione trovata. Nel capitolo (6) vengono messi al confronto gli algoritmi implementati, analizzando i risultati ottenuti ed evidenziando qualità e difetti di ognuno di essi. Infine, nel capitolo (7), vengono presentate e discusse le conclusioni sul lavoro svolto.

Capitolo 2

Formulazione del problema

Non può esservi vera conoscenza
laddove non si può applicare
nessuna delle scienze matematiche
– Leonardo da Vinci

Dato un grafo $G(V, E)$, dove V è un insieme di n vertici ed $E \subseteq V \times V$ è un insieme di ℓ lati. Siano $w_E : E \to \mathbb{R}$ e $w_V : V \to \mathbb{R}$ due funzioni peso che associano rispettivamente ad ogni lato e ad ogni vertice un numero reale. Sono dati m sottoinsiemi $V_1, ...V_k$ di V, tali che $\bigcup_{k=1}^{m} V_k = V$ e $V_k \bigcap V_l = \emptyset$, $\forall k \neq l$ che costituiscono, quindi, le m classi di una partizione $K = \{V_1, V_2, .., V_k\}$ di V.

Il Max Edge Weighted Clique problem with Multiple Choice contraints (MEWCMC) consiste nel trovare una clique in G, ovvero un sottografo completo $M \subseteq V$ tale che $(i, j) \in E$ $\forall i, j \in M$ avente peso massimo calcolato come $z(M) = \frac{1}{2} \sum_{v_i \in M} \sum_{v_j \in M} w_{ij}$.

I vincoli di Multiple Choice impongono che M contenga esattamente 1 vertice per ogni classe, ovvero $|M \cap V_k| = 1$ $\forall V_k \in K$.

2.1 Formulazione quadratica

La formulazione più naturale per MEWCMC è la seguente: 2.1, 2.2, 2.3.

$$z = max \sum_{i \in V} \sum_{j \in V} w_{ij} x_i x_j \tag{2.1}$$

$$s.t. \quad \sum_{i \in V_k} x_i = 1 \qquad \forall k \in K \tag{2.2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{2.3}$$

Tale formulazione si ottiene introducendo n variabili binarie x_i (2.3) , una per ogni vertice $i \in V$ tali per cui $x_i = 1$ se il vertice associato $i \in M$ fa parte

della clique, $x_i = 0$ altrimenti. Infatti per ogni coppia di vertici $i, j \in M$ deve valere necessariamente $x_i x_j = 1$ in modo che possa essere considerato il peso associato all'arco che sottende i due vertici della clique. Se uno dei due termini del prodotto è pari a 0 allora il contributo w_{ij} è nullo.

Definiamo la matrice W in cui gli elementi $w_{i,j}$ sono i pesi associati ai lati (i, j), $i, j \in V$ tali che

$$w_{ij} = \begin{cases} w_E(v_i, v_j) & se\, v_i \neq v_j \\ w_V(v_i) & se\, v_i = v_j \end{cases}$$

L'obiettivo (2.1) consiste nel massimizzare la somma dei pesi associati ai lati che compongono la clique. I vincoli di *Multiple Choice* (2.2) garantiscono che sia selezionato un solo vertice in ogni classe V_k.

Questa formulazione ha il vantaggio di essere molto semplice, tuttavia il termine $x_i x_j$ rende la funzione obiettivo quadratica.

In commercio esistono solutori molto efficienti in grado di risolvere modelli di programmazione quadratica in modo molto efficiente se il problema fornito è convesso. Il modello da noi fornito non è convesso e quindi non possiamo trarre un particolare vantaggio da un solutore per la programmazione quadratica. Per questo motivo abbiamo dovuto linearizzare la funzione obiettivo per ottenere un modello di programmazione lineare intero, per il quale esistono un numero sterminato di solutori, sia commerciali che non, in grado di fornire tecniche molto efficienti per la risoluzione all'ottimo del MEWCMC.

2.2 Una formulazione lineare intera

Dato che non possiamo sfrutta il vantaggio offerto da un solutore per la programmazione quadratica convessa, abbiamo proposto un modello di programmazione lineare intera (PLI). La tecnica che permette di linearizzare un modello quadratico è nota in letteratura. Di seguito viene riportato il modello di PLI:

$$z = max \sum_{i \in V} \sum_{j \in V} w_{ij} y_{ij} \tag{2.4}$$

$$s.t. \quad y_{ij} \leq x_i \qquad \forall i, j \in V \tag{2.5}$$

$$y_{ij} \leq x_j \qquad \forall i, j \in V \tag{2.6}$$

$$y_{ij} \geq x_i + x_j - 1 \qquad \forall i, j \in V \tag{2.7}$$

$$\sum_{i \in V_k} x_i = 1 \qquad \forall k \in K \tag{2.8}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{2.9}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{2.10}$$

Introduciamo a tale proposito n^2 ulteriori variabili binarie y_{ij} che assumono valore 1 se entrambi i vertici i e j fanno parte della clique M, $y_{ij} = 0$ altrimenti.

Ovvero $y_{ij} = 1 \iff x_i = x_j = 1 \iff x_i \cdot x_j = 1$. La funzione obiettivo in forma lineare è rappresentata in 2.4. Per linearizzare il termine quadratico $x_i \cdot x_j$, introduciamo tre nuovi gruppi di vincoli di attivazione (2.5, 2.6 e 2.7), rispettivamente per le variabili decisionali x_i e x_j che vincolano $y_{ij} = 1$ alla condizione $x_i = x_j = 1$.

I vincoli 2.7 sono necessari per le istanze in cui compaiono nella matrice W pesi negativi. Infatti senza tali vincoli sarebbe possibile avere $x_i = x_j = 1$ con la corrispondente variabile variabile $y_{ij} = 0$.

Il vincolo 2.8 rimane invariato e continua ad imporre che venga selezionato esattamente un vertice $i \in V$ per ogni $V_k \in K$.

La formulazione presentata può essere rafforzata introducendo il vincolo di taglio (2.11). Da $x_i = 1$ segue $\sum_{j \in V} y_{ij} = m$ ovvero il vertice $i \in V$ selezionato per far parte della clique M dovrà avere esattamente $(m-1)$ lati incidenti; il termine m deriva dal fatto che deve essere considerato anche il peso w_{jj} del vertice $j = i$.

$$\sum_{j \in V} y_{ij} = m x_i \ \ \forall i \in V \tag{2.11}$$

Dato che il modello è lineare si possono utilizzare strumenti simbolici di modellizzazione ([8]) e solutori *general purpose* come *ILOG CPLEX* in grado di trovare la soluzione ottima.

Un esempio di modello e istanza in formato AMPL è riportato in appendice (B).

2.3 Una formulazione semidefinita per il MEWCMC

I risultati ottenuti con Cplex erano attesi: nonostante sia possibile rappresentare il MEWCMC con un modello lineare, la sua natura è intrinsecamente quadratica. In letteratura sono stati studiati alcuni problemi [10, 16] simili al MEWCMC ed è stato mostrato che il rilassamento della formulazione semidefinita può essere molto stretto se applicato a problemi che hanno una natura di questo tipo.

Formalmente, la *programmazione semidefinita* (semidefinite programming - **SDP**) considera quei problemi che ottimizzano una funzione lineare soggetta al vincolo che una combinazione affine di matrici simmetriche sia semidefinita positiva. Una trattazione specifica sulla programmazione semidefinita e sulla dualità sono state riportate in appendice A. Intuitivamente possiamo raccogliere le variabili del problema in una matrice che è richiesta essere semidefinita positiva. Ci sono vari modi per ottenere la formulazione semidefinita per MEWCMC. Seguendo l'idea presentata in [16] abbiamo utilizzato come base di partenza la formulazione quadratica (2.1). Nel capitolo A.4 è mostrato come sia possibile ottenere un modello semidefinito partendo anche dalla formulazione lineare.

Introduciamo una matrice quadrata incognita $Y = \{x_i x_j\}$ di dimensione

n, per ogni $i, j \in V$. La matrice Y è equivalente ad una matrice ottenuta moltiplicando ogni elemento del vettore delle incognite x per il suo trasposto:

$$Y = xx^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

Sviluppando tale prodotto prodotto si ottiene:

$$Y = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & x_2x_2 & \cdots & x_2x_n \\ \cdots & \cdots & \cdots & \cdots \\ x_nx_1 & \cdots & \cdots & x_nx_n \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & x_2^2 & \cdots & x_2x_n \\ \cdots & \cdots & \cdots & \cdots \\ x_nx_1 & \cdots & \cdots & x_n^2 \end{bmatrix}$$

ma poiché ogni variabile decisionale x_i può assumere solo il valore 0 o 1, ed $x_i^2 = x_i$, i valori sulla diagonale della matrice Y possono essere riscritti come segue:

$$Y = \begin{bmatrix} x_1 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & x_2 & \cdots & x_2x_n \\ \cdots & \cdots & \cdots & \cdots \\ x_nx_1 & \cdots & \cdots & x_n \end{bmatrix}$$

Gli elementi diagonali di Y rappresentano i vertici selezionati mentre quelli non diagonali rappresentano i lati.

Il MEWCMC può quindi essere formulato come il seguente problema di programmazione semidefinita:

$$\begin{aligned} z = \quad & max\, W \bullet Y && (2.12) \\ s.t. \quad & rank(Y) = 1 && (2.13) \\ & \sum_k S_k \bullet Y = 1 \quad k = 1, ..., m && (2.14) \\ & Y \succeq 0 && (2.15) \end{aligned}$$

Il simbolo $\bullet$ indica il prodotto di *Frobenius* fra matrici calcolato come $\sum_i \sum_j W_{ij} X_{ij}$.

Con $rank(Y)$ indichiamo il rango della matrice Y, mentre $Y \succeq 0$ impone che Y sia semidefinita positiva. I è la matrice di identità di dimensione n. In particolare i vincoli (2.13, 2.15) garantiscono che la matrice Y sia costituita da valori interi e possa essere rappresentata tramite il prodotto xx^T, dove x è il vettore delle variabili decisionali x_i. Infatti consideriamo un generico elemento Y_{ij}; affinché la matrice abbia rango 1 deve essere necessariamente $Y_{ii} = Y_{ij} = Y_{ji} = Y_{jj}$. Osservando che gli elementi $Y_{ij} = Y_{ii}Y_{jj}$ la condizione precedente può essere verificata solo se i valori nella matrice sono binari. I vincoli di multiple choice 2.14 impongono che per ogni classe $k \in K$, ci sia

esattamente un vertice selezionato. Infatti dal vincolo di multiple choice lineare (2.8) possiamo costruire una matrice S_k per ogni classe $k \in K$, definita come segue:

$$\sum_{i \in V_k} Y_{ii} = \sum_{i \in V} \sum_{j \in V} (S_k)_{ij} Y_{ij} = S_k \bullet Y = 1 \quad \forall k \in K$$

2.4 Formulazione dei vincoli di multiple choice

Ad esempio, consideriamo un'istanza di MEWCMC in cui $|V| = 6$ vertici ed $m = 2$ classi ognuna costituita da 3 elementi tali che $V_1 = \{v_1, v_2, v_3\}$, $V_2 = \{v_4, v_5, v_6\}$. La matrice $S_k$corrispondente ad ogni classe $k \in K$ è definita come segue:

$$S_1 = \begin{bmatrix} \mathbf{1} & & & & & 0 \\ & \mathbf{1} & & & & \\ & & \mathbf{1} & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0 & & & & & 0 \\ & 0 & & & & \\ & & 0 & & & \\ & & & \mathbf{1} & & \\ & & & & \mathbf{1} & \\ 0 & & & & & \mathbf{1} \end{bmatrix}$$

Nelle matrici di esempio sono stati omessi gli elementi nulli per chiarezza. Si osservano gli 1 sulla diagonale in corrispondenza dei vertici che fanno parte alla classe; i vincoli di multiple choice 2.14 impongono che la matrice Y debba avere esattamente un valore pari a 1 in quelle posizioni.

Una soluzione ammissibile per l'istanza potrebbe essere:

$$Y = \begin{bmatrix} 0 & & & & & 0 \\ & \mathbf{1} & & \mathbf{1} & & \\ & & 0 & & & \\ & \mathbf{1} & & \mathbf{1} & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix}$$

Infatti,

$$S_1 \bullet Y = \begin{bmatrix} \mathbf{1} & & & & & 0 \\ & \mathbf{1} & & & & \\ & & \mathbf{1} & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \bullet \begin{bmatrix} 0 & & & & & 0 \\ & \mathbf{1} & & \mathbf{1} & & \\ & & 0 & & & \\ & \mathbf{1} & & \mathbf{1} & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} = 1$$

$$S_2 \bullet Y = \begin{bmatrix} 0 & & & & & 0 \\ & 0 & & & & \\ & & 0 & & & \\ & & & \mathbf{1} & & \\ & & & & \mathbf{1} & \\ 0 & & & & & \mathbf{1} \end{bmatrix} \bullet \begin{bmatrix} 0 & & & & & 0 \\ & \mathbf{1} & & \mathbf{1} & & \\ & & 0 & & & \\ & \mathbf{1} & & \mathbf{1} & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} = 1$$

Inoltre la matrice $Y \succeq 0$ è semidefinita positiva, ha rango unitario e può essere ottenuta come prodotto xx^T, dove $x^T = [0, 1, 0, 1, 0, 0]$ è il vettore soluzione per la formulazione quadratica (2.1).

Nella sottosezione 4.3 sono stati proposti degli ulteriori vincoli in grado di rafforzare la formulazione.

Capitolo 3

Bound primali

Possiamo vedere nel futuro solo per un piccolo tratto, ma possiamo pure vedere che in questo piccolo tratto c'e' molto da fare.
– Alan M. Turing, Computing Machinery and Intelligence, Mind, 1950

La ricerca di una soluzione ammissibile per il MEWCMC è semplice: dato che il grafo è completo, è sufficiente scegliere un vertice in ogni partizione ma. Invece trovare la soluzione ottima è computazionalmente difficile. Infatti, dato che il problema è $NP-difficile$, a meno che $P = NP$, non si conoscono algoritmi polinomiali che diano la garanzia di ottimalità della soluzione trovata. In questo capitolo descriveremo alcuni algoritmi in grado di determinare un bound primale per il problema.

Il primo algoritmo proposto è un'euristica basata su Tabu Search (TS) (sottosezione 3.3), in grado di fornire rapidamente delle buone soluzioni per il problema.

L'algoritmo di TS può anche essere sfruttato per produrre bound primali di partenza per un algoritmo esatto e permette di affrontare istanze di dimensioni considerevoli, senza fornire tuttavia garanzia di ottimalità.

In questo capitolo viene inoltre mostrata la tecnica del rounding (sottosezione 3.1) in grado di sfruttare la soluzione del rilassamento semidefinito per ricavare una soluzione ammissibile intera. Nella sezione 3.2 viene spiegato come ottenere un bound primale valido partendo dalla soluzione fornita dal rilassamento combinatorio.

3.1 Rounding

La tecnica del *rounding* (arrotondamento) è largamente usata nei problemi di ottimizzazione per determinare, in modo semplice, una soluzione primale ammissibile per il problema originario partendo da una soluzione frazionaria, ottenuta dal calcolo del bound duale. In particolare, nei problemi di *programmazione lineare intera* (PLI), alcune tecniche di rilassamento (ad esempio il

rilassamento continuo) prevedono l'eliminazione dei vincoli di integralità imposti sulle variabili decisionali; il problema così rilassato consente di ottenere un bound duale per il problema di partenza costituito da una soluzione frazionaria, in genere non ottima.

Il rounding consiste nel trasformare le variabili frazionarie in valori interi in modo che la soluzione risultante sia ammissibile.

Nel nostro caso la soluzione del rilassamento semidefinito (sottosezione 4.2) è costituita da una matrice $Y \geq 0$ semidefinita positiva così composta: sulla diagonale $\mathbf{x} = diag(Y)$ si trovano i valori frazionari delle variabili decisionali x_i, $i = 1, .., |V|$.

I vincoli di Multiple Choice, impongono che per ogni classe $V_k \in K$ sia selezionato esattamente un vertice (2.14). Sulla diagonale della matrice soluzione, la somma dei valori frazionari associati alle variabili appartenenti alla stessa classe V_k è pari a 1. Il *rounding* tiene conto delle sole informazioni contenute nella diagonale arrotondando a 1 la variabile che ha il valore frazionario massimo in ogni classe, ed arrotando a 0 ogni altra variabile. Al termine della procedura otteniamo una soluzione ammissibile per MEWCMC. La politica di arrotondamento implementata premia la selezione dei vertici che danno maggior contributo nella funzione obiettivo del problema rilassato. In particolare, quando il rilassamento semidefinito restituisce una soluzione che ha valore ottimo, questa coincide con il valore della soluzione ottenuta dal rounding della stessa.

Si osserva tuttavia che quando la qualità del bound duale trovato è scarsa, i valori frazionari delle variabili associate alla classe V_k tendono ad uniformarsi tutte a $\frac{1}{|V_k|}$. In questo caso particolare l'arrotondamento diventa di fatto una scelta casuale.

3.1.1 Un esempio di rounding

In figura 3.1 è mostrato l'esempio di una matrice Y (per chiarezza, solo la sezione triangolare bassa) che rappresenta i valori frazionari, ottenuti mediante il rilassamento semidefinito di una istanza con 3 classi, formate da 3 elementi ciascuna. Rappresentiamo l'insieme dei vertici $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$, e il seguente schema di partizionamento: $V_1 = \{v_1, v_2, v_3\}$, $V_2 = \{v_2, v_3, v_4\}$, $V_3 = \{v_5, v_6, v_7\}$.

Il vettore $\mathbf{x} = diag(Y)$, contiene i valori seguenti.

$\mathbf{x} = [0, 0.18, 0.82, 0, 0.37, 0.63, 0.23, 0.77, 0]$. Come si vede, i vincoli di Multiple Choice sono rispettati: la somma dei valori frazionari associati alle variabili, per ogni classe è pari a 1.

Dopo aver eseguito l'arrotondamento otteniamo un vettore soluzione $\mathbf{x} = [0, 0, 1, 0, 0, 1, 0, 1, 0]$ in cui ci sono esattamente m vertici selezionati: uno per ogni classe V_k. La *clique* M che rappresenta la soluzione primale è quindi $M = \{v_3, v_6, v_8\}$.

	1	2	3	4	5	6	7	8	9
1	0.00								
2	0.00	**0.18**							
3	0.00	0.00	**0.82**						
4	0.00	0.00	0.00	0.00					
5	0.00	**-0.12**	**0.49**	0.00	**0.37**				
6	0.00	**0.30**	**0.33**	0.00	0.00	**0.63**			
7	0.00	**0.20**	**0.03**	0.00	**-0.12**	**0.35**	**0.23**		
8	0.00	**-0.02**	**0.79**	0.00	**0.49**	**0.28**	0.00	**0.77**	
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figura 3.1: Esempio di una matrice Y semidefinita positiva che rappresenta i valori frazionari, ottenuti mediante il rilassamento semidefinito di una istanza con 3 classi, formate da 3 elementi ciascuna. Y è simmetrica ma per chiarezza è stata riportata solo la matrice triangolare bassa.

3.2 Euristica primale combinatoria

In questo paragrafo presentiamo una tecnica euristica in grado di determinare una soluzione primale basata sulle proprietà combinatorie del problema.

Per ogni vertice $i \in V_s$ il valore

$$\eta(i) = w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t} \{\frac{w_{jj}}{m-1} + w_{ij}\}$$

fornisce una stima del contributo del vertice i nella classe V_k.

Se prendiamo in ogni classe $V_k \in K$ il vertice

$$i_k^* = \arg\max_{i \in V_k} \{w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t} \{\frac{w_{jj}}{m-1} + w_{ij}\}\}$$

che ha il contributo $\eta(i)$ massimo, otteniamo una soluzione ammissibile per MEWCMC. Una volta effettuata la scelta dei vertici in ogni classe, si ottiene una clique il cui peso può essere facilmente valutato.

Nella nella sottosezione 4.1 abbiamo mostrato come sfruttare l'idea proposta per definire un bound duale combinatorio valido per il problema.

La figura 3.2 mostra la soluzione ammissibile ottenuta mediante la tecnica basata sui contributi $\eta(i)$.

3.3 Tabu Search

Il Tabu Search rappresenta un'evoluzione del classico "metodo di discesa", utilizzato per trovare il minimo di una funzione reale f su un insieme S (spazio delle soluzioni). Tale metodologia consiste nel partire da una soluzione iniziale, esplorarne un opportuno intorno, scegliere una mossa che porti dalla soluzione corrente alla migliore dell'intorno. Il metodo di discesa si arresta in ogni punto di ottimo locale: se nell'intorno non esistono soluzioni migliori di quella corrente, la ricerca termina. Se l'insieme S delle soluzioni non è convesso, la

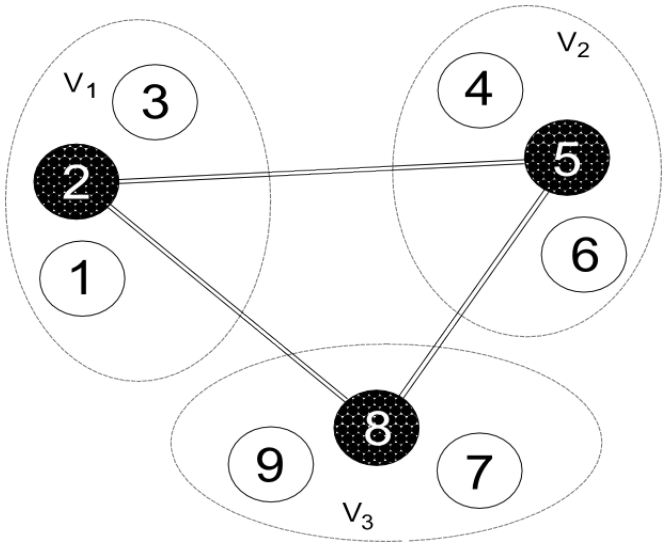

Figura 3.2: La figura mostra un esempio in cui il rilassamento combinatorio determina una soluzione ammissibile per il problema di partenza.

soluzione ottima individuata dal metodo di discesa potrebbe essere ben lontana dalla soluzione ottima globale.

La tecnica del Tabu Search venne proposta da Fred Glover [12] proprio per ovviare a questo problema. Innanzitutto, per poter allontanarsi dai minimi locali, il Tabu Search consente mosse peggioranti. Tuttavia, così facendo, vi è il rischio di compiere dei cicli, allontanandosi e tornando immediatamente nel minimo locale. Il presupposto del Tabu Search consiste allora nel rendere proibite (tabu) le ultime mosse eseguite nel cammino di ricerca, in modo da impedire che l'algoritmo possa tornare sui propri passi e ricadere in un minimo locale già esplorato. L'approccio basato su Tabu Search sfrutta l'utilizzo di una memoria per aumentare l'efficacia del processo di ricerca, ovvero viene tenuta traccia sia delle informazioni locali (come il valore corrente della funzione obiettivo) sia di alcune informazioni relative all'itinerario percorso. Tali informazioni vengono impiegate per guidare la scelta nell'intorno corrente verso una nuova soluzione migliorante.

Notazione

Denotiamo con s una soluzione per $MEWCMC$ composta da m elementi $v_i \in N,\ i = 1, ..., m$. Sia M la *clique* rappresentata dagli e_i elementi, $M = \{v_i \in N : i = 1, .., m\}$. Introduciamo l'operatore $\sigma(i)$ per stabilire l'appartenenza dell'elemento v_i alla classe $V_k \in K$.

$$\sigma(i) = k \ se\, i \in V_k$$

Data una clique M e un elemento $v_i \in V$, indichiamo con $\hat{W}_i^M$ la somma dei pesi associati ai lati tra vertici in M ed i. In questa somma escludiamo i vertici per cui $\sigma(i) = \sigma(j)\, \forall j \in M$. Definiamo inoltre con $\widetilde{w}_i$ il peso dell'elemento v_i. Definiamo

$$\hat{W}_i^M = \sum_{j \in M | \sigma(j) \neq \sigma(i)} w_{ij}$$

Indichiamo con $W_i^M = \hat{W}_i^M + w_V(i)$
dove $W_V(i)$ è il peso associato al vertice i.

3.3.1 Componenti di base dell'algoritmo

Le componenti generali del Tabu Search richiedono:

- la costruzione di una soluzione iniziale da cui poter partire con la ricerca;
- l'individuazione dell'intorno di una soluzione;
- la scelta di una politica per la selezione di una soluzione appartenente a tale intorno.

E' necessario inoltre:

- rappresentare le soluzioni proibite mediante l'utilizzo di *liste tabu;*
- determinare le condizioni di terminazioni della ricerca.

3.3.2 Soluzione di partenza

La nostra implementazione è molto semplice: una soluzione di partenza è costruita prendendo il primo vertice di ogni classe della partizione. La soluzione così determinata risulta ammissibile e può essere utilizzata come punto di partenza per il Tabu Search.

3.3.3 Intorno di una soluzione

Presa una clique M ammissibile per MEWCMC, definiamo un intorno $I(M)$ costituito da tutte le possibili soluzioni M' ottenute scambiando per ogni classe $V_k \in K$ l'elemento selezionato $j \in V_k \cap M$ con un diverso elemento della stessa classe.

Tale scambio di elementi, permette di conservare l'ammissibilità della soluzione.

L'esplorazione dell'intorno consiste nel determinare la migliore soluzione M^* tale che

$$z(M^*) = max_{M' \in I(M)} z(M)$$

ovvero si tratta di valutare quale fra tutti i possibili scambi è quello più vantaggioso.

Il valore della nuova M^* può essere calcolato come segue. Supponiamo che lo scambio comporti l'uscita del vertice v_j e l'ingresso di v_i e sia $M_j = M \backslash \{v_j\}$. Prima dell'operazione il valore della funzione obiettivo è

$$z(s) = \sum_{v_h \in M} (\frac{1}{2}\hat{W}_h^M + w_V(h)) = W_j^M + \sum_{v_h \in M_j} (\frac{1}{2}\hat{W}_h^{M_j} + w_V(h))$$

Analogamente dopo aver sostituito il vertice $v_j$con $v_i$otteniamo la soluzione M' e la corrispondente clique M'. Sia $M_i^{'} = M^{'}\backslash\{v_i\}$, possiamo calcolare il valore della funzione obiettivo come

$$z(M') = \sum_{v_h \in M'} (\frac{1}{2}\hat{W}_h^{M^{'}} + w_V(h)) = W_i^{M'} + \sum_{v_h \in M_i^{'}} (\frac{1}{2}\hat{W}_h^{M_i^{'}} + w_V(h))$$

Dalla relazione $\Delta z = z(M^{'}) - z(M) = (W_i^{M'} - W_j^M)$ si deduce che il valore della nuova funzione obiettivo $z(M') = z(M) + \Delta z$; poiché $M\backslash\{v_j\} = M'\backslash\{v_i\}$. Risulta quindi che

$$z(M') = \sum_{v_h \in M} (\frac{1}{2}\hat{W}_h^M + w_V(h)) + (W_i^{M'} - W_j^M)$$

3.3.4 Liste tabu

Le liste tabu hanno una duplice funzione: cercare di evitare l'esplorazione ciclica di soluzioni e pilotare la ricerca verso migliori ottimi locali. L'idea consiste nell'associare ad ogni iterazione dell'algoritmo una lista di soluzioni vietate (tabu). Una strategia efficace tiene traccia delle ultime soluzioni memorizzando le azioni necessarie per trasformare una soluzione in un'altra [12].

Nel nostro caso, lo scambio di due vertici comporta che entrambi siano marcati *tabu, e* non possano essere coinvolti in altri scambi, per un fissato numero di iterazioni (*lunghezza della lista tabu*). Le liste tabu così implementate impediscono di esplorare tutte le soluzioni dell'intorno che si ottengono coinvolgendo uno o più vertici che sono stati marchiati tabu.

Una soluzione tabu non viene però scartata se il valore della funzione obiettivo associato è globalmente migliorante (**criterio di aspirazione**).

Per questo problema abbiamo introdotto due liste tabu ℓ_e e ℓ_u di lunghezza diversa con l'obiettivo di controllare rispettivamente l'entrata e l'uscita degli elementi dalla soluzione. La lunghezza delle due liste è un parametro dell'algoritmo ed è stata valutata sperimentalmente.

Dalla campagna di test sulle istanze i valori riportati di seguito si sono rivelati la scelta migliore.

$$\begin{cases} |\ell_e| = 8 \\ |\ell_u| = 1 \end{cases}$$

La lunghezza delle due liste tabu rimane costante durante tutta l'esecuzione.

3.3.5 Condizioni di terminazione

A differenza della ricerca locale pura, in cui l'algoritmo termina una volta raggiunto un ottimo locale, il TS necessita di una o più condizioni di terminazione.

Durante la fase di sperimentazione, se la lunghezza delle liste tabu non è ben tarata si osserva chiaramente l'esplorazione ciclica delle soluzioni. Nonostante questo fenomeno sia difficile da eliminare suggerisce strategie di terminazione efficaci. Il TS termina se per un determinato numero I_c di iterazioni non si riscontrano miglioramenti globali.

L'unica condizione di terminazione rimane il raggiungimento del numero I_{tot} di iterazioni totale definito a priori.

Il parametro relativo alle condizioni di terminazione utilizzato nelle simulazioni è il seguente:

$\{\ I_c = 1000$

Capitolo 4

Bound duali

L'uomo e' sempre pronto a morire per un'idea,
purchè essa non gli sia chiara.
– Legge di Elridge sulla guerra

Preso un generico problema P tale che l'ottimo sia dato da $\max f_{(x)}$ con $x \in X$ si dice rilassamento di P il nuovo problema P' definito come

$$\begin{array}{ll} \max & f'_{(x)} \\ s.t. & x \in X' \end{array}$$

che rispetta le seguenti proprietà:

1. $f'_{(x)} \geq f_{(x)} \ \ \forall x \in X$
2. $X' \supseteq X$

si dice che $f'_{(x)}$ è un problema con un insieme di soluzioni ammissibili più ampio (o identico) e una funzione obiettivo che nell'insieme di partenza assume valori migliori (o uguali).

In questo capitolo presentiamo prima un rilassamento basato sulle proprietà combinatorie di MEWCMC che frutta la proposizione (1) e un rilassamento ottenuto dal modello semidefinito che sfrutta la proposizione (2).

4.1 Bound duale combinatorio

Di seguito presentiamo un bound duale combinatorio per il MEWCMC. Nella sottosezione 3.2 abbiamo illustrato l'euristica combinatoria dalla quale abbiamo preso spunto per definire un bound duale combinatorio.

Presa una classe V_s consideriamo il contributo del vertice i come

$$w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t} \{\frac{w_{jj}}{m-1} + w_{ij}\}$$

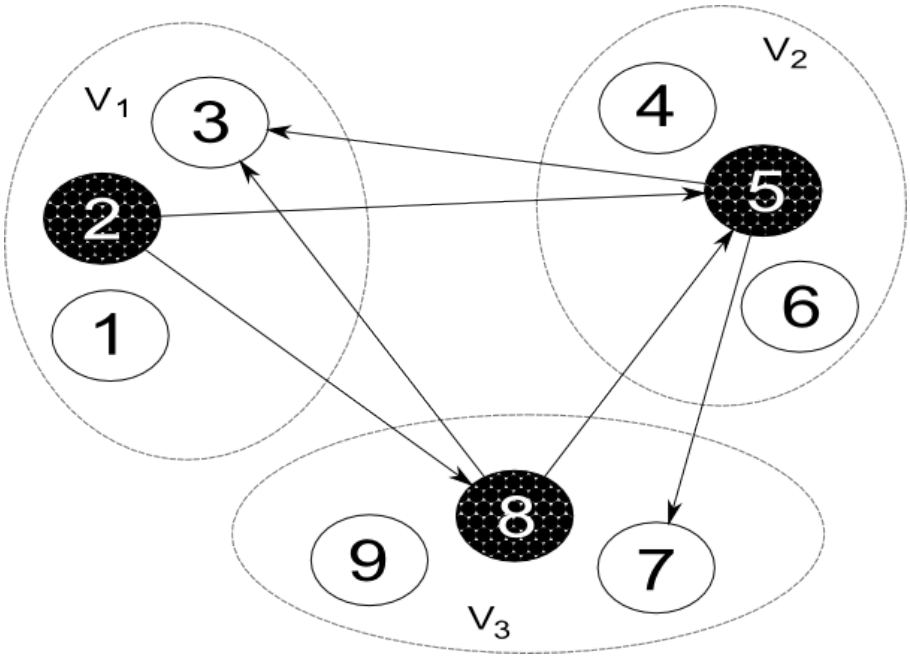

Figura 4.1: Un esempio di rilassamento combinatorio applicato a MEWCMC

Questo termine rappresenta una stima per eccesso del contributo al peso complessivo di una clique ottenuta scegliendo il vertice $i \in V_s$. Il vertice i_k^* che ha il massimo contributo nella classe V_k è individuato da

$$i_k^* = \arg\max_{i \in V_k}\{w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t}\{\frac{w_{jj}}{m-1} + w_{ij}\}\}$$

Il contributo associato a tale vertice è il massimo nella classe V_k e può essere calcolato come

$$DB_c(V_s) = \frac{1}{2}\max_{i \in V_s}\{w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t}\{\frac{w_{jj}}{m-1} + w_{ij}\}\} \tag{4.1}$$

Il termine $DB_C(V_s)$ rappresenta di fatto una stima per eccesso del contributo che la classe $V_k$darà al valore soluzione totale. Intuitivamente se sommiamo il contributo di ogni classe della classe $V_k \in K$ otteniamo una stima per eccesso della migliore soluzione che possiamo individuare per il problema P. Tale valore rappresenta un bound duale combinatorio valido per il problema P calcolato come

$$DB_c(P) = \frac{1}{2}\sum_{V_s \in K}\max_{i \in V_s}\{w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t}\{\frac{w_{jj}}{m-1} + w_{ij}\}\} \tag{4.2}$$

La figura 4.1 mostra un esempio di come potrebbe essere una soluzione ottenuta mediante il metodo sopra descritto. I lati sono rappresentati orientati per poter evidenziare quali sono i lati scelti per ogni vertice selezionato (pallino nero). Il pallino individua il vertice della classe che presenta il massimo contributo. Gli archi uscenti coinvolgono i vertici delle altre classi che concorrono

a determinare il contributo totale. Il bound duale è calcolato quindi tenendo conto di m contributi indipendenti, uno per ogni classe. Per questo motivo si osserva che, nonostante siano rispettati i vincoli di multiple choice, la soluzione mostrata non è ammissibile. Infatti il peso della clique composta dai soli vertici selezionati ha un peso non maggiore della soluzione mostrata in figura (4.1).

Mostriamo di seguito, con l'aiuto della tecnica della maggiorazione, la correttezza che il bound duale combinatorio appena presentato. Rappresentiamo come

$$\max f = \sum_{V_s \in K} \sum_{i \in V_s} w_{ii} y_{ii} + \frac{1}{2} \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} w_{ij} y_{ij} \tag{4.3}$$

la funzione obiettivo di MEWCMC

Vogliamo trovare il massimo della funzione f in modo che i vertici selezionati siano una clique e siano rispettati i vincoli di multiple choice ovvero sia selezionato esattamente un vertice per ogni classe. L'idea consiste nello spostare il peso associato ai vertici sugli $(m-1)$ lati incidenti ad ogni vertice selezionato. Definiamo quindi

$$\tilde{w}_{ij} = \frac{w_{ii}}{m-1} + \frac{w_{jj}}{m-1} + w_{ij}$$

Sostituendo $\tilde{w}_{ij}$ nell'espressione 4.3 otteniamo

$$\max f = \sum_{V_s \in K} \frac{1}{2} \sum_{i \in V_s} \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij}$$

Dato che i vincoli di multiple choice richiedono che sia selezionato esattamente un vertice per ogni classe, possiamo scrivere scegliere fra tutti i vertici di ogni classe, quello che ha il contributo massimo, ovvero

$$\sum_{i \in V_s} \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij} \leq \max_{i \in V_s} \{ \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij} \}$$

Sommando i contributi di tutte le classi $k \in K$ otteniamo analogamente

$$\max f = \sum_{V_s \in K} \frac{1}{2} \sum_{i \in V_s} \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij} \leq \frac{1}{2} \sum_{V_s \in K} \max_{i \in V_s} \{ \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij} \} \tag{4.4}$$

Considerando il valore originale di $\tilde{w}_{ij}$ e sapendo che $\sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \tilde{w}_{ij} y_{ij}$ sono $(m-1)$ termini possiamo scrivere la disequazione 4.4

$$\begin{aligned}
\max f = & \ \tfrac{1}{2} \textstyle\sum_{V_s \in K} \max_{i \in V_s} \{ (m-1) \frac{w_{ii}}{(m-1)} + \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \{ \frac{w_{jj}}{m-1} + w_{ij} \} \} & = \\
\max f \leq & \ \tfrac{1}{2} \textstyle\sum_{V_s \in K} \max_{i \in V_s} \{ w_{ii} + \sum_{V_t \in K, t \neq s} \sum_{j \in V_t} \{ \frac{w_{jj}}{m-1} + w_{ij} \} \} & \leq \\
\leq & \ \tfrac{1}{2} \textstyle\sum_{V_s \in K} \max_{i \in V_s} \{ w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t} \{ \frac{w_{jj}}{m-1} + w_{ij} \}
\end{aligned}$$

Che è esattamente l'espressione usata per il calcolo del bound duale combinatorio (4.2).

Nella sezione 3.2 viene mostrato come ottenere ottenere una soluzione ammissibile partendo da una soluzione del rilassamento combinatorio.

Se il valore del rilassamento coincide con la soluzione ottima per il problema duale allora, tale soluzione, è ammissibile per il problema di partenza. Supponiamo che il rilassamento combinatorio determini la soluzione ottima, con l'aiuto dell'esempio riportato in figura 3.2, mostriamo che il valore del rilassamento combinatorio calcolato coincide esattamente con il valore della soluzione ottima. Utilizzando la definizione 4.2 e sviluppando, otteniamo:

$$\begin{aligned}
\frac{1}{2}\Big[w_2 + (\frac{w_5}{2} + w_{25} + \frac{w_8}{2} + w_{28}) \quad &+ \\
w_5 + (\frac{w_2}{2} + w_{25} + \frac{w_8}{2} + w_{58}) \quad &+ \\
w_8 + (\frac{w_2}{2} + w_{28} + \frac{w_5}{2} + w_{58})\,\Big] \quad &= \\
= \frac{1}{2}[2w_2 + 2w_5 + 2w_8 + 2w_{25} + 2w_{28} + 2w_{58}] \quad &= \\
= w_2 + w_5 + w_8 + w_{28} + w_{25} + w_{58}
\end{aligned}$$

Il valore ottenuto è proprio il peso, dei vertici e dei lati, della clique formata dai vertici selezionati.

4.2 Rilassamento semidefinito

Negli ultimi anni si è osservata una crescente attenzione nei confronti della programmazione semidefinita; parte dell'interesse deriva certamente dalla qualità dei bound che essa permette di ottenere anche su problemi di ottimizzazione generici [33, 27, 2, 17, 18] soprattutto se come MEWCMC hanno una natura quadratica. Un altro motivo dell'interesse è legato alla significatività dei contributi forniti dalla comunità scientifica, sia in termini teorici che pratici. I primi algoritmi specifici, basati sul metodo del punto interno [5, 23, 4], hanno implementazioni sempre più efficienti, disponibili anche sotto forma di librerie dinamiche.

4.2.1 Il rilassamento di MEWCMC

Dal modello di programmazione semidefinita (2.3) ottenuto mediante riformulazione del modello quadratico si può ottenere un bound duale per il MEWCMC. A questo proposito vengono rilassati il vincolo di rango (2.13). La matrice soluzione Y ottenuta rimane semidefinita positiva ma può contenere valori frazionari. In figura 3.1 è mostrato un esempio di soluzione Y ottenuta con il rilassamento semidefinito.

Dato che lo spazio può non essere strettamente convesso, ci possono essere più soluzioni duali non ammissibili che hanno lo stesso valore ottimo. Può verificarsi che la matrice Y ottenuta dal rilassamento abbia valore ottimo per il problema, senza tuttavia avere rango 1. Si osserva tuttavia che il bound duale è tanto più stretto quanto meno frazionari sono i valori nella matrice. Si osserva infatti che una soluzione ottima intera mostra sulla diagonale solo valori binari che indicano chiaramente, per ogni classe della partizione, il vertice selezionato.

Attivando in modo selettivo i vincoli proposti della sezione 2.3 si possono ottenere dei miglioramenti sulla qualità del bound duale anche se il beneficio si paga con un tempo di calcolo più lungo. È stato fondamentale quindi trovare un compromesso tra la qualità del rilassamento e il tempo necessario ad ottenerlo.

4.3 Rafforzamento della formulazione semidefinita

Il modello di programmazione semidefinita è stato rilassato per ottenere un bound duale valido per il MEWCMC. A questo proposito sono stati studiati alcuni vincoli in grado di rafforzare la formulazione e quindi permettere di aumentare la qualità del bound.

4.3.1 Vincolo di cardinalità

Fra i vincoli più semplici che si possono imporre troviamo senz'altro il vincolo di cardinalità $I \bullet Y = m$. Tale vincolo agisce solamente sulla diagonale della matrice soluzione Y imponendo che siano selezionati m vertici in totale.

4.3.2 Improved Multiple Choice

Da questo momento chiameremo i vincoli di multiple choice, simple multiple choice (SMC).

Introduciamo dei vincoli di multiple choice rafforzati (IMC) come segue. È facile verificare che

$$\begin{array}{ll} \sum_{j \in V_k} x_i x_j = x_i & \forall i \in V \\ & \forall V_k \in K \end{array}$$

Infatti volendo rappresentare tali vincoli nella forma $M_{ij} \bullet Y = 0$, dobbiamo definire delle opportune matrici M_{ik}.

Identifichiamo due casi: $\begin{cases} i \in V_k & IMC_A \\ i \notin V_k & IMC_B \end{cases}$

Nel primo caso: IMC_A, otteniamo $|V|$ vincoli totali, $|V_k|$ per ogni classe $k \in K$. Sulla base dell'esempio 2.4 rappresentiamo le matrici $M_{ik} : k = 1, i = \{1, 2, 3\}$ associate alla classe V_2 e i relativi vincoli IMC_A

$$M_{11} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{1} & & & 0 \\ \mathbf{1} & 0 & & & & \\ \mathbf{1} & & 0 & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \Rightarrow M_{11} \bullet Y = 0$$

$$M_{21} = \begin{bmatrix} 0 & \mathbf{1} & & & & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & & & \\ & \mathbf{1} & 0 & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \Rightarrow M_{21} \bullet Y = 0$$

$$M_{31} = \begin{bmatrix} 0 & & \mathbf{1} & & & 0 \\ & 0 & \mathbf{1} & & & \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \Rightarrow M_{31} \bullet Y = 0$$

Nel secondo caso $i \notin V_k$ otteniamo $n(m-1)$ vincoli IMC_B. Analogamente a quanto sopra riportato, riportiamo le matrici associate all'esempio 2.4 considerando il vertice $i = \{1, 2, 3\}$ e la classe V_2 otteniamo:

$$M_{12} = \begin{bmatrix} \mathbf{-1} & & & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & 0 & & & & \\ & & 0 & & & \\ \mathbf{1} & & & 0 & & \\ \mathbf{1} & & & & 0 & \\ \mathbf{1} & & & & & 0 \end{bmatrix} \Rightarrow M_{12} \bullet Y = 0$$

$$M_{22} = \begin{bmatrix} 0 & & & & & 0 \\ & \mathbf{-1} & & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & & 0 & & & \\ & \mathbf{1} & & 0 & & \\ & \mathbf{1} & & & 0 & \\ & \mathbf{1} & & & & 0 \end{bmatrix} \Rightarrow M_{22} \bullet Y = 0$$

$$M_{32} = \begin{bmatrix} 0 & & & & & 0 \\ & 0 & & & & \\ & & \mathbf{-1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ & & \mathbf{1} & 0 & & \\ & & \mathbf{1} & & 0 & \\ 0 & & \mathbf{1} & & & 0 \end{bmatrix} \Rightarrow M_{32} \bullet Y = 0$$

I vincoli IMC_A e IMC_B impediscono che si possano considerare contributi derivanti da lati presi fra elementi della stessa classe V_k.

Sommando le matrici M_{ik} associate ai vincoli IMC_A otteniamo un vincolo unico, caratterizzato da una sola matrice M_C

$$M_C = \begin{bmatrix} 0 & \mathbf{1} & \mathbf{1} & & & \\ \mathbf{1} & 0 & \mathbf{1} & & 0 & \\ \mathbf{1} & \mathbf{1} & 0 & & & \\ & & & 0 & \mathbf{1} & \mathbf{1} \\ & 0 & & \mathbf{1} & 0 & \mathbf{1} \\ & & & \mathbf{1} & \mathbf{1} & 0 \end{bmatrix} \Rightarrow M_c \bullet Y = 0$$

Il vincolo IMC_C così ottenuto è un surrogato di IMC_A. Come risulta più evidente nella matrice M_c, si osservano elementi pari a 1 nelle posizioni in cui si vogliono proibire contributi non nulli nella matrice soluzione Y.

4.3.3 Vincoli di taglio

Analogamente ai vincoli di taglio 2.11 definiti per la formulazione lineare, possiamo ottenere una matrice che impone tali vincoli per il modello semidefinito. Surrogando la famiglia di vincoli

$$\sum_{j \in V} x_{ij} = m x_{ii} \quad \forall i \in V \tag{4.5}$$

otteniamo il vincolo $4C''$

$$\sum_i \sum_j x_{ij} = m \sum_i x_{ii}$$

Rappresentiamo di seguito la matrice associata al vincolo di taglio appena definito sul problema di esempio 2.4:

$$M_{4C''} = \begin{bmatrix} 1-m & & & & 1 & 1 & 1 \\ & 1-m & & & 1 & 1 & 1 \\ & & 1-m & & 1 & 1 & 1 \\ 1 & 1 & 1 & 1-m & & & \\ 1 & 1 & 1 & & 1-m & & \\ 1 & 1 & 1 & & & 1-m & \end{bmatrix}$$

Possiamo scrivere equivalentemente il vincolo $4C''$ come:

$$M_{4C''} \bullet Y = 0$$

Se $i \in V$ è scelto nella classe $V_k \in K$ allora devono essere scelti $(m-1)$ lati incidenti ad i aventi come estremi dei vertici di classi diverse.

Scomponendo il vincolo $4C''$ si possono ottenere n vincoli considerando la relazione 4.5. Per ogni vertice $i \in V$ possiamo definire un vincolo $4C''_i$ e una matrice $M_{4C''_i}$ tali che:

$$M_{4C''_i} \bullet Y = 0 \quad \forall i \in V \quad Vincolo\, 4C''_i$$

Appoggiandoci all'esempio 2.4 riportiamo le matrici $M_{4C_i''}$ con $i \in V_1$ ossia $i = \{1, 2, 3\}$:

$$M_{4C_1''} = \begin{bmatrix} \mathbf{1-m} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & & & & & \\ 0 & & & & & \\ \frac{1}{2} & & & & & \\ \frac{1}{2} & & & & & \\ \frac{1}{2} & & & & & \end{bmatrix} \quad i = 1 \in V_1 \Rightarrow M_{4C_1''} \bullet Y = 0$$

$$M_{4C_2''} = \begin{bmatrix} & 0 & & & & \\ 0 & \mathbf{1-m} & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ & 0 & & & & \\ & \frac{1}{2} & & & & \\ & \frac{1}{2} & & & & \\ & \frac{1}{2} & & & & \end{bmatrix} \quad i = 2 \in V_1 \Rightarrow M_{4C_2''} \bullet Y = 0$$

$$M_{4C_3''} = \begin{bmatrix} & & 0 & & & \\ & & 0 & & & \\ 0 & 0 & \mathbf{1-m} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ & & \frac{1}{2} & & & \\ & & \frac{1}{2} & & & \\ & & \frac{1}{2} & & & \end{bmatrix} \quad i = 3 \in V_1 \Rightarrow M_{4C_3''} \bullet Y = 0$$

4.3.4 Vincoli di taglio con rafforzamento diagonale

Dai precedenti vincoli di taglio, possiamo ottenere delle relazioni che ci permettono di rafforzare la rappresentazione diagonale di Y, imponendo un vincolo al numero di lati incidenti ad un vertice scelto e la mutua esclusione dei vertici, appartenenti alla medesima classe.

Consideriamo la relazione

$$\sum_j (x_{jj} - x_{ij}) = m(1 - x_{ii}) \quad \forall i \in V \tag{4.6}$$

Per verificare la validità di questa relazione distinguiamo due casi: $x_{ii} = \begin{cases} 1 \\ 0 \end{cases}$

Da $x_{ii} = 1$ segue $\sum_{j \in V} x_{jj} = \sum_{j \in V} x_{ij}$ ovvero deve essere selezionato un lato tra i ed ogni altro vertice j selezionato.

Se invece, $x_{ii} = 0$, allora $x_{ij} = 0$, $\forall j \in V$ e $\sum_{j \in V, j \neq i} x_{jj} = m$ ossia devo selezionare altri m vertici.

Dalla relazione 4.6 si ricavano n vincoli, uno per ogni vertice $i \in V$. Si definisce quindi la matrice $M_{4C_i'''}$, associata al vertice i, tale che

$$M_{4C_i'''} \bullet Y = m \quad \forall i \in V$$

Riportiamo di seguito alcune delle matrici $M_{4C_i'''}$ con $i = \{1, 2, 3\}$ riferite all'esempio 2.4.

$$M_{4C_1'''} = \begin{bmatrix} \mathbf{m} & & & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} \\ & 1 & & & & \\ & & 1 & & & \\ -\frac{\mathbf{1}}{\mathbf{2}} & & & 1 & & \\ -\frac{\mathbf{1}}{\mathbf{2}} & & & & 1 & \\ -\frac{\mathbf{1}}{\mathbf{2}} & & & & & 1 \end{bmatrix}$$

$$M_{4C_2'''} = \begin{bmatrix} 1 & & & & & \\ & \mathbf{m} & & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} \\ & & 1 & & & \\ & -\frac{\mathbf{1}}{\mathbf{2}} & & 1 & & \\ & -\frac{\mathbf{1}}{\mathbf{2}} & & & 1 & \\ & -\frac{\mathbf{1}}{\mathbf{2}} & & & & 1 \end{bmatrix}$$

$$M_{4C_3'''} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \mathbf{m} & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} \\ & & -\frac{\mathbf{1}}{\mathbf{2}} & 1 & & \\ & & -\frac{\mathbf{1}}{\mathbf{2}} & & 1 & \\ & & -\frac{\mathbf{1}}{\mathbf{2}} & & & 1 \end{bmatrix}$$

Il modello semidefinito proposto non può essere utilizzato direttamente per ottenere la soluzione ottima a causa del vincolo di rango (2.13). Fortunatamente rilassando tale vincolo, è possibile ottenere un bound duale valido per il problema, utilizzando un solutore general purpose per la programmazione semidefinita.

Capitolo 5

Branch and Bound

I computer sanno contare solo da 0 ad 1.
Il resto è illusione.
– anonimo

Il metodo del Branch and Bound (B&B) è stato proposto da *A. H. Land e A. G. Doig* nel 1960 per e risolvere problemi $NP-difficili$ di ottimizzazione combinatoria.

Consiste nell'esplorazione di un albero di ricerca (5.1), i cui nodi rappresentano i sottoproblemi di un problema P_0 di partenza. Ogni sottoproblema è radice di un nuovo albero; ai suoi figli corrispondono F sottoinsiemi $(P_1, ..., P_F | \bigcup_{k=1,...,F} P_k = P_0)$ dell'insieme di soluzioni S_k. È desiderabile ai fini dell'efficienza che i sottoinsiemi siano disgiunti ovvero $F(P_i) \bigcap F(P_j) = \emptyset \ \ \forall (P_i, P_j) : i \neq j$. Le foglie dell'albero di branching rappresentano tutte le possibili soluzioni del problema.

L'esplorazione dell'albero è implicita: per ogni sottoproblema sono calcolati dei bound che forniscono forniscono stime per eccesso (*upper bound*) e per difetto (l*ower bound*) del valore della soluzione ottima.

Ogni sotto problema P_i si può considerare risolto se almeno una delle seguenti condizioni è verificata:

- Si trova la soluzione ottima di P_i.
- Si dimostra che $F(P_i)$ non ammette soluzioni valide.
- Si dimostra che l'upper bound del sottoproblema P_i è non superiore del valore della migliore soluzione conosciuta.

Nel caso del nostro algoritmo un *upper bound* è calcolato tramite le tecniche di rilassamento discusse nel capitolo (4), e dei *lower bound* sono calcolati utilizzando le euristiche illustrate nel capitolo (3) e la migliore soluzione conosciuta è quella fornita dai migliori lower bound calcolati.

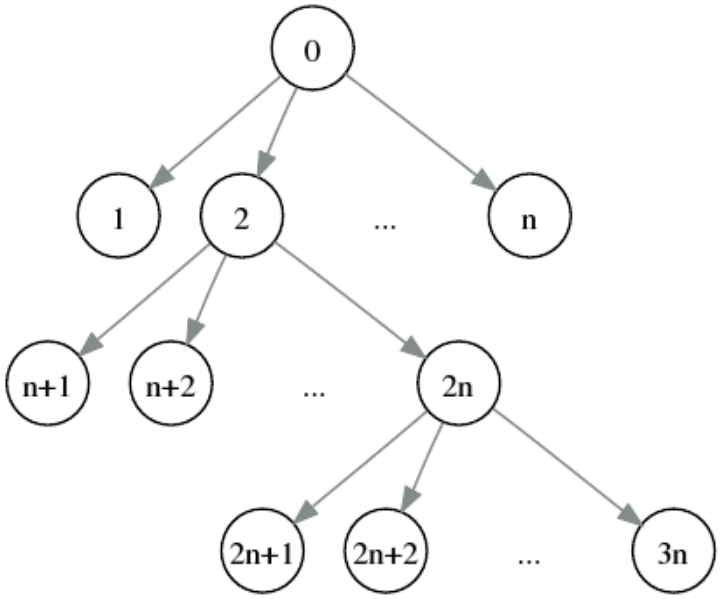

Figura 5.1: Esempio di albero di esplorazione di un algoritmo Branch and Bound

Le prestazioni di un algoritmo di B&B dipendono da molti fattori fra cui il modo in cui ogni sottoproblema è partizionato, la politica di esplorazione dell'albero, le tecniche di bounding, eventuali tecniche di riduzione del problema.

La politica di branching ha il compito di decidere il partizionamento dei nodi dell'albero, riducendo la dimensione dei sottoproblemi generati. La nostra tecnica di branching è descritta nella sottosezione 5.3 e prevede ad ogni passo una divisione binaria bilanciata dei nodi di una classe scelta.

Fra le possibili tecniche di esplorazione dell'albero di ricerca, sono frequentemente utilizzate la visita in profondità (depth-first), in grado di fornire rapidamente delle soluzioni primali e la tecnica di best-first in cui viene data precedenza ai sottoproblemi che presentano un bound duale più promettente. Questa tecnica è stata scelta come politica di esplorazione per il nostro algoritmo esatto.

Ci sono anche altre tecniche che permettono di aumentare le prestazioni di un algoritmo B&B, ad esempio nel capitolo (5.1) abbiamo mostrato come è stato possibile ridurre la dimensione del problema iniziale mediante un algoritmo di preprocessing. Inoltre, un sottoproblema ritenuto sufficientemente piccolo, ricorriamo a tecniche di enumerazione esplicita come descritto nel capitolo (5.5).

5.1 Preprocessing

Supponiamo di imporre che, in una classe V_k, il vertice i appartenga alla clique M. Il problema si riduce alla scelta di altri $|M|-1$ vertici da altre classi. Inoltre scegliendo dalla classe V_h il vertice j è necessario pagare un contributo fisso

pari a $\hat{W}_j = w_V(j) + W_{ij}$. Sia $\hat{z}_i$ il valore del bound combinatorio calcolato sul sottoproblema in cui la classe V_k è esclusa e sono considerati i pesi modificati $\hat{W}$. $\hat{z}_i$ è un upper bound al valore della soluzione ottima quando si impone che i appartenga alla clique M.

Sia $\bar{z}$ un lower bound al valore della soluzione ottima: se $\hat{z}_i + w_V(i) \leq \bar{z}$ allora non è possibile ottenere soluzioni di valore superiore a $\bar{z}$ imponendo che i appartenga alla clique M. Pertanto il vertice i può essere escluso dal problema.
Questo calcolo può essere ripetuto per ogni vertice del grafo. Successivamente possiamo ricavare il lower bound $\bar{z}$ sia mediante l'euristica combinatoria descritta in (3.2) che utilizzando l'algoritmo di Tabu Search presentato in (3.3).

Una volta ottenuti i valori $\hat{z}_i$ ed i lower bound $\bar{z}$ procediamo alla verifica delle condizioni appena illustrate ed all'eventuale rimozione di vertici del problema.

5.2 Strategia di ricerca

La strategia di ricerca nell'albero delle decisioni ha un contributo decisivo nel numero di nodi da esplorare con un impatto diretto sul tempo totale di calcolo e l'utilizzo di memoria per enumerare tutte le soluzione.

Il nostro algoritmo sfrutta una politica di esplorazione *best first* dell'albero di branching, nella quale si dà precedenza alla valutazione di sottoproblemi dal bound duale più promettente. Si affida all'euristica primale il compito di raggiungere soluzioni ammissibili partendo da soluzioni parziali probabilmente vicine all'ottimo, descritte dai nodi più promettenti.

5.3 Strategia di Branching

Inizialmente il problema non ha alcuna variabile vincolata, il nodo radice contiene tutte le soluzioni possibili. L'approccio Branch and Bound richiede di dividere il problema di partenza in sottoproblemi più facili da risolvere. Questo importante compito è assegnato alla strategia di branchinig, dalla quale spesso dipende il segreto dell'efficienza dell'algoritmo di enumerazione. In questo lavoro sono state provate alcune strategie, sfruttando le proprietà del problema e tenendo conto delle possibilità messe a disposizione dalle tecniche di rilassamento. Una lunga campagna di test ha permesso di individuare la più efficiente da cui partire con perfezionamenti successivi.

5.3.1 Branching binario bilanciato

Questa tecnica sfrutta i valori frazionari delle variabili decisionali presenti sulla diagonale della matrice Y ottenuta con il rilassamento del modello semidefinito. I vincoli di Multiple Choice (2.14) impongono nel rilassamento che le variabili frazionarie associate ai vertici appartenenti alla stessa classe sommino a 1.

Ovvero, se $x = Diag\{Y\}$ è il vettore che rappresenta la diagonale della matrice Y

$$\sum_{i \in V_k} x_i = 1 \quad \forall k \in K$$

Senza perdita di generalità, supponiamo che elementi di una stessa classe risiedono in posizioni consecutive del vettore x. Ad esempio il vettore x associato ai vertici di un'istanza formata da 6 vertici e partizionata in 3 classi (V_1, V_2, V_3) avrà la seguente forma: $x^T = [x_1, x_2, x_3, x_4, x_5, x_6]$ con $\{x_1, x_2\} \in V_1$, $\{x_3, x_4\} \in V_2$, $\{x_5, x_6\} \in V_3$.

L'idea del branching binario bilanciato consiste nel generare due figli partizionando una classe in due sottoinsiemi. Il primo figlio si ottiene fissando le variabili nella prima sottoclasse a 1 mentre il secondo figlio si ottiene fissando a 0 la seconda metà.

La classe su cui eseguire il branching è individuata fra quelle con il maggior numero di variabili frazionarie poiché la qualità del rilassamento dipende in gran parte da questo valore. A questo punto si tratta di discernere quale classe considerare, fra tutte quelle individuate. Rappresentiamo con

$$\mathcal{F}_{ki} = |\sum_{j \leq i \in V_k} x_j - \frac{1}{2}|$$

il valore che indica nella classe V_k, di quanto la somma in valore assoluto delle variabili frazionarie x_j con $j \leq i$ si scosta dal valore $\frac{1}{2}$ e

$$i_k^* = \arg\min_{i \in V_k}\{\mathcal{F}_{ki}\}$$

è la variabile frazionaria in V_k che ha il minimo scostamento dal valore $\frac{1}{2}$. Tale variabile viene utilizzata per identificare la separazione fra le variabili che dovranno essere fissate.

$$k^* = \arg\min_{k \in K}\{\mathcal{F}_k\}$$

dove

$$\mathcal{F}_k = \min_{i \in V_k}\{\mathcal{F}_{ki}\}$$

La classe k^* viene eletta come più bilanciata e l'elemento i_k^* è l'elemento separatore fra le variabili che dovranno essere fissate nel primo e nel secondo figlio. $\mathcal{F}_k$ indica il valore minimo di scostamento dal valore $\frac{1}{2}$ associato dalla classe V_k .

5.4 Bounding

Questa fase dell'algoritmo di enumerazione consiste nel valutare un determinato sottoproblema. Da questo problema ricaviamo un bound duale, ovvero una stima per eccesso del valore della migliore soluzione applicando il rilassamento

combinatorio e semidefinito. Inoltre possiamo conoscere una soluzione primale sfruttando gli algoritmi euristici presentati nel capitolo 3.
Per quanto riguarda il calcolo del bound duale si procede dapprima con il calcolo del rilassamento combinatorio per poi eseguire il rilassamento semidefinito. Queste due valutazioni sono indipendenti e forniscono due diversi bound duali. Il valore del bound più stretto viene associato al sottoproblema in esame. Infatti, dato che entrambe le valutazioni sono corrette, la tecnica che restituisce un valore più vicino all'ottimo è da preferirsi perché permette di stringere maggiormente il gap residuo dal valore ottimo.

La necessità di utilizzare a cascata due rilassamenti è sorta per fronteggiare alcune tipologie di istanze. In particolare, il rilassamento semidefinito ha mostrato particolare efficienza sulle istanze con molte classi di piccole dimensioni, al contrario, si è visto che al crescere della cardinalità delle classi della partizione, il valore del bound semidefinito diventa sensibilmente più scarso e richiede l'ausilio del rilassamento combinatorio. Nonostante il calcolo del bound combinatorio sia molto meno gravoso in termini computazionali di quello semidefinito, abbiamo pensato ad una tecnica in grado di attivare selettivamente la valutazione del primo bound duale. La fase di enumerazione parte con entrambe le tecniche di rilassamento attivate, ma il contributo del bound combinatorio viene disattivato non appena il vantaggio diventa nullo. Da quel livello dell'albero in poi, l'operazione di branching provvede ad impedire l'utilizzo del rilassamento combinatorio nei nodi figli creati.

Un ulteriore aumento delle prestazioni è stato ottenuto modificando i parametri dell'algoritmo del cammino centrale e sfruttando le peculiarità dell'approccio basato sul metodo del punto interno. A tale proposito abbiamo sfruttato due possibilità: la prima, già illustrata in (4.2) consiste nel fornire all'algoritmo un punto di partenza, rappresentato dai valori ottimi delle variabili duali associate al problema del nodo padre; la seconda ottimizzazione impone di terminare le iterazioni dell'algoritmo, prima del raggiungimento della convergenza, se il valore duale scende sotto la soglia rappresentata del valore della migliore soluzione primale conosciuta (*early branching*). Infatti se si verifica questa condizione possiamo affermare che il problema in questione non può contenere la soluzione ottima e quindi possiamo astenerci da ulteriori valutazioni.

Per quanto riguarda i bound primali, entrambe le tecniche di rilassamento, combinatorio e semidefinito permettono di ricavare rapidamente una soluzione ammissibile. Nel primo caso fruttiamo un'euristica basata sulle proprietà combinatorie del problema (3.2) mentre, per il rilassamento semidefinito utilizziamo la tecnica del rounding illustrata nel capitolo 3.1.

5.5 Enumerazione esplicita

Il rilassamento semidefinito così come quello combinatorio sono gli strumenti che ci permettono di avere una stima per eccesso del valore della soluzione ottima del problema. Questo approccio è molto potente e permette di affrontare in modo "semplice" problemi anche molto complessi. Tuttavia il rilassamento semidefinito richiede per sua natura uno sforzo computazionale non indifferente nel valutare i nodi dell'albero di decisione. L'idea dell'enumerazione esplicita suggerisce di risolvere il sottoproblema in esame enumerando esplicitamente tutte le soluzioni. Ovviamente questo approccio si rende possibile solo dopo che il problema originario, difficile da risolvere, è stato opportunamente ridotto. Una volta enumerato esplicitamente un nodo dell'albero di ricerca, tale nodo può essere chiuso definitivamente (caso 1, nella descrizione del B&B). Le soluzioni enumerate sono tutte ammissibili e il loro valore viene utilizzato come bound primale per il problema di partenza.

Le soluzioni contenute in un sottoproblema crescono esponenzialmente col numero di vertici rimanenti ma, sono ovviamente in numero finito. Per stabilire quanto difficile sia il problema da risolvere calcoliamo l'ordine di grandezza del corrispondente numero di soluzioni.

Dato un sottoproblema P_i con $k_i = |K_i|$ classi indichiamo con ℓ_k il numero di variabili "libere" nella classeV_k. Si verifica facilmente che il numero di soluzioni ammissibili è pari a

$$S(P_i) = \prod_{k \in K_i} \ell_k$$

La tecnica di enumerazione esplicita è sicuramente vantaggiosa se il numero di operazioni richieste è minore o uguale al numero di operazioni richieste per risolvere un rilassamento semidefinito.

Sperimentalmente abbiamo determinato che l'enumerazione esplicita è vantaggiosa se il numero di soluzioni da enumerare è dell'ordine di 10^5.

Capitolo 6

Analisi sperimentale

Nell'industria dei computer, ci sono tre tipi di bugie:
le menzogne, le menzogne spudorate e i benchmark
– Dal Jargon file, r. 2.1.5

In questo capitolo vengono analizzati i risultati dei test condotti sia su benchmark presenti in letteratura ([6]) che su benchmark da noi proposti al fine di valutare sia la qualità delle singole componenti che le prestazioni generali del nostro algoritmo.

Nella prima parte vengono mostrati dei test mirati, in grado di evidenziare nel dettaglio i contributi delle componenti del nostro algoritmo. Nella seconda parte viene mostrato il confronto della versione completa del nostro algoritmo, con il il solutore commerciale *ILOG Cplex* e, ove possibile, con un algoritmo esatto basato sul rilassamento Lagrangeano presentato da Bosio in ([6])

6.1 Organizzazione degli esperimenti

Gli algoritmi sono stati implementati in standard *C* (1999), compilati con *gcc 4.3.3* ed eseguiti su un PC x86 con processore Intel$^{(R)}$ Core^{TM} 2 Duo (CPU E6850) a 3.0GHz con 2GB di RAM, in ambiente GNU Linux^{TM} versione 2.6.28, compilato a 32bit.

L'algoritmo implementato non è parallelo pertanto non vengono sfruttati contemporaneamente i due *core* del processore.

Il benchmark da noi proposto ($S1$) consiste di 165 istanze, divise in 3 gruppi A, B, C.

I pesi W_{ij} associati ai vertici e ai lati in ogni gruppo sono stati scelti come segue:

- Tipo A: si definiscono n elementi e_i, $i = 1, .., n$ ognuno dei quali è rappresentato da un punto in uno spazio a 3 dimensioni con coordinate intere uniformi nell'intervallo $[1, 100]$. Per ogni coppia di elementi (e_i, e_j) viene associato un peso w_{ij}calcolato come $int(||e_i - e_j||_2)$. Ad ogni vertice

$e_i \in V$ viene inoltre associato in peso $\tilde{w}_i$ estraendo un numero intero uniforme nell'intervallo $[1, 10]$.

- Tipo B: I pesi w_{ij} associati ai lati del grafo sono estratti da una distribuzione discreta uniforme nell'intervallo $[1, 100]$ mentre i pesi $\tilde{w}_i$ associati ai vertici sono estratti da una distribuzione discreta uniforme in $[1, 10]$.

- Tipo C: analogamente alle istanze di tipo B, i pesi w_{ij} sono estratti da una distribuzione discreta uniforme nell'intervallo $[-50, 50]$ mentre i pesi associati ai vertici sono stati ottenuti da una distribuzione discreta uniforme tra $[-7, 7]$.

Ogni gruppo è composto da 55 istanze con un numero di nodi compreso tra 30 e 300, un numero di classi tra 3 e 150 e un numero di elementi per classe che varia da 2 fino ad un massimo di 100.

Il nome di queste istanze è rappresentato nella forma a_n100c2, dove la prima lettera indica il gruppo di appartenenza ($\{a = A, b = B, c = C\}$), $n100$ indica il numero di vertici (in questo caso pari a 100), infine $c2$ rappresenta il numero di vertici in ogni classe (in questo caso $c = 2$).

Abbiamo considerato anche tre gruppi di istanze D, E, F costruiti rispettivamente come A, B, C, ma in cui il peso $w_V(i)$ di ogni vertice è stato fissato a 0. Dato che i risultati non differiscono in modo significativo da A, B, C omettiamo la loro presentazione.

Il benchmark proposto in [6], identificato con $(S2)$ è composto da 168 istanze e diviso in 4 classi: $\{B1, B2, B3, B4\}$. Analogamente al benchmark precedente, la partizione dei vertici è costituita da classi con la stessa cardinalità. Per determinare i pesi associati rispettivamente agli spigoli w_{ij} ed ai vertici $\tilde{w}_i$ è stata utilizzata la seguente formula

- Tipo B1: i lati dono distribuiti secondo la relazione $1 \leq w_{ij} \leq 1000r^h$ $(1 \leq w_{ij} \leq 1000)$ con $h \in \{1, ..., 5\}$, $r \in (0, 1)$ e $1 \leq \tilde{w}_i \leq 1000$ per i pesi associati ai vertici.

- Tipo B2: è analogo al tipo B1 ma senza pesi associati ai vertici, $\tilde{w}_i = 0 \, \forall i$.

- Tipo B3: questa classe ha la caratteristica di comprendere anche pesi negativi, sia sui lati che sui vertici. I pesi w_{ij} associati ai lati sono generati secondo la relazione:$-1000 \leq w_{ij} \leq 1000r^h$ $(-1000 \leq w_{ij} \leq 1000)$ con $h \in \{1, ..., 5\}$, $r \in (0, 1)$, mentre i pesi $\tilde{w}_i$ associati ai vertici sono generati utilizzando una distribuzione uniforme nell'intervallo $[-1000, 1000]$.

- Tipo B4: è analogo a B3 ma con $\tilde{w}_i = 0 \, \forall i$.

Il nome dell'istanza riflette le sue caratteristiche. Ad esempio, $1 - 6 - 2$ indica in ordine da sinistra: il tipo $(B1)$, il numero di vertici del grafo $(n = 6)$ e la cardinalità di ogni classe della partizione $(c = 2)$.

L'algoritmo esatto basato sull'enumerazione implicita è stato testato su entrambi i benchmark ($S1$, $S2$) e sono stati evidenziati i risultati del confronto con il solutore commerciale ILOG Cplex 11.2 e con l'algoritmo di Branch and Bound proposto da Bosio (solo sulle istanze benchmark $S2$).

Tutti gli esperimenti computazionali sono stati eseguiti in un tempo limite di 1 ora. Qualora la simulazione non sia terminata viene riportato il *gap* in percentuale dal valore della migliore soluzione primale. Indicheremo quindi con *Best Known* il valore della migliore soluzione primale conosciuta.

Nella prima parte mostriamo i risultati al nodo radice dell'albero delle decisioni, focalizzando l'attenzione sul tempo di calcolo e sui valori dei bound trovati sia con le tecniche di rilassamento che con le euristiche primali. Di seguito abbiamo riportato i confronti dei tempi di calcolo dell'algoritmo di enumerazione $SDB_{B\&B}$ con il solutore Cplex. Per questo motivo abbiamo fornito a Cplex il modello di programmazione lineare intera (2.2) con l'aggiunta dei vincoli di taglio (2.11). Cplex è in grado di trovare la soluzione ottima mediante un algoritmo basato sul Branch and Bound. La tecnica principale impiegata dal solutore per determinare un bound duale è rappresentata dal rilassamento continuo ottenuto eliminando i vincoli di integralità sulle variabili decisionali (2.10). Le simulazioni sono state ottenute lasciando i parametri di default e imponendo un tempo limite di 1 ora. In appendice (B) è mostrato come è stato codificato il modello in linguaggio AMPL.

Nella seconda sezione degli esperimenti abbiamo spostato l'attenzione verso l'analisi delle singole componenti dell'algoritmo esatto. In particolare sono stati condotti dei test in grado di evidenziare il ruolo del preprocessing, del rilassamento combinatorio e della tecnica di enumerazione esplicita. Attivando e disattivando selettivamente dalla versione dell'algoritmo completa i contributi delle varie tecniche, abbiamo potuto mostrare più chiaramente, i contributi dovuti ai vari algoritmi implementati.

Per poter mostrare i risultati della vastissima campagna di simulazioni abbiamo dovuto aggregare i risultati relativi a diverse istanze. Tutti i valori delle tabelle rappresentano quindi i valori medi. Come primo criterio abbiamo scelto di raggruppare le istanze per dimensione n, successivamente abbiamo ulteriormente aggregato le istanze per numero di classi m. Nel caso del benchmark $S2$ abbiamo scelto come secondo criterio di aggregazione la cardinalità della classe (c) al posto di del numero di classi (m).

Per il benchmark $S1$ abbiamo applicato il seguente schema per tutti i tipi di istanze:

$$
\begin{array}{ll}
n & m \\
piccole: [30-80] & \begin{cases} [3-7] \\ [8-14] \\ 15-40 \end{cases} \\
medie: [100-150] & \begin{cases} [3-5] \\ [10-24] \\ [30-75] \end{cases} \\
grandi: [200-300] & \begin{cases} [3-5] \\ [10-40] \\ [58-150] \end{cases}
\end{array}
$$

Per il benchmark $S2$ abbiamo aggregato le istanze secondo lo schema seguente:

$$
\begin{array}{ll}
n & c \\
piccole: [30-33] & \begin{cases} 2 \\ 3 \\ 4 \\ 5 \end{cases} \\
medie: [45-80] & \begin{cases} 2 \\ 3 \\ 4 \\ 5 \end{cases}
\end{array}
$$

6.2 Bound primali

La prima fase dei test considera i bound primali ottenuti al nodo radice dell'albero delle decisioni.

In questa sezione vengono mostrati i confronti tra i risultati ottenuti con la versione dell'algoritmo completa $SDP_{B\&B}$ e il solutore *Cplex*.

Nelle tabelle di sintesi indichiamo con $gap\,PB$ il gap medio, espresso in valori percentuali tra il bound primale ottenuto dall'euristica (PB) e il $Best\,Known$ (BK) calcolato come

$$gap = \frac{BK - PB}{BK}$$

Il $gap\,PB_{comb}$ indica il gap medio, espresso in valori percentuali tra il valore del bound primale combinatorio (sezione 3.2) e il $Best\,Known$. Analogamente indichiamo con $gap\,PB_{sem}$, $gap\,PB_{TS}$, $gap\,PB_{cplex}$ il gap del bound primale rispetto al $Best\,Known$, rispettivamente ottenuto col rounding (sezione 3.1), mediante l'euristica di TS e calcolato da Cplex a cui è stato fornito il modello di PLI con i vincoli di taglio, al termine dell'elaborazione del nodo radice.

Il termine $\Delta_{PB_{cplex}-PB_{TS}}$ indica la differenza tra $gap\,PB_{cplex}$ e $gap\,PB_{TS}$.

Le prime due tabelle (6.1, 6.2) associate rispettivamente al benchmark $S1$ e $S2$ mostrano l'efficacia dei bound primali, con i relativi gap dal $Best\,Known$. Si nota immediatamente che la colonna riferita al Tabu Search ha i gap più bassi degli altri algoritmi euristici, (salvo alcune eccezioni marcate in grassetto, nel benchmark $S1$) sui valori medi. Nel benchmark $S1$ si contano 146 istanze su 165 (88.48%) in cui il Tabu Search trova il $Best\,Known$. Nel benchmark $S2$ invece abbiamo verificato che 134 volte su 168, pari al 79.76%, il TS ha trovato la soluzione ottima. Nelle rimanenti istanze il Tabu Search trova una soluzione primale molto vicina all'ottimo; il gap medio (gap PB_{TS}) mostra valori sempre minori del 2%.

Dato che il nostro algoritmo prevede l'utilizzo del TS iniziale, abbiamo scelto tale bound primale per il confronto con Cplex. La colonna $\Delta_{PB_{cplex}-PB_{TS}}$ riporta infatti la differenza media del gap in percentuale tra PB_{cplex} e PB_{TS}. Entrambe le tabelle mostrano che la nostra euristica è molto più efficiente di *Cplex* salvo in alcuni casi isolati, evidenziati in grassetto nel benchmark $S2$. La differenza riscontrata è tuttavia molto contenuta, sempre minore del 0.5%.

La tabella 6.1, riferita al benchmark $S1$ mostra per il tipo di istanze A dei bound primali molto stretti, inferiori al 10% per tutte le euristiche primali. Come sarà più chiaro nel corso dell'illustrazione degli altri risultati, questo tipo di istanze risultano particolarmente facili; per questo motivo, sia il rounding della soluzione semidefinita che l'euristica combinatoria permettono di determinare delle soluzioni primali molto vicine all'ottimo. L'osservazione può essere estesa al comportamento riscontrato per il bound primale determinato da Cplex.

Per quanto riguarda la classe B, si nota che il PB_{TS} continua a mantenere un gap molto stretto a differenza degli altri bound primali (PB_{sem} e PB_{comb}) che mostrano gap più elevati (fino al 43%) per le istanze con un numero limitato di classi.

Si osserva nell'ultima classe C di istanze del benchmark $S1$ che i gap (PB_{comb}, PB_{sem}) aumentano fino al 90%; il rounding mantiene l'andamento osservato per la classe B pur avendo gap più contenuti rispetto all'euristica combinatoria. L'incremento del gap è osservabile anche per Cplex il quale perde in alcuni casi (in grassetto) più del 50%.

La tabella 6.2 mostra i bound primali al nodo radice per il benchmark $S2$. Per tutti i tipi di istanze ($B1$, $B2$, $B3$, $B4$), gli algoritmi euristici mostrano un comportamento simile. Il gap medio del rounding (PB_{sem}) si mantiene sempre contenuto tra 0% e 9.22% mentre l'euristica combinatoria mostra dei gap molto maggiori, compresi tra 3.52% e 67.25%. Il gap del Tabu search si mantiene sempre molto contenuto, al di sotto del 1.7%. Per questo benchmark anche Cplex mostra una particolare efficienza nel calcolo del bound primale, infatti oltre ad avere gap abbastanza contenuti ($0 - 18\%$), si evidenziano in

grassetto dei casi persino migliori del Tabu Search, seppur per pochi decimi percentuali.

6.3 Bound duali

L'analisi dei bound duali al nodo radice rappresenta un passo fondamentale nella comprensione di un algoritmo di tipo Branch and Bound. Infatti da questa informazione è possibile capire se le tecniche di rilassamento utilizzate sono efficaci oppure no. Intuitivamente, se la distanza del bound duale è molto vicina al valore ottimo, possiamo aspettarci che una politica di branching ben studiata possa enumerare in fretta tutte le soluzioni. Al contrario, se il gap associato al bound duale è elevato significa che il bound duale rappresenta una stima molto lasca del valore della soluzione ottima e quindi ci possiamo aspettare, che l'algoritmo di enumerazione richieda più tempo.

Nonostante il calcolo del rilassamento semidefinito avvenga attraverso un algoritmo polinomiale, al crescere della dimensione del problema da valutare, il carico computazionale cresce in modo sensibile. Per questo motivo abbiamo prestato particolare attenzione ai vari aspetti che possono influenzare le prestazioni dell'algoritmo del cammino centrale.

Abbiamo verificato che alcune combinazioni di vincoli permettono di ottenere dei bound migliori di altre; a valori più stringenti del rilassamento corrispondono di solito tempi di calcolo maggiori. Per ottenere i risultati mostrati in seguito è stato necessario trovare la combinazione di vincoli che garantisse il corretto compromesso tra la qualità del bound trovato e il tempo di calcolo necessario.

In particolare, nella sezione (4.2.1) abbiamo mostrato come ottenere un rilassamento dal programma semidefinito (2.3).

Fra i più recenti solutori in grado di risolvere efficientemente problemi di programmazione semidefinita abbiamo scelto *DSDP5* [4] sviluppato da Benson e Ye con il supporto delle divisioni di matematica e scienze computazionali del dipartimento per l'energia degli Stati Uniti. Il DSDP 5 implementa un particolare algoritmo basato sul metodo del punto interno chiamato *dual-scaling algorithm* ed è disponibile come libreria sia per MATLAB che per C, C++.

Il solutore scelto è stato realizzato per poter risolvere un programma semidefinito generale; per migliorare le prestazioni è necessario poter modificare alcuni parametri che regolano il comportamento interno dell'algoritmo di dual-scaling. Infatti, come mostrato dagli autori stessi ([3]), la scelta corretta dei valori di alcuni parametri può influenzare significativamente le prestazioni. A tal proposito il solutore DSDP5 presenta notevole elasticità e ha permesso, in modo relativamente semplice di adattare il comportamento del solutore al nostro problema specifico.

Dato che la convergenza dell'algoritmo del cammino centrale dipende dalla prossimità del punto iniziale dal cammino centrale, abbiamo fornito come punto di partenza i valori delle variabili y_i del problema duale. Ovvero, nella

$S1(A)$		*root gap* %				
n	m	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-80]$	$[3-7]$	4.17	0.63	0.00	1.46	1.46
	$[8-14]$	3.51	0.02	0.00	3.34	3.34
	$[15-40]$	0.42	0.01	0.00	0.28	0.28
$[100-150]$	$[3-5]$	10.15	1.00	0.04	2.58	2.55
	$[10-24]$	2.85	0.05	0.00	2.13	2.13
	$[30-75]$	0.16	0.00	0.00	3.78	3.78
$[200-300]$	$[3-5]$	8.76	2.81	0.00	6.32	6.32
	$[10-40]$	2.35	0.05	0.00	14.07	14.07
	$[48-150]$	0.07	0.00	0.00	7.49	7.49

$S1(B)$		*root gap* %				
n	m	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-80]$	$[3-7]$	30.49	18.48	0.03	9.54	9.51
	$[8-14]$	14.24	5.72	0.02	6.35	6.32
	$[15-40]$	3.08	0.91	0.00	1.47	1.47
$[100-150]$	$[3-5]$	37.64	31.93	0.34	16.68	16.33
	$[10-24]$	19.28	13.35	0.06	10.82	10.76
	$[30-75]$	3.44	0.64	0.00	3.63	3.63
$[200-300]$	$[3-5]$	37.02	43.99	0.79	9.00	8.21
	$[10-40]$	18.65	11.75	0.00	13.49	13.49
	$[48-150]$	3.08	0.73	0.00	4.56	4.56

$S1(C)$		*root gap* %				
n	m	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-80]$	$[3-7]$	73.60	42.09	0.50	10.58	10.08
	$[8-14]$	77.57	35.78	0.00	6.77	6.77
	$[15-40]$	33.84	9.48	0.11	3.42	3.31
$[100-150]$	$[3-5]$	71.25	84.82	0.47	36.93	36.45
	$[10-24]$	68.68	44.67	0.04	27.72	27.68
	$[30-75]$	40.39	6.32	0.11	36.25	36.14
$[200-300]$	$[3-5]$	88.59	76.46	1.16	19.48	18.32
	$[10-40]$	82.80	55.63	0.00	63.14	63.14
	$[48-150]$	49.63	11.38	0.25	**72.62**	72.38

Tabella 6.1: Confronto fra bound primali al nodo radice per il benchmark $S1$

$B1$		*root gap %*				
n	c	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-44]$	2	5.43	0.33	0.00	0.20	0.20
	3	11.23	1.30	0.00	2.24	2.24
	4	8.95	1.95	0.55	1.55	1.00
	5	16.54	5.96	0.00	3.74	3.74
$[45-65]$	2	4.71	0.55	0.23	0.27	0.04
	3	10.61	1.07	0.57	6.16	5.60
	4	17.76	4.25	0.45	2.65	2.21
	5	18.02	5.45	1.17	3.17	2.00

$B2$		*root gap %*				
n	c	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-44]$	2	8.68	1.09	0.08	0.07	**-0.02**
	3	18.35	3.19	0.00	3.10	3.10
	4	18.96	6.13	0.00	6.82	6.82
	5	37.85	0.00	0.00	0.00	0.00
$[45-65]$	2	5.87	0.37	0.00	0.47	0.47
	3	17.09	3.30	0.00	4.12	4.12
	4	26.59	2.60	0.49	11.73	11.24
	5	30.43	6.63	0.01	10.32	10.31

$B3$		*root gap %*				
n	c	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-44]$	2	17.88	1.23	0.48	0.00	**-0.48**
	3	14.82	1.01	0.02	0.02	0.01
	4	34.49	2.87	0.66	2.21	1.54
	5	3.52	4.02	0.00	0.00	0.00
$[45-65]$	2	30.28	2.92	0.70	2.54	1.84
	3	35.94	3.87	0.16	7.86	7.70
	4	45.00	9.22	1.70	2.26	0.56
	5	38.94	8.33	1.29	4.65	3.36

$B4$		*root gap %*				
n	c	PB_{comb}	PB_{sem}	PB_{TS}	PB_{cplex}	$\Delta_{PB_{cplex}-PB_{TS}}$
$[30-44]$	2	33.77	2.39	0.00	0.11	0.11
	3	49.78	6.02	0.00	6.02	6.02
	4	67.25	18.72	0.00	10.14	10.14
	5	27.81	12.49	0.00	1.57	1.57
$[45-65]$	2	36.59	4.68	0.57	0.41	**-0.16**
	3	59.90	8.99	0.00	18.65	18.65
	4	63.84	21.77	0.00	12.54	12.54
	5	50.69	20.34	0.10	10.79	10.70

Tabella 6.2: Confronto fra bound primali al nodo radice per il benchmark di Bosio $S2$

```
Iter  PP Objective     DD Objective    PInfeas  DInfeas    Nu      StepLength   Pnrm
-------------------------------------------------------------------------------------
0    1.00000000e+02  -1.13743137e+05   2.2e+00  3.8e+02  1.1e+05  0.00  0.00    0.00
1    1.36503342e+06  -6.65779055e+04   5.1e+00  2.2e+02  1.1e+04  1.00  0.33    4.06
2    1.36631922e+05  -6.21604409e+03   5.4e+00  1.9e+01  4.5e+02  1.00  1.00    7.85
3    5.45799174e+03  -3.18292092e+03   1.5e-03  9.1e+00  7.5e+01  1.00  1.00   17.63
4    1.02930559e+03  -5.39166166e+02   1.1e-05  5.3e-01  2.7e+01  1.00  1.00    7.58
5    4.30074471e+02  -3.02460061e+01   3.3e-09  0.0e+00  5.6e+00  1.00  1.00   11.36
...
11   8.99999824e+00   8.99999617e+00   1.1e-16  0.0e+00  1.7e-08  1.00  1.00    7.03
12   8.99999668e+00   8.99999629e+00   2.9e-19  0.0e+00  3.4e-09  1.00  1.00   14.19
```

Iter	indica l'iterazione corrente
PP Objective	il valore del bound primale
DD Objective	il valore del bound duale
PInfeas	l'inammissibilità per il problema primale
DInfeas	l'inammissibilità per il problema duale
Nu	il parametro barriera ν
StepLength	indica il multiplo del passo nel primale e duale
Pnrm	rappresenta la prossimità al cammino centrale

Figura 6.1: Output che mostra le iterazioni dell'algoritmo del cammino centrale utilizzato per il rilassamento semidefinito.

fase di branching vengono passate dal nodo padre ai nodi figli le variabili duali ottime del rilassamento. Di seguito sono riportati alcuni fra i parametri che abbiamo ritenuto importante modificare.

- Il parametro ρ è detto potenziale e deve essere maggiore di 1.
- Il parametro penalità Γ permette di influenzare significativamente la convergenza dell'algoritmo.
- Il parametro *PNorm* permette di stabilire quanto la soluzione finale deve essere vicina al cammino centrale.
- Il parametro *GapTollerance* permette di stabilire la condizione di stop; l'algoritmo termina quando il duality-gap relativo è minore del valore della tolleranza.
- Il parametro *ReuseMatrix* indica quante volte riutilizzare l'Hessiana della funzione barriera in ogni iterazione dell'algoritmo.

$$\begin{cases} \rho = 5 \\ \Gamma = 0 \\ PNorm = 1.0 \\ GapTollerance = 10^{-4} \\ ReuseMatrix = 2 \end{cases}$$

I valori dei parametri che seguono sono stati determinati attraverso un'estesa campagna di simulazione.

Il solutore permette inoltre di monitorare con diversi gradi di verbosità l'evoluzione della computazione. La figura 6.1 mostra un esempio di output restituito dal solutore DSDP5 durante il rilassamento del modello semidefinito.

In ogni riga sono riportate le informazioni significative associate alle iterazioni dell'algoritmo.

Per rafforzare la formulazione semidefinita sono stati scelti alcuni fra i vincoli proposti in (4.2.1). Numerosi test hanno evidenziato che pagando un opportuno costo computazionale, alcuni vincoli sono più stringenti di altri. Per poter agganciare questa tecnica di rilassamento abbiamo dovuto determinare quali vincoli considerare nel $SDP_{B\&B}$, al fine di assicurare il migliore compromesso tra la qualità del bound e il tempo necessario ad ottenerlo.

La configurazione di vincoli conclusiva prevede i vincoli di multiple choice semplici (2.14), rafforzati con i vincoli migliorati di multiple choice IMC_A (4.3.2) con l'aggiunta dei vincoli di taglio $4C''$ (4.3.3). La campagna completa con i risultati dettagliati è stata omessa.

Nelle tabelle (6.3, 6.4) mostriamo il confronto dei bound duali implementati nell'algoritmo $SDP_{B\&B}$ ed individuate da Cplex al nodo radice sfruttando il rafforzamento del rilassamento continuo (2.2) tramite generazione di tagli generici .

Analogamente a quanto definito per i bound primali, indichiamo con $gap\,DB$ il gap medio in percentuale, tra il valore del bound duale e il $Best\,Known$ (BK) calcolato come

$$gap = \frac{DB - BK}{BK}$$

Quindi il termine $gap\,DB_{comb}$ indica il gap medio percentuale tra il valore del bound duale combinatorio e il $Best\,Known$. Analogamente indichiamo con $gap\,DB_{sem}$,$gap\,DB_{cplex}$ il gap del bound duale ottenuto rispettivamente dal rilassamento semidefinito e da Cplex.

Indichiamo con $Time(s)\,DB_{comb}$, $Time(s)\,DB_{sem}$ e $Time(s)\,DB_{cplex}$ il tempo medio in secondi necessario per il calcolo rispettivamente del bound duale combinatorio, rilassamento semidefinito e bound duale di Cplex.

La prima tabella (6.3), relativa al benchmark $S1$ mostra in modo molto evidente la differenza tra i tempi di calcolo necessari alla valutazione del nodo radice. Nelle ultime due colonne si nota che il tempo impiegato da Cplex cresce con il numero delle classi della partizione. Da pochi secondi necessari per valutare istanze con $[3-7]$ classi fino ad un tempo mille volte superiore per istanze con $[10-40]$ classi. Al contrario i nostri algoritmi di bounding risultano molto più efficienti sia in termini computazionali che in termini di qualità. Infatti, per le istanze citate i tempi necessari al bounding vanno da pochi centesimi fino a 7 secondi per le istanze con un maggior numero di classi.

Per quanto riguarda la qualità dei bound del nostro algoritmo notiamo una caratteristica ricorrente. Il bound combinatorio funziona molto bene sulle istanze che hanno un basso numero di classi mentre la qualità peggiora via via che le classi aumentano. Il comportamento del bound semidefinito è esattamen-

te l'opposto, ovvero notiamo gap abbastanza scarsi che vanno progressivamente migliorando al crescere del numero di partizioni.

Analogamente a quanto avviene per il rilassamento semidefinito, notiamo che anche Cplex tende a fornire migliori risultati sulle istanze che hanno un numero elevato di classi.

Questa caratteristica dei bound duali assicura una maggiore robustezza rispetto alle istanze che possono essere fornite all'algoritmo. Infatti, come spiegato nella sezione 5.4, entrambe le tecniche di rilassamento vengono impiegate per valutare ogni sottoproblema.

Nella prima tabella (6.3) notiamo che il bound duale determinato dal nostro algoritmo rimane sempre più stretto di quello trovato da Cplex che ha bisogno di un tempo di CPU molto maggiore.

Tale comportamento può essere spiegato come segue: supponendo che le classi abbiano lo stesso numero di vertici, siccome l'insieme dei vertici è un dato dell'istanza, al crescere del numero di classi deve diminuire necessariamente la cardinalità di ogni singola classe. Entrambe le tecniche di rilassamento considerano una soluzione costituita da valori frazionari in cui rimangono rispettati i vincoli di multiple choiche. Per questo motivo la somma delle variabili decisionali appartenenti alla stessa classe deve necessariamente essere 1. Al diminuire del numero di classi aumenta necessariamente la cardinalità di ogni singola classe. Di conseguenza, il rilassamento dovrà distribuire i contributi della sommatoria su tutti gli elementi della classe ottenendo una stima molto lasca rispetto al valore della soluzione ottima.

Il bound combinatorio ha un comportamento opposto rispetto al rilassamento semidefinito, ovvero permette di trovare bound di maggiore qualità sulle istanze che hanno un limitato numero di classi ($[3-5]$).

Per quanto riguarda i test condotti sul benchmark $S2$, la tabella 6.4 mostra che i rilassamenti, ottenuti dal modello semidefinito e dal modello lineare intero (Cplex) sono sempre migliori del bound combinatorio. I risultati sono conformi ai precedenti, infatti la cardinalità delle classi si mantiene contenuta $(2-5)$.

Emerge dai risultati che l'algoritmo sviluppato mostra rispetto a Cplex una migliore qualità nei bound trovati. Inoltre osserviamo che il tempo necessario ad ottenerli risulta molto inferiore, anche di alcuni ordini di grandezza.

6.4 Contributo del bound combinatorio

Al fine di rendere visibile il contributo del bound combinatorio derivante dall'esplorazione dell'albero di ricerca, abbiamo condotto una campagna di test con due diverse versioni dell'algoritmo di enumerazione. La prima ($SDP_{B\&B}^{-\{P,E\}}$) differisce dalla versione originale per la disattivazione degli algoritmi di preprocessing e di enumerazione esplicita. Il confronto è stato effettuato con una seconda versione ($SDP_{B\&B}^{-\{P,C,E\}}$) che si discosta dalla prima per l'assenza del bound combinatorio. In questo modo abbiamo potuto mettere in eviden-

$S1(A)$		$Gap\ \%$			$Time(s)$		
n	m	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-80]$	$[3-7]$	16.88	0.84	48.08	0.00	0.06	2.96
	$[8-14]$	19.79	0.22	61.74	0.00	0.12	13.77
	$[15-40]$	11.91	0.01	6.01	0.00	0.15	54.99
$[100-150]$	$[3-5]$	18.74	3.17	58.88	0.00	0.81	11.43
	$[10-24]$	23.52	0.14	75.50	0.00	0.79	150.98
	$[30-75]$	13.61	0.00	25.96	0.00	1.02	930.26
$[200-300]$	$[3-5]$	19.31	3.33	64.79	0.00	6.51	55.16
	$[10-40]$	25.59	0.07	87.83	0.00	5.07	1,437.35
	$[48-150]$	13.66	0.00	44.78	0.00	6.71	3,267.78

$S1(B)$		$Gap\ \%$			$Time(s)$		
n	m	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-80]$	$[3-7]$	15.21	33.62	51.74	0.00	0.06	5.36
	$[8-14]$	27.55	13.34	67.77	0.00	0.09	37.55
	$[15-40]$	23.68	1.17	11.06	0.00	0.10	134.15
$[100-150]$	$[3-5]$	6.76	98.72	56.16	0.00	0.67	10.41
	$[10-24]$	32.93	18.96	92.69	0.00	0.61	424.37
	$[30-75]$	27.20	1.14	39.44	0.00	0.79	1,565.02
$[200-300]$	$[3-5]$	4.54	159.04	55.19	0.00	5.11	103.90
	$[10-40]$	39.00	19.46	122.46	0.00	3.39	2,079.10
	$[48-150]$	29.89	1.27	65.15	0.00	4.92	3,284.28

$S1(C)$		$Gap\ \%$			$Time(s)$		
n	m	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-80]$	$[3-7]$	42.99	88.16	63.71	0.00	0.06	3.86
	$[8-14]$	105.66	50.46	104.17	0.00	0.09	21.16
	$[15-40]$	218.62	9.88	29.30	0.00	0.11	75.80
$[100-150]$	$[3-5]$	15.29	220.90	60.30	0.00	0.59	13.67
	$[10-24]$	127.91	73.28	166.23	0.00	0.50	364.58
	$[30-75]$	342.10	14.62	96.48	0.00	0.68	2,603.98
$[200-300]$	$[3-5]$	11.04	328.58	59.71	0.00	3.95	111.68
	$[10-40]$	174.48	86.42	277.60	0.00	2.95	2,024.19
	$[48-150]$	504.31	19.92	420.08	0.00	4.60	3,617.47

Tabella 6.3: Confronto fra bound duali al nodo radice per il benchmark $S1$

$S2(B1)$			$Gap\ \%$			$Time(s)$	
n	c	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-44]$	2	28.49	0.97	0.83	0.00	0.04	4.52
	3	36.84	2.98	15.37	0.00	0.03	6.37
	4	36.13	3.65	19.52	0.00	0.03	4.41
	5	31.73	5.62	13.02	0.00	0.02	2.42
$[45-65]$	2	33.10	0.99	1.95	0.00	0.07	16.16
	3	42.01	2.84	25.30	0.00	0.07	22.17
	4	45.47	4.98	31.00	0.00	0.09	19.85
	5	46.98	8.49	35.46	0.00	0.09	17.33

$S2(B2)$			$Gap\ \%$			$Time(s)$	
n	c	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-44]$	2	39.22	1.88	2.08	0.00	0.04	5.32
	3	53.22	3.40	23.44	0.00	0.03	6.85
	4	57.57	7.69	35.00	0.00	0.03	3.91
	5	50.53	11.54	25.08	0.00	0.03	2.29
$[45-65]$	2	41.78	1.43	3.97	0.00	0.07	17.53
	3	59.56	4.60	36.50	0.00	0.06	27.17
	4	63.65	7.10	47.69	0.00	0.09	20.75
	5	66.89	10.44	49.15	0.00	0.10	26.07

$S2(B3)$			$Gap\ \%$			$Time(s)$	
n	c	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-44]$	2	119.82	3.51	0.16	0.00	0.04	1.74
	3	65.31	5.48	6.57	0.00	0.03	2.98
	4	66.52	10.32	12.86	0.00	0.03	3.01
	5	28.38	4.63	2.79	0.00	0.03	0.63
$[45-65]$	2	166.89	6.09	5.99	0.00	0.08	14.64
	3	105.18	7.14	25.05	0.00	0.08	20.78
	4	99.61	13.33	38.77	0.00	0.08	13.73
	5	81.89	16.57	32.39	0.00	0.10	19.49

$S2(B4)$			$Gap\ \%$			$Time(s)$	
n	c	DB_{comb}	DB_{sem}	DB_{cplex}	DB_{comb}	DB_{sem}	DB_{cplex}
$[30-44]$	2	147.68	5.60	0.84	0.00	0.04	2.67
	3	85.69	8.85	15.69	0.00	0.03	5.15
	4	105.62	23.73	39.00	0.00	0.03	3.50
	5	67.40	19.75	13.80	0.00	0.03	1.95
$[45-65]$	2	191.62	9.04	8.38	0.00	0.07	15.62
	3	128.97	9.98	44.03	0.00	0.07	20.54
	4	135.43	24.82	62.07	0.00	0.08	17.51
	5	108.86	29.43	50.11	0.00	0.09	25.09

Tabella 6.4: Confronto fra bound duali al nodo radice per il benchmark di Bosio $S2$

za il contributo fornito dall'attivazione del bound combinatorio, senza avere influenze spurie legate agli algoritmi che abbiamo disattivato.

I test eseguiti sono stati progettati per mettere in evidenza il tempo di calcolo e il numero totale di nodi esplorati. La tabella 6.7 mostra il confronto delle due versioni dell'algoritmo $SDP_{B\&B}$.

Le prime due colonne con i risultati sono riferite alla versione $SDP_{B\&B}^{-\{P,E\}}$ e mostrano il tempo medio in secondi ($T(s)$) seguito dal gap rimanente in percentuale ($Gap\,\%$). Analogamente, le due colonne che seguono contengono i risultati relativi alla versione $SDP_{B\&B}^{-\{P,C,E\}}$. Le ultime due colonne sono riferite al numero medio di nodi esplorati ($Exp\,nodes$) per quelle istanze che sono terminate entro il tempo limite.

Osservando i risultati si nota che il bound combinatorio dà un contributo significativo nelle istanze medio - grandi ($n = [100 - 300]$) in cui ci sono poche classi. I valori che maggiormente sottolineano questa caratteristica sono stati marchiati in grassetto. Nonostante gran parte delle istanze grandi di tipo B e C non terminino nel tempo limite, osserviamo che il gap medio rimanente molto più basso nella versione dell'algoritmo con il bound combinatorio attivato. Si legge infatti una differenza più marcata in corrispondenza di istanze con $[3 - 5]$ classi. Ad esempio nella tabella che riporta le istanze di tipo B, in corrispondenza della riga $n[200 - 300]\, m[3 - 5]$ vediamo che il gap medio rimanente migliora del 52%. Allo stesso modo, per le istanze di tipo C, osserviamo che nella stessa posizione il gap rimanente passa da 111.93 a 3.29% nella versione con il rilassamento combinatorio attivo.

Come già evidenziato durante l'analisi dei bound al nodo radice in (6.2), il rilassamento combinatorio offre un sostegno concreto nella risoluzione delle istanze che hanno un limitato numero di classi, in cui invece, il rilassamento semidefinito mostra dei forti limiti. L'attivazione congiunta delle due tecniche permette di aumentare la robustezza all'algoritmo di enumerazione rispetto a tutte le classi di istanze proposte nei benchmark.

6.5 Contributo del preprocessing

Per verificare il contributo dell'algoritmo di preprocessing abbiamo fatto una campagna di simulazioni specifica al fine di evidenziare possibili miglioramenti. Il vantaggio del preprocessing si può misurare in diversi aspetti ad esempio, al nodo radice con un bound duale più stretto, che si traduce di solito in un tempo minore di calcolo totale.

Per questo test specifico sono state compilate due versioni dell'algoritmo di enumerazione $SDP_{B\&B}$. La prima versione $SDP_{B\&B}^{-E}$ si discosta dall'algoritmo di enumerazione originale per l'assenza dell'enumerazione esplicita ($-E$). Per il confronto abbiamo utilizzato la versione $SDP_{B\&B}^{-\{P,E\}}$ che differisce dalla precedente per l'assenza dell'algoritmo di preprocessing ($-\{P, E\}$).

I risultati sono riportati nelle tabelle 6.5, 6.6.
Nella prima tabella (6.5) sono contenuti i risultati delle simulazioni fino al nodo radice sulle istanze del benchmark $S1$.

Indichiamo con $gapDB^{-E}$ il gap medio espresso in percentuale relativo al bound duale (combinatorio DB_{comb} o semidefinito DB_{semi}) associato all'algoritmo $SDP^{-E}_{B\&B}$. Analogamente indichiamo con $gap\,DB^{-\{PE\}}$ il gap medio del bound duale associato all'algoritmo $SDP^{-\{P,E\}}_{B\&B}$.

I risultati mostrano che il preprocessing funziona molto bene nelle istanze con un basso numero di classi. Questo risultato era atteso poiché il preprocessing si basa sul rilassamento combinatorio. Come è più evidente sui tipi di istanze B e C, il gap associato al bound semidefinito riceve un beneficio significativo anche di alcune decine di punti percentuali sul gap medio, rispetto alla versione senza il preprocessing. I valori in grassetto mettono in evidenza nella tabella (6.5) questo aspetto.

Come abbiamo già accennato in precedenza, la qualità del rilassamento semidefinito è legato al numero di vertici in ogni partizione. L'azione del preprocessing consiste appunto nel ridurre le soluzioni del problema, eliminando quei vertici che non faranno mai parte della soluzione ottima. Il beneficio del preprocessing sfuma molto rapidamente all'aumentare del numero di classi.

La seconda tabella (6.6) mostra il confronto dei tempi medi di calcolo e il numero medio di nodi dell'albero di decisione esplorati.

Le prime due colonne con i risultati sono riferite alla versione $SDP^{-E}_{B\&B}$ e mostrano il tempo medio in secondi ($T(s)$) seguito dal gap rimanente in percentuale ($Gap\,\%$). Analogamente, le due colonne che seguono contengono i risultati relativi alla versione $SDP^{-\{P,E\}}_{B\&B}$. Le ultime due colonne sono riferite al numero medio di nodi esplorati ($Exp\,nodes$) per quelle istanze che sono terminate entro il tempo limite.
I risultati mostrati in tabella (6.6) evidenziano lo stesso andamento già visto per l'analisi al nodo radice. Il preprocessing permette di avere un vantaggio significativo su quelle istanze che hanno un basso numero di classi. Questo aspetto si osserva sistematicamente in tutti i tipi di istanze (A, B, C). Il beneficio maggiore si osserva nei tipi B, C dove si vede chiaramente un miglioramento sia in termini di tempo totale che in termini di numero di nodi esplorati. Ad esempio fra le istanze di grandi dimensioni di tipo C, la versione con il preprocessing riduce di 80 secondi il tempo medio di calcolo e il numero di nodi esplorati viene ridotto da 350 a 460.

6.6 Contributo dell'enumerazione esplicita

Questa tecnica si contrappone all'enumerazione detta implicita poiché prevede l'enumerazione di tutte le soluzioni ammissibili di un problema. Tale approccio non è di solito conveniente in problemi che hanno una natura combinatoria come il MEWCMC. Infatti abbiamo mostrato nella sottosezione (5.5) che il numero si soluzione S associate ad un'istanza sono pari a $S = m^c$ dove c rappresenta la cardinalità costante delle m classi della partizione dei vertici. Per questo motivo si verifica rapidamente che all'aumentare di m e c, l'enumerazione esplicita diventa presto impraticabile.

Nell'algoritmo $SDP_{B\&B}$ abbiamo introdotto questa tecnica per rafforzare

$S1(A)$		*root gap %*			
n	m	DB^{-E}_{comb}	$DB^{-\{P,E\}}_{comb}$	DB^{-E}_{semi}	$DB^{-\{P,E\}}_{semi}$
$[30-80]$	$[3-7]$	16.88	17.07	**0.84**	**0.92**
	$[8-14]$	19.79	19.79	0.22	0.22
	$[15-40]$	11.91	11.91	0.01	0.01
$[100-150]$	$[3-5]$	18.74	18.74	**3.17**	**3.21**
	$[10-24]$	23.52	23.52	0.14	0.14
	$[30-75]$	13.61	13.61	0.00	0.00
$[200-300]$	$[3-5]$	19.31	19.31	**3.33**	**3.34**
	$[10-40]$	25.59	25.59	0.07	0.07
	$[48-150]$	13.66	13.66	0.00	0.00

$S1(B)$		*root gap %*			
n	m	DB^{-E}_{comb}	$DB^{-\{P,E\}}_{comb}$	DB^{-E}_{semi}	$DB^{-\{P,E\}}_{semi}$
$[30-80]$	$[3-7]$	15.21	15.29	**33.62**	**36.88**
	$[8-14]$	27.55	27.55	13.34	13.34
	$[15-40]$	23.68	23.68	1.17	1.17
$[100-150]$	$[3-5]$	6.76	6.77	**98.72**	**129.40**
	$[10-24]$	32.93	32.93	18.96	18.96
	$[30-75]$	27.20	27.20	1.14	1.14
$[200-300]$	$[3-5]$	4.54	4.59	**159.04**	**202.48**
	$[10-40]$	39.00	39.00	19.46	19.46
	$[48-150]$	29.89	29.89	1.27	1.27

$S1(C)$		*root gap %*			
n	m	DB^{-E}_{comb}	$DB^{-\{P,E\}}_{comb}$	DB^{-E}_{semi}	$DB^{-\{P,E\}}_{semi}$
$[30-80]$	$[3-7]$	42.99	43.02	**88.16**	**97.00**
	$[8-14]$	105.66	105.66	50.46	50.46
	$[15-40]$	218.62	218.62	9.88	9.88
$[100-150]$	$[3-5]$	15.29	15.32	**220.90**	**280.76**
	$[10-24]$	127.91	127.91	73.28	73.28
	$[30-75]$	342.10	342.10	14.62	14.62
$[200-300]$	$[3-5]$	11.04	11.25	**328.58**	**439.76**
	$[10-40]$	174.48	174.48	86.42	86.42
	$[48-150]$	504.31	504.31	19.92	19.92

Tabella 6.5: Tabella che mostra il contributo del preprocessing al nodo radice. Il confronto è stato effettuato tra versione $SDP^{-E}_{B\&B}$ (senza enumerazione esplicita) e la versione $SDP^{-\{P,E\}}_{B\&B}$(senza il preprocessing e senza l'enumerazione esplicita).

$S1(A)$		$SDP^{-E}_{B\&B}$		$SDP^{-\{P,E\}}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP^{-E}_{B\&B}$	$SDP^{-\{P,E\}}_{B\&B}$
	$[3-7]$	**0.49**	**0.00**	**0.54**	**0.00**	**10.00**	**11.00**
$[30-80]$	$[8-14]$	1.15	0.00	1.16	0.00	12.20	12.20
	$[15-40]$	0.58	0.00	0.59	0.00	4.14	4.14
	$[3-5]$	**20.65**	**0.00**	**25.01**	**0.00**	**43.86**	**49.00**
$[100-150]$	$[10-24]$	10.04	0.00	10.02	0.00	21.57	21.57
	$[30-75]$	3.35	0.00	3.27	0.00	4.00	4.00
	$[3-5]$	**387.87**	**0.00**	**469.61**	**0.00**	**97.29**	**108.14**
$[200-300]$	$[10-40]$	73.29	0.00	73.54	0.00	20.71	20.71
	$[48-150]$	18.85	0.00	18.10	0.00	3.40	3.40

$S1(B)$		$SDP^{-E}_{B\&B}$		$SDP^{-\{P,E\}}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP^{-E}_{B\&B}$	$SDP^{-\{P,E\}}_{B\&B}$
	$[3-7]$	17.03	0.00	17.17	0.00	340.00	350.00
$[30-80]$	$[8-14]$	85.60	0.00	85.69	0.00	1,125.00	1,125.00
	$[15-40]$	50.17	0.00	50.04	0.00	481.00	481.00
	$[3-5]$	**295.88**	**0.12**	**305.23**	**0.19**	**1,878.67**	**1,922.33**
$[100-150]$	$[10-24]$	1,313.98	3.55	1,312.38	3.32	8,665.00	8,665.00
	$[30-75]$	939.85	0.47	937.71	0.55	1,549.67	1,549.67
	$[3-5]$	1,329.85	1.42	1,494.74	1.43	1,432.50	1,486.50
$[200-300]$	$[10-40]$	-	14.12	-	14.21	-	-
	$[48-150]$	-	1.86	-	1.90	-	-

$S1(C)$		$SDP^{-E}_{B\&B}$		$SDP^{-\{P,E\}}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP^{-E}_{B\&B}$	$SDP^{-\{P,E\}}_{B\&B}$
	$[3-7]$	18.15	0.00	18.41	0.00	336.00	347.00
$[30-80]$	$[8-14]$	55.15	0.00	55.23	0.00	741.80	741.80
	$[15-40]$	19.40	0.00	19.33	0.00	191.86	191.86
	$[3-5]$	**584.27**	**0.00**	**589.53**	**0.00**	**2,694.71**	**2,734.43**
$[100-150]$	$[10-24]$	1,862.41	10.20	1,859.54	11.31	10,831.00	10,831.00
	$[30-75]$	980.18	3.56	975.22	3.47	1,733.67	1,733.67
	$[3-5]$	**302.03**	**3.09**	**481.86**	**3.20**	**350.33**	**459.67**
$[200-300]$	$[10-40]$	-	46.51	-	47.20	-	-
	$[48-150]$	-	14.10	-	13.99	-	-

Tabella 6.6: Tabella dei tempi (medi) che mostra il contributo del preprocessing. Il confronto è stato fatto tra la versione dell'algoritmo senza l'enumerazione esplicita ($SDP^{-E}_{B\&B}$) con la versione senza il preprocessing e senza l'enumerazione esplicita ($SDP^{-\{P,E\}}_{B\&B}$), sul benchmark $S1$. Tempo limite 1 ora.

$S1(A)$		$SDP_{B\&B}^{-\{P,E\}}$		$SDP_{B\&B}^{-\{P,C,E\}}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}^{-\{P,E\}}$	$SDP_{B\&B}^{-\{P,C,E\}}$
$[30-80]$	$[3-7]$	0.54	0.00	0.52	0.00	11.00	11.00
	$[8-14]$	1.16	0.00	1.14	0.00	12.20	12.20
	$[15-40]$	0.59	0.00	0.57	0.00	4.14	4.14
$[100-150]$	$[3-5]$	25.01	0.00	25.10	0.00	49.00	49.00
	$[10-24]$	10.02	0.00	10.04	0.00	21.57	21.57
	$[30-75]$	3.27	0.00	3.27	0.00	4.00	4.00
$[200-300]$	$[3-5]$	469.61	0.00	466.14	0.00	108.14	108.14
	$[10-40]$	73.54	0.00	72.53	0.00	20.71	20.71
	$[48-150]$	18.10	0.00	18.08	0.00	3.40	3.40

$S1(B)$		$SDP_{B\&B}^{-\{P,E\}}$		$SDP_{B\&B}^{-\{P,C,E\}}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}^{-\{P,E\}}$	$SDP_{B\&B}^{-\{P,C,E\}}$
$[30-80]$	$[3-7]$	17.17	0.00	17.25	0.00	350.00	354.33
	$[8-14]$	85.69	0.00	85.63	0.00	1,125.00	1,125.00
	$[15-40]$	50.04	0.00	50.06	0.00	481.00	481.00
$[100-150]$	$[3-5]$	305.23	0.19	1,084.05	0.98	1,922.33	4,074.67
	$[10-24]$	1,312.38	3.32	1,307.97	3.47	8,665.00	8,665.00
	$[30-75]$	937.71	0.55	936.16	0.55	1,549.67	1,549.67
$[200-300]$	$[3-5]$	1,494.74	**1.43**	-	**53.34**	1,486.50	-
	$[10-40]$	-	**14.21**	-	**20.16**	-	-
	$[48-150]$	-	1.90	-	1.90	-	-

$S1(C)$		$SDP_{B\&B}^{-\{P,E\}}$		$SDP_{B\&B}^{-\{P,C,E\}}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}^{-\{P,E\}}$	$SDP_{B\&B}^{-\{P,C,E\}}$
$[30-80]$	$[3-7]$	18.41	0.00	18.45	0.00	**347.00**	**351.67**
	$[8-14]$	55.23	0.00	54.97	0.00	741.80	741.80
	$[15-40]$	19.33	0.00	19.24	0.00	191.86	191.86
$[100-150]$	$[3-5]$	589.53	0.00	1,145.15	0.45	**2,734.43**	**4,102.33**
	$[10-24]$	1,859.54	11.31	1,852.76	10.08	10,831.00	10,831.00
	$[30-75]$	975.22	3.47	971.96	3.47	1,733.67	1,733.67
$[200-300]$	$[3-5]$	481.86	**3.20**	-	**111.93**	459.67	-
	$[10-40]$	-	**47.20**	-	**63.32**	-	-
	$[48-150]$	-	13.99	-	13.99	-	-

Tabella 6.7: Tabella dei tempi (medi) che mostra il contributo del rilassamento combinatorio confrontando la versione dell'algoritmo $SDP_{B\&B}^{-\{P,E\}}$ (senza il preprocessing e senza l'enumerazione esplicita) con la versione $SDP_{B\&B}^{-\{P,C,E\}}$ (senza il preprocessing, il rilassamento combinatorio e l'enumerazione esplicita) sul benchmark $S1$. Tempo limite 1 ora.

ulteriormente sia il contributo dato dal preprocessing che quello dato dal rilassamento combinatorio. Questo algoritmo viene attivato durante l'esplorazione dell'albero delle decisioni, per quei nodi che contengono fino a 500.000 soluzioni ammissibili.

La tabella (6.8) mostra il confronto dei tempi medi e dei nodi esplorati tra l'algoritmo $SDP_{B\&B}$ con la versione senza il contributo dell'enumerazione esplicita ($SDP^{-E}_{B\&B}$).

Le prime due colonne dei risultati sono riferite alla versione $SDP_{B\&B}$ e mostrano il tempo medio in secondi ($T(s)$), seguito dal gap rimanente in percentuale ($Gap\%$). Analogamente, le due colonne che seguono contengono i risultati relativi alla versione $SDP^{-E}_{B\&B}$. Le ultime due colonne sono riferite al numero medio di nodi esplorati ($Exp\,nodes$), per quelle istanze che sono terminate entro il tempo limite.

Le differenze principali tra le due versioni dell'algoritmo sono state marcate in grassetto. Come ci saremmo aspettati, l'enumerazione esplicita dà il massimo contributo in due casi; quando le istanze sono di piccole dimensioni ($n[30-80]$) e quando la partizione dell'insieme dei nodi è formata da poche classi (m). In questo caso il numero di soluzioni totali tende a rimanere più basso poiché il termine m^c cresce più velocemente all'aumentare di c. In altre parole, le istanze con molte classi contengono pochi elementi e quindi hanno un maggior numero di soluzioni. Di conseguenza, l'enumerazione esplicita non viene quasi mai attivata, neppure negli ultimi livelli dell'albero di esplorazione.

Dalla tabella (6.8), osserviamo che l'enumerazione esplicita dà un contributo molto significativo nella risoluzione delle istanze grandi con poche classi. Questi miglioramenti si riscontrano sia sul tempo medio di calcolo che sul numero di nodi esplorati. Ad esempio nella classe B, le istanze di grandi dimensioni ($n[200-500]$), con $m = [3-5]$ mostrano che l'enumerazione è molto efficace; il tempo di calcolo migliora da 1329 a soli 379 secondi, con un numero di nodi esplorati che si riduce ad un quarto. Analogamente per il tipo C, le istanze di medie dimensioni ($n = [100-150]$) con $m = [3-5]$ osserviamo che il miglioramento dovuto all'enumerazione esplicita è anche più consistente. Il tempo di calcolo passa da 296 secondi per la versione senza l'enumerazione esplicita a 7.45 secondi. Analogamente per il numero di nodi esplorati osserviamo una differenza importante, da 1879 a soli 30 nodi.

6.7 Confronto dei tempi di calcolo

In questa sezione mostriamo il confronto sui tempi di calcolo totali della versione completa del nostro algoritmo $SDP_{B\&B}$ con il solutore commerciale Cplex, a cui è stato fornito il modello di programmazione lineare intera (2.2), con i vincoli di taglio (2.11). Le simulazioni sono state condotte imponendo un tempo limite di 1 h; in caso non fosse stata trovata la soluzione ottima viene riportato il gap in percentuale rispetto al miglior bound primale conosciuto.

Le tabelle 6.9, 6.10 mostrano i risultati degli esperimenti computazionali rispettivamente sul benchmark $S1$ e $S2$.

$S1(A)$		$SDP_{B\&B}$		$SDP^{-E}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}$	$SDP^{-E}_{B\&B}$
$[30-80]$	$[3-7]$	**0.13**	0.00	**0.49**	0.00	**2.33**	**10.00**
	$[8-14]$	1.09	0.00	1.15	0.00	11.40	12.20
	$[15-40]$	0.59	0.00	0.58	0.00	4.14	4.14
$[100-150]$	$[3-5]$	**3.98**	0.00	**20.65**	0.00	**10.71**	**43.86**
	$[10-24]$	9.65	0.00	10.04	0.00	20.43	21.57
	$[30-75]$	3.35	0.00	3.35	0.00	4.00	4.00
$[200-300]$	$[3-5]$	**70.93**	0.00	**387.87**	0.00	**23.57**	**97.29**
	$[10-40]$	71.77	0.00	73.29	0.00	20.71	20.71
	$[48-150]$	18.45	0.00	18.85	0.00	3.40	3.40

$S1(B)$		$SDP_{B\&B}$		$SDP^{-E}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}$	$SDP^{-E}_{B\&B}$
$[30-80]$	$[3-7]$	**0.56**	0.00	**17.03**	0.00	**10.00**	**340.00**
	$[8-14]$	**32.88**	0.00	**85.60**	0.00	**456.20**	**1,125.00**
	$[15-40]$	49.94	0.00	50.17	0.00	474.71	481.00
$[100-150]$	$[3-5]$	**7.45**	0.00	**295.88**	0.12	**30.14**	**1,878.67**
	$[10-24]$	1,278.81	3.55	1,313.98	3.55	8,567.00	8,665.00
	$[30-75]$	934.62	0.47	939.85	0.47	1,549.67	1,549.67
$[200-300]$	$[3-5]$	**379.21**	0.46	**1,329.85**	1.42	**367.33**	**1,432.50**
	$[10-40]$	-	14.12	-	14.12	-	-
	$[48-150]$	-	1.86	-	1.86	-	-

$S1(C)$		$SDP_{B\&B}$		$SDP^{-E}_{B\&B}$		$Exp\,nodes$	
n	m	$T(s)$	$Gap\,\%$	$T(s)$	$Gap\,\%$	$SDP_{B\&B}$	$SDP^{-E}_{B\&B}$
$[30-80]$	$[3-7]$	**0.63**	0.00	**18.15**	0.00	**11.33**	**336.00**
	$[8-14]$	**32.80**	0.00	**55.15**	0.00	**431.00**	**741.80**
	$[15-40]$	19.38	0.00	19.40	0.00	189.86	191.86
$[100-150]$	$[3-5]$	**7.63**	0.00	**584.27**	0.00	**29.29**	**2,694.71**
	$[10-24]$	1,849.37	10.20	1,862.41	10.20	10,805.00	10,831.00
	$[30-75]$	976.07	3.56	980.18	3.56	1,733.67	1,733.67
$[200-300]$	$[3-5]$	**385.71**	1.29	**302.03**	3.09	367.33	350.33
	$[10-40]$	-	46.51	-	46.51	-	-
	$[48-150]$	-	14.10	-	14.10	-	-

Tabella 6.8: Tabella dei tempi (medi) che mostra il contributo dell'enumerazione esplicita confrontando la versione dell'algoritmo $SDP_{B\&B}$ con la versione $SDP^{-\{E\}}_{B\&B}$ (senza l'enumerazione esplicita) sul benchmark $S1$. Tempo limite 1 ora.

Le prime due colonne si riferiscono all'algoritmo $SDP_{B\&B}$ e indicano il tempo di CPU medio in secondi ($Time(s)$) e il gap in percentuale rispetto alla migliore soluzione primale conosciuta. Le successive due colonne contengono le stesse informazioni ma riferite al solutore $Cplex$. Nelle ultime due colonne troviamo il confronto dei tempi di calcolo sulle istanze che sono terminate nel tempo limite calcolato come

$$\Delta_{SDP_{B\&B}-Cplex} = (1 - \frac{time_{cplex} - time_{SDP_{B\&B}}}{time_{cplex}}) \times 100 \tag{6.1}$$

ovvero il valore della colonna indica quale è la percentuale di tempo impiegato dall'algoritmo $SDP_{B\&B}$ rispetto a Cplex. Tale valore ha il vantaggio di essere compreso tra $[0, 100]$ se il tempo computazionale di $DSP_{B\&B}$ è minore di quello di $cplex$.

La prima tabella (6.9) mostra i risultati riferiti al benchmark $S1$. Le istanze di tipo A sono state risolte all'ottimo dal nostro algoritmo di enumerazione in un tempo medio che varia da meno di un secondo (0.59) a 71 secondi per le istanze di grandi dimensioni. Il confronto, indicato nell'ultima colonna, è possibile ove ci sia almeno un' istanza che termina entro il tempo limite. I valori percentuali mostrano per questo set di istanze un divario di $2-3$ ordini di grandezza nel tempo medio di calcolo. Dove le simulazioni non terminano affatto nel tempo limite $(-)$ possiamo considerare il gap relativo rispetto alla migliore soluzione primale conosciuta. In questo caso si osservano gap residui che vanno dallo 6.51% per $n[100-150]\, m[3-5]$ fino a 110% per il gruppo di istanze $n[200-300]\, m[10-40]$.

Per quanto riguarda i tipi di istanze B, C il confronto dei tempi medi computazionali rimane dello stesso ordine di grandezza così come il confronto fra i gap residui.

Analizzando le simulazioni contenenti le istanze non aggregate osserviamo per il benchmark $S1$ il seguente numero di istanze risolte all'ottimo, divise per algoritmo:

$S1$		$SDP_{B\&B}$			$Cplex$		
Tipo	Totale istanze	# risolte	%	$gap\, z^*$ %	# risolte	%	$gap\, z^*$ %
A	55	55	100	0.00	26	47.3	67.06
B	55	36	65.4	7.27	21	38.2	78.26
C	55	36	65.4	25.83	26	47.3	528.31

la tabella mostra il numero di istanze totali risolte ($\#\, risolte$), la relativa percentuale e il gap medio ($gap\, z^*$) calcolato su tutte le istanze dello stesso tipo che non terminano entro il tempo limite. I risultati mostrano che l'algoritmo $SDP_{B\&B}$risolve tutte le istanze di tipo A mentre per i tipi B e C non riesce a chiuderne all'ottimo il 35% lasciando un gap residuo percentuale rispettivamente del 7.27% e 25.83%. Per il solutore si osservano delle grosse difficoltà, sia nella percentuale di istanze chiuse all'ottimo che nel gap residuo, per quelle

istanze che non terminano nel tempo limite. Infatti osserviamo per il tipo C che sono state chiuse solo il 47.3% delle istanze e sulle rimanenti è rimasto un gap medio pari al 528%. Tale risultato sottolinea chiaramente la difficoltà dimostrata dal solutore Cplex nell'affrontare tali istanze.

Nella tabella 6.10, riferita al benchmark $S2$, sono riportati i confronti dei tempi medi con l'algoritmo $\mathcal{L}_{B\&B}$ proposto da Bosio in ([6]) e con il solutore Cplex. Tutte le istanze di questo benchmark sono state risolte all'ottimo nel tempo limite.

Dalla prima alla terza colonna sono riportati i tempi medi rispettivamente dell'algoritmo $SDP_{B\&B}$, di Cplex e dell'algoritmo basato sul rilassamento Lagrangeano $\mathcal{L}_{B\&B}$. Le ultime due colonne mostrano il confronto dei tempi con il nostro algoritmo ($SDP_{B\&B}$) analogamente a come definito in 6.1.

I risultati della tabella mettono in evidenza che $SDP_{B\&B}$ è sempre più veloce sia dell'algoritmo basato su rilassamento Lagrangeano, sia del solutore Cplex. In base al tipo di istanze, il $SDP_{B\&B}$ è fino a 3 ordini di grandezza più veloce dagli altri due algoritmi esatti.

$S1(A)$		$SDP_{B\&B}$		$Cplex$		$\Delta_{SDP_{B\&B}-Cplex}$
n	m	$Time(s)$	$Gap\,(z^*)\,\%$	$Time(s)$	$Gap\,(z^*)\,\%$	$Time\,\%$
$[30-80]$	$[3-7]$	0.13	0.00	34.07	0.00	0.39
	$[8-14]$	1.09	0.00	244.14	0.00	0.45
	$[15-40]$	0.59	0.00	168.02	0.00	0.35
$[100-150]$	$[3-5]$	3.98	0.00	779.13	6.51	0.51
	$[10-24]$	9.65	0.00	-	48.64	-
	$[30-75]$	3.35	0.00	1,764.22	27.75	0.19
$[200-300]$	$[3-5]$	70.93	0.00	1,467.59	45.97	4.83
	$[10-40]$	71.77	0.00	-	110.70	-
	$[48-150]$	18.45	0.00	-	70.23	-

$S1(B)$		$SDP_{B\&B}$		$Cplex$		$\Delta_{SDP_{B\&B}-Cplex}$
n	m	$Time(s)$	$Gap\,(z^*)\,\%$	$Time(s)$	$Gap\,(z^*)\,\%$	$Time\,\%$
$[30-80]$	$[3-7]$	0.56	0.00	74.01	0.00	0.75
	$[8-14]$	32.88	0.00	579.66	0.00	5.67
	$[15-40]$	49.94	0.00	287.11	0.62	17.39
$[100-150]$	$[3-5]$	7.45	0.00	1,202.26	7.86	0.62
	$[10-24]$	1,278.81	3.55	-	85.37	-
	$[30-75]$	934.62	0.47	-	48.28	-
$[200-300]$	$[3-5]$	379.21	0.46	-	51.98	-
	$[10-40]$	-	14.12	-	139.38	-
	$[48-150]$	-	1.86	-	94.25	-

$S1(C)$		$SDP_{B\&B}$		$Cplex$		$\Delta_{SDP_{B\&B}-Cplex}$
n	m	$Time(s)$	$Gap\,(z^*)\,\%$	$Time(s)$	$Gap\,(z^*)\,\%$	$Time\,\%$
$[30-80]$	$[3-7]$	0.63	0.00	49.73	0.00	1.26
	$[8-14]$	32.80	0.00	484.53	0.00	6.77
	$[15-40]$	19.38	0.00	205.34	2.33	9.44
$[100-150]$	$[3-5]$	7.63	0.00	617.68	0.00	1.23
	$[10-24]$	1,849.37	10.20	-	150.95	-
	$[30-75]$	976.07	3.56	-	122.86	-
$[200-300]$	$[3-5]$	385.71	1.29	2,496.20	44.53	15.45
	$[10-40]$	-	46.51	-	724.40	-
	$[48-150]$	-	14.10	-	1,674.84	-

Tabella 6.9: Confronto dei tempi medi di calcolo dell'algoritmo di enumerazione $SDP_{B\&B}$ con Cplex 11.2 per il benchmark $S1$ in un tempo limite di 1 ora.

$S2(B1)$		$SDP_{B\&B}$	$Cplex$	$\mathcal{L}_{B\&B}$	$\Delta_{SDP_{B\&B}-Cplex}$	$\Delta_{SDP_{B\&B}-\mathcal{L}_{B\&B}}$
n	m	$Time(s)$	$Time(s)$	$Time(s)$	$Time\,\%$	$Time\,\%$
	2	0.38	5.56	166.90	6.79	0.23
$[30-44]$	3	0.24	19.89	29.25	1.21	0.82
	4	0.26	16.81	128.65	1.56	0.20
	5	0.02	5.34	12.80	0.37	0.16
	2	1.04	22.55	49.96	4.61	2.08
$[45-65]$	3	1.56	93.74	398.27	1.67	0.39
	4	4.19	89.16	168.67	4.70	2.49
	5	8.47	154.39	662.73	5.48	1.28

$S2(B2)$		$SDP_{B\&B}$	$Cplex$	$\mathcal{L}_{B\&B}$	$\Delta_{SDP_{B\&B}-Cplex}$	$\Delta_{SDP_{B\&B}-\mathcal{L}_{B\&B}}$
n	m	$Time(s)$	$Time(s)$	$Time(s)$	$Time\,\%$	$Time\,\%$
	2	0.41	7.39	188.18	5.55	0.22
$[30-44]$	3	0.23	23.41	31.65	0.98	0.73
	4	0.31	20.56	127.65	1.52	0.24
	5	0.01	6.90	12.93	0.19	0.10
	2	1.36	28.63	97.24	4.76	1.40
$[45-65]$	3	3.95	117.77	478.27	3.35	0.83
	4	2.71	106.82	123.00	2.54	2.21
	5	3.39	158.59	889.75	2.14	0.38

$S2(B3)$		$SDP_{B\&B}$	$Cplex$	$\mathcal{L}_{B\&B}$	$\Delta_{SDP_{B\&B}-Cplex}$	$\Delta_{SDP_{B\&B}-\mathcal{L}_{B\&B}}$
n	m	$Time(s)$	$Time(s)$	$Time(s)$	$Time\,\%$	$Time\,\%$
	2	0.31	1.68	197.10	18.19	0.16
$[30-44]$	3	0.19	4.21	12.20	4.61	1.59
	4	0.29	4.92	383.25	5.95	0.08
	5	0.01	1.12	13.00	1.19	0.10
	2	1.52	21.21	113.13	7.15	1.34
$[45-65]$	3	2.32	92.94	23.32	2.50	9.97
	4	4.32	106.67	248.52	4.05	1.74
	5	13.30	218.85	310.04	6.08	4.29

$S2(B4)$		$SDP_{B\&B}$	$Cplex$	$\mathcal{L}_{B\&B}$	$\Delta_{SDP_{B\&B}-Cplex}$	$\Delta_{SDP_{B\&B}-\mathcal{L}_{B\&B}}$
n	m	$Time(s)$	$Time(s)$	$Time(s)$	$Time\,\%$	$Time\,\%$
	2	0.37	2.92	214.53	12.57	0.17
$[30-44]$	3	0.30	9.05	14.35	3.27	2.06
	4	0.43	11.25	47.67	3.85	0.91
	5	0.02	3.03	11.07	0.66	0.18
	2	2.06	23.61	106.13	8.71	1.94
$[45-65]$	3	2.79	137.44	252.45	2.03	1.11
	4	16.14	209.00	36.90	7.72	43.73
	5	14.96	291.72	470.40	5.13	3.18

Tabella 6.10: Confronto dei tempi medi di calcolo dell'algoritmo di enumerazione $SDP_{B\&B}$ con Cplex 11.2 e $\mathcal{L}_{B\&B}$ sul benchmark $S2$ in un tempo limite di 1 ora.

Capitolo 7

Conclusioni

Se dovessi rinascere, farei l'idraulico.
– Albert Einstein

Il Max Edge Weighted Clique problem with Multiple Choice contraints è un problema di ottimizzazione su grafo molto generico e, come molti altri problemi simili, è $NP - difficile$. Attualmente non sono noti algoritmi in grado di risolvere in tempo polinomiale questa classe di problemi: per ottenere metodi di risoluzione efficaci, è necessario studiare il problema in modo approfondito.

I metodi proposti in questa tesi hanno permesso di raggiungere quattro tipi di risultati: teorici, algoritmici, tecnologici e computazionali.

Sul versante teorico sono state introdotte delle nuove formulazioni anche sfruttando strumenti matematici di crescente interesse, quali la programmazione quadratica e semidefinita; diverse classi di vincoli proposti per formulazioni lineari sono stati riadattati a modelli di programmazione semidefinita. Inoltre abbiamo introdotto una tecnica di bounding basata esclusivamente sulle proprietà combinatorie del problema.

Sul versante algoritmico abbiamo proposto un'euristica di tipo Tabu Search, un nuovo algoritmo esatto che incorpora tecniche di bounding, preprocessing e di enumerazione esplicita ed abbiamo sperimentato entrambi gli algoritmi su due benchmark di istanze ($S1$, $S2$). L'euristica di Tabu Search è in grado di trovare 146 ottimi su 165 istanze del benchmark ($S1$), pari al 88.48% del totale, e 134 ottimi su 168 istanze del benchmark ($S2$) con un successo pari al 79.76%.

L'algoritmo esatto sfrutta le tecniche combinatorie e di programmazione semidefinta menzionate, incorpora euristiche primali di tipo combinatorio e di rounding ed include procedure di riduzione del problema. Tale algoritmo è stato confrontato con il solutore commerciale *ILOG Cplex 11.2.* I risultati computazionali mostrano che, nel tempo limite di un ora, l'algoritmo risolve il 77% delle istanze considerate nel benchmark ($S1$), mentre Cplex ne risolve solamente il 44%. Inoltre quando entrambi i metodi garantiscono l'ottimalità, il nostro algoritmo è fino a tre ordini di grandezza più veloce; quando le simulazioni non terminano entro il tempo limite, il gap residuo del nostro algoritmo

è un ordine di grandezza inferiore rispetto a quello di Cplex.

Tali risultati sono stati raggiunti anche grazie all'utilizzo di solutori di programmazione semidefinita ancora sperimentali, che utilizzano tecnologie innovative, ma che hanno anche richiesto una fase estesa di *tuning* dei parametri.

L'analisi sperimentale ha permesso di evidenziale alcune caratteristiche del problema: ad esempio i test hanno evidenziato quali strutture delle istanze rendono il problema computazionalmente più difficile.

Come già sottolineato, il MEWCMC è un problema molto generico: molti altri problemi reali possono essere ricondotti al MEWCMC tramite l'introduzione di vincoli aggiuntivi. Gli ottimi risultati sia teorici che computazionali raggiunti suggeriscono che le metodologie introdotte nella tesi possano essere adattate per risolvere anche questi problemi. Inoltre il successo delle tecniche di bounding, basate sulla programmazione semidefinita, suggeriscono che l'impiego di tecniche di *second-order conic programming,* ancora fortemente sperimentali per problemi di ottimizzazione combinatoria, possano portare ad algoritmi ancora più efficienti.

Appendice A

Programmazione semidefinita

Se i fatti e la teoria non
concordano, cambia i fatti
– Albert Einstein

Il crescente interesse per la programmazione semidefinita, nel corso dell'ultimo ventennio è dovuto alla maggiore attenzione verso quei problemi di ottimizzazione in cui le variabili non sono rappresentate da vettore bensì da una matrice simmetrica che deve essere semidefinita positiva. A tale fine, la programmazione semidefinta (semidefinite programming - SDP) ha lo scopo di determinazione una matrice semidefinita positiva al fine di ottimizzare una funzione obiettivo lineare soggetta a vincoli lineari.

Vedremo brevemente in questo capitolo come la programmazione lineare (LP) possa essere considerata un'estensione di quella semidefinita (SDP).

A.1 Richiami di ottimizzazione convessa

In questa sezione sono riportate alcune nozioni importanti della programmazione convessa per facilitare la comprensione della programmazione semidefinita.

Formulazione generale di un problema di programmazione lineare

$$
\begin{aligned}
& & max \quad & c^t x & \text{(A.1)}\\
(LP) \quad & & s.t. \quad & Ax \leq b & \text{(A.2)}\\
& & & x \geq 0 & \text{(A.3)}
\end{aligned}
$$

Questo problema rappresenta il problema primale; il problema duale ad esso associato è:

$$\begin{array}{rll} & min & b^t y \qquad (A.4) \\ (LD) & s.t. & A^t y \geq c \qquad (A.5) \\ & & y \geq 0 \qquad (A.6) \end{array}$$

dove $c, x \in \mathbb{R}^n$, $b, y \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$.

A.1.0.1 Dualità debole

Se $\tilde{x}$ è una soluzione ammissibile primale, soddisfa cioè i vincoli del primale e $\tilde{y}$ e ammissibile duale allora:

$$c^t \tilde{x} \geq b^t \tilde{y} \qquad (A.7)$$

A.1.0.2 Dualità forte

Il problema primale (P) ha una soluzione ottima se e solo se il problema duale (D) ammette una soluzione ottima. In questo caso i due valori coincidono:

$$c^t x - b^t y = 0$$

A.1.0.3 Teorema degli scarti complementari

Siano x^* e y^* due soluzioni ammissibili primale e duale rispettivamente. Esse sono soluzioni ottime se e solo se valgono le seguenti condizioni:

$$\begin{array}{lcll} x_j^* > 0 & \Rightarrow & \sum_i a_{ij} y_i^* = c_j & \forall j = 1, ..., .n \\ \sum_i a_{ij} y_j^* > c_j & \Rightarrow & x_j^* = 0 & \forall j = 1, ..., n \end{array}$$

A.1.0.4 Programmazione convessa

Formulazione generale di un problema di programmazione convessa

$$\begin{array}{rll} & min & f(x) \\ (CP) & s.t. & g_i(x) \leq 0 \qquad i = 1, ..., m \\ & & (a^k)^t x - b_k = 0 \quad k = 1, ..., p \end{array}$$

dove le funzioni $f, g_i : \mathbb{R} \to \mathbb{R}$ sono convesse e le derivate continue e $a^k \in \mathbb{R}^n$, $b_k \in \mathbb{R}$.

La regione ammissibile del problema (CP) è un insieme convesso.

Ogni ottimo locale di (CP) è ottimo globale.

Condizioni di ottimalità per (CP) (condizioni di **K**arush - **K**uhn - **T**ucker):

1. $g_i(x) \leq 0 \quad i = 1, ..., m$
2. $(a^k)^t x - b_k = 0 \quad k = 1, ..., p$
3. $\lambda_i \geq 0 \quad i = 1, ..., m$

4. $\lambda_i g_i(x) = 0 \quad i = 1, ..., m$

5. $\nabla f(x) + \sum_i \lambda_i \nabla g_i(x) + \sum_k \nu_k a^k = 0$

dove $\lambda \in \mathbb{R}^n$, $\nu \in \mathbb{R}^p$ sono i vettori dei moltiplicatori di Lagrange e il termine $\nabla f(x)$ è il gradiente di x nella funzione Lagrangeana:

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_k \nu_k [(a^k)^t x - b_k]$$

Duale del problema (CP) secondo Wolfe

$$\begin{array}{ll} max & \mathcal{L}(x, \lambda, \nu) \\ s.t. & \nabla_x \mathcal{L}(x, \lambda, \nu) = 0 \\ & \lambda \geq 0 \end{array}$$

Sufficienza e necessità delle condizioni di KKT

Sufficienza: Se (x^*, λ^*, ν^*) è una soluzione che soddisfa le condizioni di KKT è ottima per il problema (CP).

Definizione: I vincoli del problema (CP) si dicono fortemente consistenti se vale la condizione di regolarità di Slater:

$$\exists x^0 \ t.c. \ g_i(x) < 0 \quad i = 1, ..., m \quad (a^k)^t x - b_k = 0 \quad k = 1, ..., p$$

Necessità: Siano i vicoli di (CP) fortemente consistenti, i vettori a^k linearmente indipendenti e sia x^* un ottimo di (CP). Allora esistono due vettori λ^* e ν^* tali che (x^*, λ^*, ν^*) è una soluzione che rispetta le condizioni di KKT.

A.2 Definizione del programma semidefinito

In questa sezione si espone la teoria della Programmazione semidefinita.

Consideriamo il seguente problema:

$$\begin{array}{ll} min & c^t x \\ s.t. & F(x) \geq 0 \end{array} \tag{A.8}$$

dove $c, x \in \mathbb{R}^m$, $F(x) = F_0 + \sum_{i=1}^m x_i F_i$ e $F_0, .., F_m \in S^{n \times n}$.

Il vettore x costituisce le variabili del problema mentre il vettore c e le matrici F_i, $i = 1, ..., m$, rappresentano i dati dello stesso.

La disuguaglianza $F(x) \geq 0$ significa che la matrice $F(x)$ è semidefinita positiva, ovvero:

$$z^t F(x) z \geq 0 \quad \forall z \in \mathbb{R}^n$$

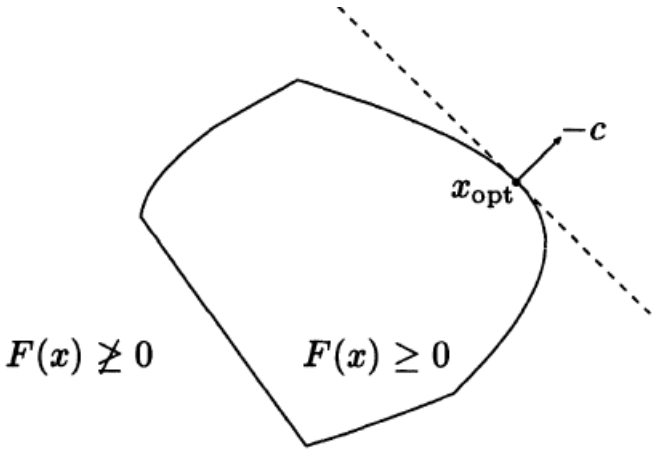

Figura A.1: Un semplice esempio di programma semidefinito con $x \in \mathbb{R}^2$, $F(x) \in \mathbb{R}^{7\times 7}$.

La disuguaglianza $F(x) \geq 0$ è anche detta disuguaglianza lineare di matrici; il problema A.8 è detto programma semidefinito. Questo problema fa parte dell'ottimizzazione convessa poiché la funzione obiettivo ed i vincoli sono funzioni convesse.

La funzione obiettivo è convessa perché è lineare; per quanto riguarda i vincoli, invece, date due matrici semidefinite positive $F(x) \geq 0$, $F(y) \geq 0$ segue

$$F(\lambda x + (1-\lambda)y) = \lambda F(x) + (1-\lambda)F(y) \geq 0 \quad \forall \lambda : 0 \leq \lambda \leq 1$$

ovvero la combinazione convessa di due matrici semidefinite positive è ancora una matrice semidefinita positiva.

Un semplice esempio per A.8 è mostrato in figura A.1. L'insieme delle soluzioni ammissibili $\{x : F(x) \geq 0\}$ costituisce l'insieme convesso racchiuso dalla linea scura. Il problema di programmazione semidefinita consiste, in questo caso, nello spostarsi il più lontano possibile in direzione $-c$, rimando nella regione ammissibile.

A.3 Dualità

In programmazione semidefinta il problema A.8 rappresenta il problema primale; il problema duale ed esso associato è il seguente:

$$\begin{array}{ll} max & -TrF_0Z \\ s.t. & TrF_iZ = c_i \quad i = 1, ..., m \\ & Z \geq 0 \end{array} \tag{A.9}$$

La variabile del problema è la matrice $Z \in S^{n\times n}$, i suoi dati sono $c \in \mathbb{R}^m$ e $F_i \in \mathbb{R}^{n\times n}$ per $i = 1, ..., m$.
Diremo che una matrice $Z \in S^{n\times n}$ è ammissibile per il problema duale se rispetta i vincoli $TrF_iZ = c_i \ \forall i = 1, ..., m$ e $Z \geq 0$.

In modo analogo un vettore $x \in \mathbb{R}^m$ si dirà ammissibile primale se rispetta i vincoli del primale e quindi se $F(x) \geq 0$.
Mentre nel problema primale, la variabile $x \in \mathbb{R}^m$ è libera, cioè non soggetta a vincoli di non negatività, la variabile del problema duale, $Z \in S^{n \times n}$, risulta soggetta a tale vincolo.

Si osserva che il problema A.9 può essere posto nella forma A.8, quindi è Anch'esso un programma semidefinito. La dimostrazione è disponibile in [7] a pagina 19.

La proprietà più importante del problema duale semidefinito è che esso fornisce un bound al valore ottimo del problema primale.

Siano $x \in \mathbb{R}^m$ un vettore ammissibile primale e $Z = Z^t \in \mathbb{R}^{n \times n}$ ammissibile duale, allora:

$$\begin{array}{ll} c^t x + TrF_0Z = (TrF_iZ = c_i) & = \sum_{i=1}^m TrF_iZx_i + TrF_0Z \\ & = (linearità\,della\,traccia) \quad = TrZF(x) \geq 0 \end{array}$$

Si ottiene

$$c^t x + TrF_0Z \quad = TrZF(x) \geq 0$$

o equivalentemente

$$-TrF_0Z \leq c^t x \tag{A.10}$$

cioè il valore della funzione obiettivo del problema duale è sempre minore o uguale di quello del problema primale.

Il duality gap Chiameremo la loro differenza $\eta = c^t x + TrF_0Z = TrF(x)Z$ il *duality gap* associato a x e Z. E' noto che η è una funzione lineare in x e in Z. Siano ora

$$p* = inf\{c^t x | F(x) \geq 0\}$$

$$d* = sup\{-TrF_0Z | Z = Z^t \geq 0,\, TrF_iZ = c_i,\, i = 1, ..., m\}$$

i rispettivi valori ottimi dei problemi A.8 e A.9. Dalla relazione A.10 si osserva facilmente che

$$\begin{array}{ll} -TrF_0Z \leq p^* & \forall x\, ammissibile\, primale \\ d^* \leq c^t x & \forall Z\, ammissibile\, duale \end{array}$$

cioè le matrici ammissibili duali forniscono un lower bound per il problema primale e viceversa i vettori ammissibili primali costituiscono un upper bound per il problema duale.

Da ciò segue facilmente:

$$d^* \leq p^*$$

cioè il valore del problema duale è minore - uguale del valore ottimo del problema primale.

Definiamo infine gli insiemi ottimi primale e duale:

$$X_{opt} = \{x | F(x) \geq 0,\, c^t x = p^*\}$$
$$Z_{opt} = \{Z | Z \geq 0,\, TrF_i Z = c_i,\, i = 1, ..., m\,,\, TrF_0 Z = d^*\}$$

Si osserva che l'insieme X_{opt} (o Z_{opt}) può essere vuoto anche se p^* (o d^*) è un valore finito.

Teorema primale - duale: I valori ottimi dei problemi primale e duale coincidono, cioè $p^* = d^*$, se:

1. il problema primale A.8 è strettamente ammissibile cioè $\exists x\, t.c.\, F(x) > 0$
2. il problema duale A.9 è strettamente ammissibile cioè $\exists Z = Z^t\, t.c.\, TrF_i Z = c_i$ per $i = 1, ..., m$

Se valgono entrambe le condizioni allora gli insiemi ottimo X_{opt}e Z_{opt} non sono vuoti. Per la dimostrazione del teorema si veda [32] di Nesterov e Nemirovskyalle pagine 103-110.

Assumiamo adesso che gli insiemi ottimi X_{opt} e Z_{opt} siano non vuoti: esistano cioè x e Z ammissibili tali che:

$$c^t x = TrF_0 Z = p^* = d^*$$

Da A.10 segue $TrF(x)Z = 0$ ma $F(x)$, $Z \geq 0$ allora per la proposizione 2 si concludiamo che:

$$ZF(x) = 0$$

Questa condizione si chiama, in analogia al caso lineare, condizione degli scarti complementari.

Il teorema primale - duale ci fornisce infine le condizioni di ottimalità per il programma semidefinito A.8 sotto l'ipotesi di stretta ammissibilità primale - duale: x è soluzione ottima per A.8 se e solo se esiste Z tale che:

$$\begin{cases} F(x) \geq 0 \\ Z \geq 0 \\ TrF_i Z = c_i \quad i = 1, ..., m \\ ZF(x) = 0 \end{cases}$$

A.4 SDP come estensione di LP

La programmazione semidefinita può essere considerata come l'estensione della programmazione lineare. Consideriamo a tale proposito il seguente problema lineare:

$$\begin{aligned} min \quad & c^t x \\ s.t. \quad & Ax + b \geq 0 \end{aligned} \tag{A.11}$$

dove $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$.

In questo caso il simbolo $\geq$ è inteso componente per componente: pertanto il problema consiste nel minimizzare una funzione obiettivo lineare soggetta ad un vincolo composto da m disequazioni lineari.

Osservazione: $v \in \mathbb{R}^n$, $v \geq 0 \Leftrightarrow diag(v) \geq 0$ dove quest'ultima formula indica che la matrice diagonale, che ha come elementi le componenti del vettore v, è semidefinita positiva.

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \quad v \geq 0 \, \forall i = 1, ..., n$$

Allora

$$diag(v) = \begin{pmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_n \end{pmatrix} \geq 0$$

In base all'osservazione precedente posso esprimere il problema A.11 nella forma A.8 imponendo:

$$F(x) = diag(Ax + b)$$

e osservando:

$$diag(Ax + b) = diag(Ax) + diag(b)$$

si ottiene infatti:

$$\begin{aligned} diag(Ax) &= \begin{pmatrix} a_{11}x_1 + ... + a_{1n}x_n & & 0 \\ & \ddots & \\ 0 & & a_{m1}x_1 + ... + a_{mn}x_n \end{pmatrix} \\ &= \begin{pmatrix} a_{11}x_1 & & 0 \\ & \ddots & \\ 0 & & a_{m1}x_1 \end{pmatrix} + ... + \begin{pmatrix} a_{1n}x_n & & 0 \\ & \ddots & \\ 0 & & a_{mn}x_n \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & & 0 \\ & \ddots & \\ 0 & & a_{m1} \end{pmatrix} x_1 + ... + \begin{pmatrix} a_{1n} & & 0 \\ & \ddots & \\ 0 & & a_{mn} \end{pmatrix} x_n \\ &= diag(a_1)x_1 + ... + diag(a_n)x_n \end{aligned}$$

dove a_i è l'$i-$esima colonna della matrice $A \in \mathbb{R}^{m \times n}$
$\Longrightarrow diag(Ax + b) = \sum_{i=1}^{n} diag(a_i)x_i + diag(b)$

Ponendo pertanto $F_i = diag(a_i)$ e $F_0 = diag(b)$ il problema A.11 diventa allora:

$$\begin{array}{ll} min & c^t x \\ s.t. & F(x) \geq 0 \end{array} \tag{A.12}$$

La programmazione semidefinita è quindi un'estensione di quella lineare: essa sostituisce i vincoli di disuguaglianza componente per componente con la disuguaglianza lineare di matrici.

Ci sono tuttavia importanti differenze tra i due problemi: i risultati della dualità sono più deboli per SDP rispetto a LP e non ci sono ancora metodi del simplesso semplici per SDP come in LP.

Utilizzando A.9 scrivo ora il duale di A.12:

$$\begin{array}{lll} max & -Tr\, diag(b)Z & \\ s.t. & Tr\, diag(a_i)Z = c_i & i = 1, ..., m \\ & Z \geq 0 & \end{array} \tag{A.13}$$

Questo problema può essere notevolmente semplificato osservando che la funzione obiettivo e il vincolo sottoposto ad uguaglianza utilizzano solo gli elementi diagonali di Z. Inoltre, se $Z \geq 0$ possiamo ritenere senza perdita di generalità 0 gli elementi non diagonali della stessa matrice, la quale rimane semidefinita positiva.
Possiamo pertanto scrivere:

$$Z = diag(z)$$

Il problema A.13 diventa:

$$\begin{array}{lll} max & -b^t y & \\ s.t. & a_i^t z = c_i & i = 1, ..., m \\ & z \geq 0 & \end{array} \tag{A.14}$$

il quale è proprio il duale di A.11 in LP.

Osservazione: La particolare struttura diagonale del problema appena studiato ci ha permesso di ridurre in modo notevole il numero di variabili dello stesso. In generale il problema duale può essere semplificato quando le matrici F_i sono strutturate: se ad esempio $F(x)$ è una matrice a blocchi diagonale, una tale struttura è assunta anche dalla matrice incognita Z.

Come già accennato, i risultati della dualità in LP sono più forti che in SDP: i valori ottimi di A.11,A.14 sono sempre uguali eccetto il caso in cui entrambi i problemi siano non ammissibili.

Possiamo infine rileggere la condizione degli scarti complementari:

$$ZF(x) = 0$$

Se $F(x) = diag(Ax + b)$, $Z = diag(z)$ allora vale:

$$0 = ZF(x) = diag(z)diag(Ax + b) \iff z_i(Ax + b)_i = 0 \ \ \forall i = 1, ..., n$$

Quest'ultima condizione coincide con la condizione degli scarti complementari valida in LP.

A.5 L'algoritmo del cammino centrale

Il solutore DSDP5 è in grado di trovare soluzioni (X_j, y_i, S_j) per problemi di ottimizzazione semidefinita espressi nella forma:

$$\begin{array}{lll} (P) & \inf & \sum_{j=1}^{p} C_j \bullet X_j \\ & s.t. & \sum_{j=1}^{p} A_{i,j} \bullet X_j = b_i \quad i = 1, ..., m, \quad X_j \in K_j \end{array}$$

$$\begin{array}{lll} (D) & \sup & \sum_{i=1}^{m} b_i y_i \\ & s.t. & \sum_{i=1}^{m} A_{i,j} y_i + S_j = C_j \quad j = 1, ..., p, \quad S_j \in K_j \end{array}$$

La formulazione (P) si riferisce al problema primale mentre (D) si riferisce a quello duale. Il simbolo $\bullet$ indica il prodotto di Frobenius.

Introduciamo di seguito la notazione necessaria per poter dare un accenno al funzionamento dell'algoritmo del cammino centrale utilizzato dal *dual*-scaling.

Le soluzioni che soddisfano le equazioni lineari sono dette ammissibili, mentre le altre sono dette non ammissibili. Lo spazio interno al cono delle matrici semidefinite positive è indicato da $\hat{K}$, e l'insieme delle soluzioni ammissibili per (P) e (D) sono indicate rispettivamente indicate con $\mathcal{F}^0(P)$ e $\mathcal{F}^0(D)$.

Il cammino centrale (*central-path*) è un arco di soluzioni strettamente ammissibili che gioca un ruolo cruciale negli algoritmi di tipo *primale-duale*. Come verrà illustrato in seguito, tale cammino C è parametrizzato da uno scalare $\nu > 0$ in modo che ogni soluzione parametrizzata $(X_\nu, y_\nu, S_\nu) \in C$ soddisfi le equazioni del sistema A.15.

Per una trattazione esaustiva dell'argomento rimandiamo a [26, 3].

Per semplicità vengono fatte le seguenti assunzioni:

1. il cono K è formato da un solo blocco semidefinito
2. le matrici A_i sono linearmente indipendenti
3. esiste una soluzione ammissibile $X \in \mathcal{F}^0(P)$
4. si conosce un punto di partenza $(y, S) \in \mathcal{F}^0(D)$ per il problema duale

Sotto queste assunzioni sappiamo che esiste una coppia di soluzioni ottime primali X^*e duali (y^*, S^*). Per le condizioni di scarto complementare vale $X^* \bullet S^* = X^* S^* = 0$

Inoltre per ogni $\nu > 0$, esiste una soluzione ammissibile primale-duale unica (X_ν, y_ν, S_ν) che soddisfa la condizione di ottimalità perturbata $X_\nu S_\nu = \nu I$. L'insieme di tutte le soluzioni $C \equiv \{(X_\nu, y_\nu, S_\nu) : \nu > 0\}$ è conosciuto come cammino centrale. Gli algoritmi basati sul *path-following* (A.2), come il cammino centrale, costruiscono una sequenza di soluzioni $(X_j, y_i, S_j) \subset$

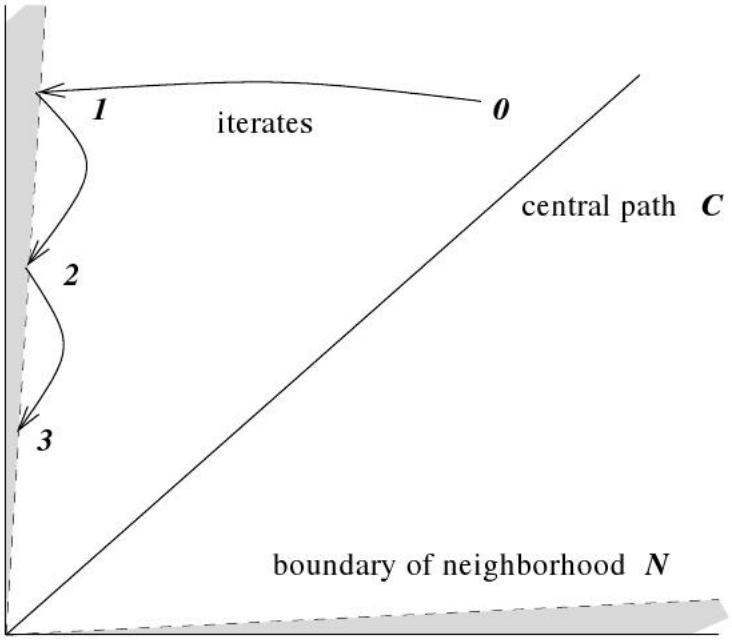

Figura A.2: La figura mostra i passi di un algoritmo di tipo path-following. L'algoritmo si muove all'interno dell'intorno individuato sul cammino centrale. Figura tratta da [26], pag. 402.

$\mathcal{F}^0(P) \times \mathcal{F}^0(D)$ in un intorno del cammino centrale in modo che il *duality-gap* tenda a zero. Il valore del *duality-gap* può essere determinato dalla funzione $\mu(X,S) = \frac{X \bullet S}{n}$ per tutte le coppie di soluzioni $(X,S) \in K \times K$. Si osserva che per tutte le soluzioni $(X,S) \in \hat{K} \times \hat{K}$, si ha $\mu(X,S) > 0$, salvo il caso in cui $XS = 0$. Inoltre, $\mu(X_\nu, S_\nu) = \nu$ per tutti i punti (X_ν, y_ν, S_ν) che giacciono sul cammino centrale.

L'algoritmo del cammino centrale applica il metodo di Newton al sistema

$$\begin{cases} AX = b \\ A^*y + S = C \\ XS = \nu I \end{cases} \tag{A.15}$$

per ottenere un algoritmo iterativo in grado di determinare la soluzione. La prima equazione impone l'ammissibilità primale, la seconda equazione garantisce l'ammissibilità duale mentre la terza impone le condizioni di ottimalità mediante il teorema dello scarto complementare.

Perturbando le variabili è possibile ricondurre il sistema precedente nella forma

$$\begin{cases} A(X + \Delta X) = b \\ A^*(\Delta y) + \Delta S = 0 \\ \nu S^{-1} \Delta S S^{-1} + \Delta X = \nu S^{-1} - X \end{cases}$$

per ricavare l'algoritmo iterativo proposto da Benson e Ye in [3].

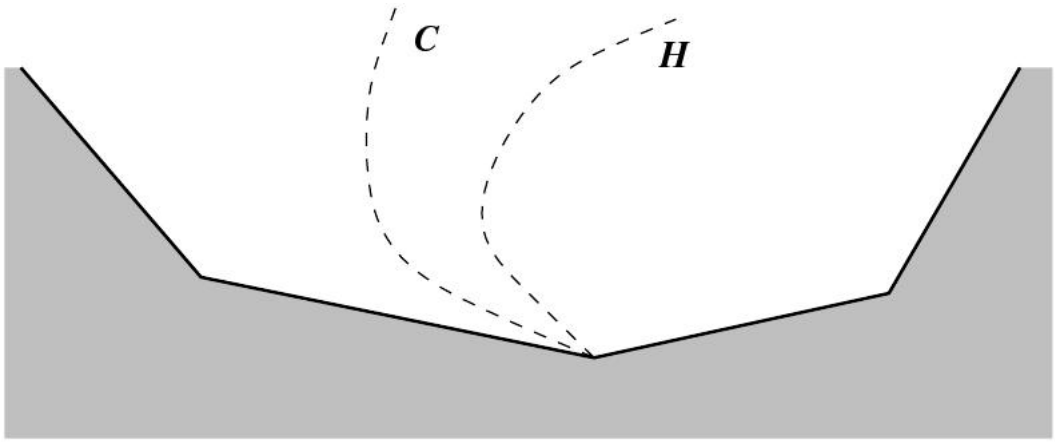

Figura A.3: La figura tratta da [26] mostra il cammino centrale C e la corrispondente traiettoria H, nello spazio ammissibile delle soluzioni.

Le figure A.2, A.3 mostrano un esempio di come un algoritmo di path-following si muove all'interno dell'intorno del cammino centrale.

L'algoritmo calcola ad ogni iterazione: la lunghezza del passo, una direzione di spostamento e una funzione penalità. I passi e la direzione indicano il modo con cui ci si sposta nella regione ammissibile, mentre la funzione barriera ha lo scopo di tenere la traiettoria compiuta dall'algoritmo lontana dai bordi della regione ammissibile.

Appendice B

Modello lineare intero per MEWCMC in linguaggio AMPL

Il testing non puo' mai rivelare l'assenza di bug.
– Dijkstra

I *generatori algebrici di modelli*, fra cui **AMPL** [8], costituiscono un'interfaccia verso il risolutore, ovvero un software in grado di gestire modelli e dati reali per fornire una soluzione. Le caratteristiche principali dei *generatori algebrici di modelli* sono le seguenti:

- fornire un linguaggio semplice per descrivere modelli complessi, un linguaggio che sia contemporaneamente ad *alto livello* e *formalmente strutturato* cioè accessibile ad un risolutore;
- permettere all'utente di comunicare con il risolutore attraverso file di testo anziché mediante strutture dati, in modo da non richiedergli conoscenze informatiche approfondite e da poter formulare il modello con un semplice editor di testo, svincolato dalla piattaforma sulla quale poi verrà risolto.
- permettere all'utente di comunicare con diversi risolutori, in modo da poter sfruttare i più potenti disponibili sul mercato;
- tenere distinti la *struttura logica* del modello (variabili di decisione, obiettivi, vincoli e le loro relazioni) dal valore di *dati numerici*.

Sebbene sia lecito scrivere in un solo file **AMPL** sia il modello sia i dati, è concettualmente preferibile tenere separati questi due termini, costruendo:

- un *file di modello*, obbligatoriamente di estensione **.mod**, che descrive la struttura logica del modello, (indici, variabili di decisione, funzione obiettivo e vincoli)
- un *file di dati*, obbligatoriamente di estensione **.dat**, che contiene l'istanza del problema

MEWCMC.mod	**Commento**
$param\, n;$	$\|V\|$
$param\, m;$	Numero classi V_k
$set\, Oggetti := 1..n;$	
$set\, Partizioni := 1..m;$	
$param\, V\, \{Partizioni, Oggetti\}\, default\, 0;$	Vertici in V_k
$param\, W\, \{Oggetti, Oggetti\};$	Matrice pesi W
$var\, x\, \{Oggetti\}\, binary;$	Variabili $x \in \{0,1\}$
$var\, y\, \{Oggetti, Oggetti\}\, binary;$	Variabili $y \in \{0,1\}$
$maximize\, z :$	
$sum\, \{i\, in\, Oggetti,\ j\, in\, Oggetti :\ i <= j\}\, W[i,j] * y[i,j];$	Obiettivo (2.4)
$subject\, to\, AttivazioneJ\, \{i\, in\, Oggetti, j\, in\, Oggetti :\ i <= j\} :$	Attivazione (2.6)
$y[i,j] <= x[j];$	
$subject\, to\, AttivazioneI \{i\, in\, Oggetti, j\, in\, Oggetti :\ i <= j\} :$	Attivazione (2.5)
$y[i,j] <= x[i];$	
$subject\, to\, Taglio\, \{i\, in\, Oggetti\} :$	Taglio (2.11)
$sum\{j\, in\, Oggetti\, y[i,j]\} = m * x[i];$	
$subject\, to\, Attivazione\, \{i\, in\, Oggetti, j\, in\, Oggetti :\ i <= j\} :$	Attivazione (2.7)
$x[i] + x[j] = y[i,j] + 1;$	
$subject\, to\, MultipleChoice\, \{k\, in\, Partizioni\} :$	Multiple choice (2.8)
$sum\{i\, in\, Oggetti :\ V[k,i] = 1\}\, x[i] = 1;$	
$end;$	Fine del modello

Tabella B.1: Esempio di traduzione del modello lineare (2.2) in AMPL (.mod)

Mantenendo fisicamente separato il modello dai dati, è possibile applicare il modello per la risoluzione di istanze differenti.

Il software **AMPL** è in grado di tradurre modelli di Programmazione Matematica scritti in linguaggio AMPL in un formato comprensibile a un generico risolutore di programmazione matematica.

B.1 Il file del modello *.mod*

Nella tabella B.1 viene proposta la traduzione in linguaggio matematico AMPL il modello lineare presentato nella sezione2.2. Il file di modello è un file di testo con estensione generalmente *.mod*.

test_a9c9.mod	**Commento**
param $n := 6;$	$\|V\|$
param $m := 3;$	Numero classi V_k
param $V :=$	
$[1,1]\,1\ [1,2]\,1$	Vertici in V_k
$[2,3]\,1\ [2,4]\,1$	"
$[3,5]\,1\ [3,6]\,1;$	"
param $W :=$	
$[1,1]\,9$	Matrice W
$[1,2]\,15\ [2,2]5$	"
$[1,3]\,40\ \ [2,3]\,33\ \ [3,3]\,9$	"
$[1,4]\,78\ \ [2,4]\,83\ \ [3,4]\,62\ \ [4,4]\,9$	"
$[1,5]\,45\ \ [2,5]\,32\ \ [3,5]\,26\ \ [4,5]\,85\ \ [5,5]\,6$	"
$[1,6]\,94\ \ [2,6]\,94\ \ [3,6]\,69\ \ [4,6]\,59\ \ [5,6]\,93\ \ [6,6]\,5;$	"
end;	Fine istanza

Tabella B.2: La tabella rappresenta la codifica di una possibile istanza in formato AMPL basata sul modello in B.1

B.2 Il file dei dati *.dat*

Una volta definito il modello del problema è possibile specificare le istanze sotto forma di modello dati. Nella tabella B.2 è riportato un file dati corrispondente ad una istanza del problema MEWCMC, con $n = 6$ vertici e $m = 3$ classi. Ogni classe V_k, $k \in \{1, 2, 3\}$ ha cardinalità $c_k = \frac{n}{m} = 3$. Dopo aver deciso l'appartenenza dei vertici ad ogni classe vine specificata la matrice dei pesi W in forma triangolare. Sulla diagonale si possono leggere i pesi associati ai vertici.

Risolvendo il modello B.1 con i dati forniti in B.2 si ottiene la clique M di peso massimo formata dai vertici $M = \{v_2, v_4, v_6\}$. Il valore ottimo corrispondente alla soluzione trovata è di $z * (M) = 255$.

La separazione del modello dai dati ha il vantaggio di poter riutilizzare lo stesso modello con diverse istanze del problema. Inoltre è possibile decidere quale solutore utilizzare per determinare la soluzione ottima dell'istanza fornita. Nel nostro caso abbiamo deciso di utilizzare il solutore commerciale *ILOG Cplex 11.2* per confrontare le prestazioni del nostro algoritmo di Branch and Bound.

Appendice C

Risultati degli esperimenti computazionali

Dio non solo gioca a dadi, ma qualche volta
tira i dadi dove non possono essere visti.
– Stephen Hawking

In questo capitolo riportiamo le tabelle che mostrano nel dettaglio i risultati delle simulazioni in un tempo limite di 1 ora. La campagna di test è stata condotta con la versione dell'algoritmo $SDP_{B\&B}$ completa. I risultati per ogni tipo di istanze sono state create due tabelle. Nella prima sono riportati i valori ottimi trovati e le informazioni al nodo radice; mentre nell'altra contiene le informazioni riguardanti il tempo di calcolo totale e l'esplorazione dell'albero delle soluzioni.

Le istanze del benchmark riportate $S1$ e $S2$ sono state così divise:

- $S1(A)$: tabelle C.1, C.2.
- $S1(B)$: tabelle C.3, C.4.
- $S1(C)$: tabelle C.5, C.6.
- $S2(B1)$: tabelle C.7, C.8.
- $S2(B2)$: tabelle C.9, C.10.
- $S2(B3)$: tabelle C.11, C.12.
- $S2(B4)$: tabelle C.13, C.14.

Per quanto riguarda la prima tabella di ogni tipo di istanza sono riportati nell'ordine: il nome dell'istanza, il valore ottimo o il valore della migliore soluzione primale, se la simulazione non termina nei 3600 secondi ($BestKnown$), il bound primale (PB) e duale (DB) al nodo radice col relativo tempo di calcolo ($T(s)$) e la distanza in percentuale ($Gap\,\%$) dei bound primale (PB) e duale

(DB) al nodo radice, dal valore ottimo.

La seconda tabella mostra nell'ordine: il nome dell'istanza, le informazioni relative al nodo in cui è stata trovata la migliore soluzione primale (*Best node*), rappresentate dal numero seriale del nodo (#) e la profondità nell'albero ad esso riferita (*depth*). Il campo successivo indica il numero totale di nodi esplorati (*Explored nodes*) e il massimo livello di profondità raggiunto durante l'esplorazione (*Max depth*). L'ultimo campo indica la distanza in percentuale (*gap %*) tra il miglior bound primale con il bound duale più lasco conosciuto, ovvero $gap = \frac{(DB-PB)}{PB}$.

Classe $S1(A)$		*Root*			*Gap* %	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
01a_n30c2	354	354	354.0	0.00	0.00	0.00
02a_n30c5	644	644	644.0	0.00	0.00	0.00
03a_n30c10	1,020	1,020	1,020.0	0.00	0.00	0.00
04a_n40c2	1,331	1,331	1,331.0	0.00	0.00	0.00
05a_n40c5	1,546	1,546	1,556.8	0.12	0.00	0.70
06a_n40c10	2,087	2,087	2,101.9	0.14	0.00	0.71
07a_n50c2	2,601	2,601	2,601.0	0.06	0.00	0.00
08a_n50c5	2,623	2,623	2,631.9	0.23	0.00	0.34
09a_n50c10	4,099	4,099	4,109.2	0.10	0.00	0.25
10a_n60c2	5,476	5,476	5,494.0	0.10	0.00	0.33
11a_n60c5	8,017	8,017	8,026.8	0.17	0.00	0.12
12a_n60c10	8,468	8,468	8,468.0	0.02	0.00	0.00
13a_n70c2	10,362	10,362	10,379.4	0.27	0.00	0.17
14a_n70c5	14,387	14,387	14,392.6	0.07	0.00	0.04
15a_n70c10	25,340	25,340	25,340.4	0.12	0.00	0.00
16a_n80c2	32,522	32,522	32,522.0	0.14	0.00	0.00
17a_n80c5	43,573	43,573	43,575.5	0.22	0.00	0.01
18a_n80c10	57,297	57,297	57,297.5	0.27	0.00	0.00
19a_n100c2	370	370	370.0	0.01	0.00	0.00
20a_n100c5	378	378	378.0	0.02	0.00	0.00
21a_n100c10	724	724	724.0	0.01	0.00	0.00
22a_n100c20	704	704	704.0	0.02	0.00	0.00
23a_n100c25	1,059	1,059	1,080.1	0.34	0.00	1.99
24a_n120c2	1,096	1,096	1,120.6	0.67	0.00	2.25
25a_n120c5	1,089	1,087	1,130.8	1.01	0.18	3.84
26a_n120c10	4,558	4,558	4,575.3	0.44	0.00	0.38
27a_n120c12	4,428	4,428	4,450.3	0.62	0.00	0.50
28a_n120c24	4,461	4,461	4,480.0	1.05	0.00	0.43
29a_n120c30	6,155	6,155	6,158.3	0.72	0.00	0.05
30a_n120c40	10,208	10,208	10,215.2	1.35	0.00	0.07
31a_n150c2	16,556	16,556	16,565.0	0.48	0.00	0.05
32a_n150c5	24,038	24,038	24,055.8	0.80	0.00	0.07
33a_n150c10	37,948	37,948	37,960.0	1.42	0.00	0.03
34a_n150c15	95,302	95,302	95,306.6	0.52	0.00	0.00
35a_n150c30	138,265	138,265	138,270.0	0.72	0.00	0.00
36a_n150c50	211,298	211,298	211,299.4	1.41	0.00	0.00
37a_n200c2	384	384	384.0	0.05	0.00	0.00
38a_n200c5	370	370	370.0	0.07	0.00	0.00
39a_n200c10	712	712	737.2	3.12	0.00	3.54
40a_n200c20	728	728	744.0	5.21	0.00	2.20
41a_n200c40	1,167	1,167	1,183.9	2.90	0.00	1.45
42a_n200c50	1,121	1,121	1,141.6	4.87	0.00	1.84
43a_n240c2	1,167	1,167	1,199.6	8.86	0.00	2.79
44a_n240c5	4,415	4,415	4,437.9	2.52	0.00	0.52
45a_n240c10	4,805	4,805	4,827.4	4.14	0.00	0.47
46a_n240c24	4,660	4,660	4,678.3	7.85	0.00	0.39
47a_n240c48	17,670	17,670	17,680.1	2.58	0.00	0.06
48a_n240c60	24,409	24,409	24,418.3	5.10	0.00	0.04
49a_n240c80	40,505	40,505	40,524.7	9.89	0.00	0.05
50a_n300c2	66,395	66,395	66,400.1	2.87	0.00	0.01
51a_n300c5	95,876	95,876	95,881.9	5.23	0.00	0.01
52a_n300c10	151,924	151,924	151,927.9	9.72	0.00	0.00
53a_n300c30	374,931	374,931	374,932.7	2.96	0.00	0.00
54a_n300c60	528,680	528,680	528,680.2	5.16	0.00	0.00
55a_n300c100	835,725	835,725	835,728.1	9.89	0.00	0.00

Tabella C.1: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S1(A)$ al nodo radice in un tempo limite di 1 ora.

Classe $S1(A)$	*Best node*		*Explored*	*Max*		
Istanza	#	*depth*	*nodes*	*dept*	$Time(s)$	$Gap\%$
01a_n30c2	0	0	1	0	0.02	0.00
02a_n30c5	0	0	1	0	0.00	0.00
03a_n30c10	0	0	1	0	0.00	0.00
04a_n40c2	0	0	3	1	0.18	0.00
05a_n40c5	0	0	1	0	0.06	0.00
06a_n40c10	0	0	1	0	0.00	0.00
07a_n50c2	0	0	3	1	0.25	0.00
08a_n50c5	0	0	7	3	0.43	0.00
09a_n50c10	0	0	1	0	0.00	0.00
10a_n60c2	0	0	1	0	0.14	0.00
11a_n60c5	0	0	17	6	1.22	0.00
12a_n60c10	0	0	3	1	0.17	0.00
13a_n70c2	0	0	5	2	0.76	0.00
14a_n70c5	0	0	17	7	1.92	0.00
15a_n70c10	0	0	7	3	0.62	0.00
16a_n80c2	0	0	3	1	0.67	0.00
17a_n80c5	0	0	13	4	2.13	0.00
18a_n80c10	0	0	15	5	1.83	0.00
19a_n100c2	0	0	3	1	1.22	0.00
20a_n100c5	0	0	7	2	2.37	0.00
21a_n100c10	0	0	15	7	3.58	0.00
22a_n100c20	0	0	9	4	1.32	0.00
23a_n100c25	0	0	1	0	0.01	0.00
24a_n120c2	0	0	3	1	1.96	0.00
25a_n120c5	0	0	29	5	14.14	0.00
26a_n120c10	0	0	5	2	2.60	0.00
27a_n120c12	0	0	43	10	14.64	0.00
28a_n120c24	0	0	15	4	4.16	0.00
29a_n120c30	0	0	1	0	0.02	0.00
30a_n120c40	0	0	1	0	0.01	0.00
31a_n150c2	0	0	3	1	3.71	0.00
32a_n150c5	0	0	7	3	6.52	0.00
33a_n150c10	0	0	9	4	8.08	0.00
34a_n150c15	0	0	35	8	22.16	0.00
35a_n150c30	6	3	47	7	22.31	0.00
36a_n150c50	0	0	1	0	0.02	0.00
37a_n200c2	0	0	5	2	12.33	0.00
38a_n200c5	0	0	5	2	11.47	0.00
39a_n200c10	0	0	9	4	19.27	0.00
40a_n200c20	0	0	43	10	62.63	0.00
41a_n200c40	0	0	13	5	18.88	0.00
42a_n200c50	0	0	17	4	17.52	0.00
43a_n240c2	0	0	3	1	12.84	0.00
44a_n240c5	0	0	3	1	12.88	0.00
45a_n240c10	0	0	15	7	54.58	0.00
46a_n240c24	0	0	25	9	66.60	0.00
47a_n240c48	0	0	29	5	70.92	0.00
48a_n240c60	0	0	19	4	32.50	0.00
49a_n240c80	0	0	1	0	0.05	0.00
50a_n300c2	0	0	3	1	27.38	0.00
51a_n300c5	0	0	3	1	26.84	0.00
52a_n300c10	0	0	19	6	138.99	0.00
53a_n300c30	0	0	29	8	148.84	0.00
54a_n300c60	0	0	85	7	356.58	0.00
55a_n300c100	0	0	1	0	0.07	0.00

Tabella C.2: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S1(A)$ in un tempo limite di 1 ora.

Classe $S1(B)$		*Root*			*Gap* %	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
01b_n30c2	6,078	6,078	6,078.0	0.01	0.00	0.00
02b_n30c5	1,188	1,188	1,188.0	0.00	0.00	0.00
03b_n30c10	295	295	295.0	0.00	0.00	0.00
04b_n40c2	11,157	11,157	11,280.3	0.05	0.00	1.10
05b_n40c5	2,117	2,117	2,117.0	0.05	0.00	0.00
06b_n40c10	557	557	557.0	0.00	0.00	0.00
07b_n50c2	17,132	17,132	17,319.8	0.09	0.00	1.10
08b_n50c5	3,229	3,229	3,547.5	0.06	0.00	9.86
09b_n50c10	871	871	871.0	0.00	0.00	0.00
10b_n60c2	25,106	25,106	25,183.8	0.12	0.00	0.31
11b_n60c5	4,649	4,649	5,113.6	0.08	0.00	9.99
12b_n60c10	1,235	1,233	1,475.1	0.07	0.16	19.44
13b_n70c2	34,168	34,168	34,370.4	0.10	0.00	0.59
14b_n70c5	6,132	6,128	6,738.5	0.15	0.07	9.89
15b_n70c10	1,735	1,735	2,048.6	0.11	0.00	18.07
16b_n80c2	43,437	43,437	43,633.3	0.23	0.00	0.45
17b_n80c5	7,829	7,829	8,644.6	0.18	0.00	10.42
18b_n80c10	2,200	2,200	2,713.1	0.15	0.00	23.32
19b_n100c2	67,876	67,876	68,393.5	0.38	0.00	0.76
20b_n100c5	12,408	12,408	13,553.6	0.35	0.00	9.23
21b_n100c10	3,451	3,446	4,348.0	0.30	0.14	25.99
22b_n100c20	913	913	1,018.8	0.32	0.00	11.58
23b_n100c25	601	601	601.0	0.01	0.00	0.00
24b_n120c2	95,598	95,598	96,193.9	0.67	0.00	0.62
25b_n120c5	17,454	17,454	18,928.4	0.59	0.00	8.45
26b_n120c10	4,877	4,870	6,302.0	0.55	0.14	29.22
27b_n120c12	3,579	3,579	4,369.4	0.49	0.00	22.09
28b_n120c24	945	945	1,020.3	0.45	0.00	7.96
29b_n120c30	596	596	623.5	0.60	0.00	4.61
30b_n120c40	316	316	316.0	0.00	0.00	0.00
31b_n150c2	150,187	150,187	150,990.7	1.12	0.00	0.54
32b_n150c5	27,304	27,304	29,270.5	0.99	0.00	7.20
33b_n150c10	7,531	7,511	9,474.4	1.03	0.27	25.80
34b_n150c15	3,503	3,503	4,395.4	0.97	0.00	25.48
35b_n150c30	947	931	1,024.5	0.77	1.69	8.18
36b_n150c50	323	323	323.0	0.03	0.00	0.00
37b_n200c2	267,583	267,583	269,223.1	2.11	0.00	0.61
38b_n200c5	47,458	47,458	50,496.7	2.14	0.00	6.40
39b_n200c10	13,326	13,326	16,170.0	1.87	0.00	21.34
40b_n200c20	3,592	3,592	4,460.8	1.93	0.00	24.19
41b_n200c40	965	965	1,028.9	2.10	0.00	6.62
42b_n200c50	606	586	628.0	2.26	3.30	3.63
43b_n240c2	382,159	382,159	384,400.0	3.40	0.00	0.59
44b_n240c5	66,563	66,563	71,275.2	3.44	0.00	7.08
45b_n240c10	18,719	18,719	22,509.6	3.47	0.00	20.25
46b_n240c24	3,654	3,654	4,468.7	3.15	0.00	22.30
47b_n240c48	980	980	1,029.9	3.27	0.00	5.09
48b_n240c60	610	609	630.3	4.36	0.16	3.33
49b_n240c80	320	320	320.0	0.04	0.00	0.00
50b_n300c2	593,195	593,195	596,923.8	9.49	0.00	0.63
51b_n300c5	104,703	104,703	110,284.6	6.13	0.00	5.33
52b_n300c10	28,528	28,528	34,088.7	5.66	0.00	19.49
53b_n300c30	3,716	3,716	4,498.3	5.33	0.00	21.05
54b_n300c60	983	966	1,039.3	6.94	1.73	5.72
55b_n300c100	323	323	323.0	0.06	0.00	0.00

Tabella C.3: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S1(B)$ al nodo radice in un tempo limite di 1 ora.

Classe $S1(B)$	*Best node*		*Explored*	*Max*		
Istanza	#	*depth*	*nodes*	*dept*	*Time*(*s*)	*Gap*%
01b_n30c2	0	0	1	0	0.01	0.00
02b_n30c5	0	0	1	0	0.00	0.00
03b_n30c10	0	0	1	0	0.00	0.00
04b_n40c2	0	0	7	2	0.63	0.00
05b_n40c5	0	0	1	0	0.05	0.00
06b_n40c10	0	0	1	0	0.00	0.00
07b_n50c2	0	0	41	7	2.06	0.00
08b_n50c5	0	0	55	6	2.56	0.00
09b_n50c10	0	0	1	0	0.00	0.00
10b_n60c2	0	0	19	5	1.31	0.00
11b_n60c5	0	0	365	10	19.64	0.00
12b_n60c10	3	2	5	2	0.23	0.00
13b_n70c2	0	0	65	8	5.97	0.00
14b_n70c5	210	7	1273	12	96.77	0.00
15b_n70c10	0	0	51	5	3.12	0.00
16b_n80c2	0	0	49	9	6.26	0.00
17b_n80c5	0	0	3141	13	333.34	0.00
18b_n80c10	0	0	587	10	45.38	0.00
19b_n100c2	0	0	361	11	85.17	0.00
20b_n100c5	0	0	5187	16	911.26	0.00
21b_n100c10	1556	10	11947	15	1646.35	0.00
22b_n100c20	0	0	19	4	2.29	0.00
23b_n100c25	0	0	1	0	0.01	0.00
24b_n120c2	0	0	639	16	239.04	0.00
25b_n120c5	0	0	5094	14	-	0.94
26b_n120c10	208	7	4908	15	-	2.85
27b_n120c12	0	0	10427	15	-	0.40
28b_n120c24	0	0	47	6	8.19	0.00
29b_n120c30	0	0	3	1	0.62	0.00
30b_n120c40	0	0	1	0	0.00	0.00
31b_n150c2	0	0	3649	19	2479.65	0.00
32b_n150c5	0	0	2912	13	-	1.88
33b_n150c10	1048	10	2854	13	-	7.38
34b_n150c15	0	0	3341	13	-	13.26
35b_n150c30	35	7	139	7	41.01	0.00
36b_n150c50	0	0	1	0	0.03	0.00
37b_n200c2	0	0	1092	12	-	0.13
38b_n200c5	0	0	1432	12	-	3.45
39b_n200c10	0	0	1346	11	-	9.69
40b_n200c20	0	0	1622	14	-	20.07
41b_n200c40	0	0	597	10	412.41	0.00
42b_n200c50	16	5	37	5	30.00	0.00
43b_n240c2	0	0	726	11	-	0.30
44b_n240c5	0	0	898	11	-	4.76
45b_n240c10	0	0	821	11	-	12.63
46b_n240c24	0	0	912	13	-	19.87
47b_n240c48	0	0	1499	11	1757.38	0.00
48b_n240c60	60	6	69	6	75.36	0.00
49b_n240c80	0	0	1	0	0.04	0.00
50b_n300c2	0	0	377	10	-	0.40
51b_n300c5	0	0	466	10	-	3.71
52b_n300c10	0	0	391	9	-	14.10
53b_n300c30	0	0	483	13	-	19.08
54b_n300c60	1000	11	651	12	-	3.22
55b_n300c100	0	0	1	0	0.06	0.00

Tabella C.4: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S1(B)$ in un tempo limite di 1 ora.

Classe $S1(C)$		*Root*			*Gap* %	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
01c_n30c2	673	673	673.0	0.00	0.00	0.00
02c_n30c5	342	342	342.0	0.00	0.00	0.00
03c_n30c10	128	128	128.0	0.00	0.00	0.00
04c_n40c2	1,628	1,610	1,665.1	0.05	1.11	2.28
05c_n40c5	604	604	604.0	0.04	0.00	0.00
06c_n40c10	245	245	245.0	0.00	0.00	0.00
07c_n50c2	1,625	1,625	1,757.5	0.07	0.00	8.15
08c_n50c5	898	898	1,253.2	0.06	0.00	39.55
09c_n50c10	357	357	357.0	0.00	0.00	0.00
10c_n60c2	2,554	2,554	2,717.6	0.11	0.00	6.40
11c_n60c5	1,154	1,154	1,665.2	0.08	0.00	44.29
12c_n60c10	532	532	696.5	0.08	0.00	30.92
13c_n70c2	2,873	2,873	3,175.9	0.14	0.00	10.54
14c_n70c5	1,473	1,473	2,106.6	0.14	0.00	43.02
15c_n70c10	589	578	971.6	0.11	1.87	64.95
16c_n80c2	4,507	4,507	4,683.7	0.20	0.00	3.92
17c_n80c5	1,898	1,898	2,596.5	0.17	0.00	36.80
18c_n80c10	793	793	1,274.6	0.14	0.00	60.74
19c_n100c2	6,475	6,475	6,816.4	0.36	0.00	5.27
20c_n100c5	2,602	2,602	3,739.5	0.31	0.00	43.72
21c_n100c10	1,152	1,152	2,029.7	0.25	0.00	76.19
22c_n100c20	422	422	498.8	0.25	0.00	18.19
23c_n100c25	276	276	276.0	0.02	0.00	0.00
24c_n120c2	7,427	7,398	8,182.3	0.59	0.39	10.17
25c_n120c5	3,505	3,505	4,921.9	0.43	0.00	40.43
26c_n120c10	1,546	1,540	2,875.1	0.49	0.39	85.97
27c_n120c12	1,179	1,179	2,053.7	0.43	0.00	74.19
28c_n120c24	404	394	499.3	0.44	2.48	23.58
29c_n120c30	263	263	301.7	0.42	0.00	14.70
30c_n120c40	156	156	156.0	0.01	0.00	0.00
31c_n150c2	8,627	8,627	9,506.1	0.90	0.00	10.19
32c_n150c5	4,839	4,839	6,865.5	0.84	0.00	41.88
33c_n150c10	2,284	2,284	4,088.5	0.92	0.00	79.00
34c_n150c15	1,220	1,220	2,116.2	0.66	0.00	73.46
35c_n150c30	438	438	507.9	0.73	0.00	15.95
36c_n150c50	156	156	156.0	0.02	0.00	0.00
37c_n200c2	14,142	13,933	15,873.7	2.04	1.48	12.24
38c_n200c5	7,902	7,902	10,974.4	1.75	0.00	38.88
39c_n200c10	3,523	3,523	6,391.6	1.77	0.00	81.42
40c_n200c20	1,315	1,315	2,147.6	1.58	0.00	63.31
41c_n200c40	438	438	513.4	1.65	0.00	17.21
42c_n200c50	290	282	313.0	2.19	2.76	7.93
43c_n240c2	18,066	18,066	20,317.3	3.63	0.00	12.46
44c_n240c5	10,205	10,205	14,421.4	2.66	0.00	41.32
45c_n240c10	4,769	4,769	8,713.2	2.94	0.00	82.70
46c_n240c24	1,348	1,348	2,169.7	2.43	0.00	60.96
47c_n240c48	459	442	511.0	2.85	3.70	11.33
48c_n240c60	284	283	313.3	2.70	0.35	10.33
49c_n240c80	160	160	160.0	0.05	0.00	0.00
50c_n300c2	27,808	27,808	30,931.9	9.50	0.00	11.23
51c_n300c5	15,049	15,049	20,712.6	5.32	0.00	37.63
52c_n300c10	6,479	6,479	11,907.7	5.88	0.00	83.79
53c_n300c30	1,401	1,401	2,192.6	4.22	0.00	56.50
54c_n300c60	449	449	513.9	4.68	0.00	14.45
55c_n300c100	163	163	163.0	0.06	0.00	0.00

Tabella C.5: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S1(C)$ al nodo radice in un tempo limite di 1 ora.

Classe $S1(C)$	*Best node*		*Explored*	*Max*		
Istanza	#	*depth*	*nodes*	*dept*	*Time*(*s*)	*Gap*%
01c_n30c2	0	0	1	0	0.00	0.00
02c_n30c5	0	0	1	0	0.00	0.00
03c_n30c10	0	0	1	0	0.00	0.00
04c_n40c2	1	1	7	2	0.62	0.00
05c_n40c5	0	0	1	0	0.04	0.00
06c_n40c10	0	0	1	0	0.00	0.00
07c_n50c2	0	0	33	7	1.85	0.00
08c_n50c5	0	0	61	6	2.84	0.00
09c_n50c10	0	0	1	0	0.00	0.00
10c_n60c2	0	0	37	5	2.40	0.00
11c_n60c5	0	0	363	9	20.81	0.00
12c_n60c10	0	0	5	2	0.24	0.00
13c_n70c2	0	0	103	10	9.68	0.00
14c_n70c5	0	0	1163	14	94.01	0.00
15c_n70c10	51	6	59	6	3.52	0.00
16c_n80c2	0	0	29	7	3.82	0.00
17c_n80c5	0	0	1119	16	117.30	0.00
18c_n80c10	0	0	567	10	46.30	0.00
19c_n100c2	0	0	103	8	23.94	0.00
20c_n100c5	0	0	12633	18	2394.81	0.00
21c_n100c10	0	0	8977	14	1303.92	0.00
22c_n100c20	0	0	17	4	1.99	0.00
23c_n100c25	0	0	1	0	0.02	0.00
24c_n120c2	34	4	2193	14	848.89	0.00
25c_n120c5	0	0	4667	15	-	5.08
26c_n120c10	2254	11	4669	14	-	6.52
27c_n120c12	0	0	5899	15	-	10.24
28c_n120c24	31	4	43	5	8.04	0.00
29c_n120c30	0	0	3	1	0.45	0.00
30c_n120c40	0	0	1	0	0.01	0.00
31c_n150c2	0	0	2905	16	2055.38	0.00
32c_n150c5	0	0	2670	14	-	14.25
33c_n150c10	0	0	2743	13	-	19.79
34c_n150c15	0	0	3225	13	-	29.75
35c_n150c30	0	0	139	7	42.85	0.00
36c_n150c50	0	0	1	0	0.02	0.00
37c_n200c2	570	9	1040	12	-	3.93
38c_n200c5	0	0	1328	12	-	19.63
39c_n200c10	0	0	1305	11	-	39.45
40c_n200c20	0	0	1563	14	-	51.35
41c_n200c40	0	0	597	10	423.57	0.00
42c_n200c50	5	3	11	3	8.55	0.00
43c_n240c2	0	0	639	11	-	5.75
44c_n240c5	0	0	791	11	-	27.13
45c_n240c10	0	0	781	11	-	50.92
46c_n240c24	0	0	893	13	-	52.91
47c_n240c48	163	9	1515	11	1796.43	0.00
48c_n240c60	46	6	79	6	85.59	0.00
49c_n240c80	0	0	1	0	0.05	0.00
50c_n300c2	0	0	336	10	-	6.74
51c_n300c5	0	0	430	10	-	26.92
52c_n300c10	0	0	391	9	-	61.09
53c_n300c30	0	0	473	12	-	50.18
54c_n300c60	0	0	684	12	-	9.05
55c_n300c100	0	0	1	0	0.06	0.00

Tabella C.6: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S1(C)$ in un tempo limite di 1 ora.

Classe $S2(B1)$		*Root*			*Gap %*	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
1-30-2	29,100	29099.51	29099.51	0.02	0	0
1-30-3	14,111	14110.9	14110.9	0.02	0	0
1-30-5	7,349	7348.64	7348.64	0	0	0
1-32-2	28,335	28334.88	28334.88	0.03	0	0
1-32-4	13,018	13018.49	13018.49	0.01	0	0
1-33-3	18,738	18737.53	18737.53	0.06	0	0
1-34-2	32,701	32700.99	32700.99	0.07	0	0
1-35-5	10,017	10016.96	10016.96	0.01	0	0
1-36-2	35,621	35620.85	35620.85	0.14	0	0
1-36-3	19,880	19880.34	20548.61	0.03	0	3.36
1-36-4	15,323	15322.77	15322.77	0.04	0	0
1-38-2	42,041	42041.18	42307.43	0.06	0	0.63
1-39-3	23,588	23587.62	24546.16	0.03	0	4.06
1-40-2	46,863	46862.66	47282.61	0.04	0	0.9
1-40-4	17,973	17594.53	19022.6	0.03	2.11	5.84
1-40-5	12,845	12845.25	12845.25	0.05	0	0
1-42-2	51,379	51378.63	51936.74	0.08	0	1.09
1-42-3	26,900	26900.49	27393.33	0.05	0	1.83
1-44-2	54,909	54908.71	55341.03	0.06	0	0.79
1-44-4	22,445	22444.76	23394.02	0.04	0	4.23
1-45-3	30,437	29947.45	31576	0.06	1.61	3.74
1-45-5	15,023	15022.56	15883.98	0.04	0	5.73
1-46-2	59,279	59209.89	59857.52	0.07	0.12	0.98
1-48-2	61,849	61848.77	62014.11	0.06	0	0.27
1-48-3	34,400	34400.19	35227.08	0.09	0	2.4
1-48-4	22,147	22032.95	22773.47	0.05	0.51	2.83
1-50-2	63,849	63775.08	64963.17	0.07	0.12	1.75
1-50-5	17,177	16888.02	18409.11	0.05	1.68	7.17
1-51-3	35,495	35184.24	36461.75	0.07	0.88	2.72
1-52-2	76,041	75576.43	77028.89	0.08	0.61	1.3
1-52-4	27,811	27811.49	28885.06	0.1	0	3.86
1-54-2	80,605	80416.9	81130.33	0.09	0.23	0.65
1-54-3	40,655	40655.03	41733.52	0.08	0	2.65
1-55-5	19,309	19095.01	21830.83	0.09	1.11	13.06
1-56-4	30,025	29781.52	32306.97	0.1	0.81	7.6
1-60-5	25,080	24612.24	26463.79	0.14	1.87	5.52
1-65-5	24,944	24723.34	27569.64	0.16	0.89	10.53

Tabella C.7: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S2(B1)$ al nodo radice in un tempo limite di 1 ora.

Classe $S2(B1)$	*Best node*		*Explored*	*Max*	
Istanza	#	*depth*	*nodes*	*dept*	$Time(s)$
1-30-2	0	0	1	0	0.02
1-30-3	0	0	1	0	0.02
1-30-5	0	0	1	0	0.00
1-32-2	0	0	1	0	0.03
1-32-4	0	0	1	0	0.01
1-33-3	0	0	1	0	0.06
1-34-2	0	0	1	0	0.07
1-35-5	0	0	1	0	0.01
1-36-2	0	0	1	0	0.14
1-36-3	0	0	3	1	0.16
1-36-4	0	0	1	0	0.04
1-38-2	0	0	3	1	0.32
1-39-3	0	0	9	3	0.56
1-40-2	0	0	7	2	0.61
1-40-4	6	2	7	2	0.29
1-40-5	0	0	1	0	0.05
1-42-2	0	0	11	3	0.84
1-42-3	0	0	9	3	0.40
1-44-2	0	0	15	4	0.99
1-44-4	0	0	17	4	0.71
1-45-3	12	4	45	6	2.51
1-45-5	0	0	11	5	0.45
1-46-2	1	1	13	3	0.42
1-48-2	0	0	7	3	0.33
1-48-3	0	0	19	5	0.90
1-48-4	4	2	15	4	0.54
1-50-2	4	2	45	6	1.86
1-50-5	16	5	31	5	1.21
1-51-3	12	3	37	6	1.66
1-52-2	8	3	39	6	1.73
1-52-4	0	0	35	6	1.62
1-54-2	2	1	17	5	0.86
1-54-3	0	0	25	5	1.18
1-55-5	73	5	131	7	6.32
1-56-4	33	5	201	9	10.42
1-60-5	9	3	43	6	2.15
1-65-5	309	8	499	11	32.21

Tabella C.8: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S2(B1)$ in un tempo limite di 1 ora.

Classe $S2(B2)$		*Root*			*Gap* %	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
2-30-2	21,214	21214.5	21,214.5	0.01	0.00	0.00
2-30-3	8,859	8859.3	8,859.3	0.01	0.00	0.00
2-30-5	4,152	4151.8	4,151.8	0.00	0.00	0.00
2-32-2	21,722	21722.3	21,722.3	0.02	0.00	0.00
2-32-4	8,368	8368.0	8,368.0	0.00	0.00	0.00
2-33-3	12,417	12416.8	12,416.8	0.04	0.00	0.00
2-34-2	24,841	24840.8	24,840.8	0.05	0.00	0.00
2-35-5	7,396	7396.0	7,396.0	0.00	0.00	0.00
2-36-2	27,045	27044.5	27,044.5	0.11	0.00	0.00
2-36-3	13,677	13677.5	14,006.2	0.03	0.00	2.40
2-36-4	9,827	9826.8	9,826.8	0.04	0.00	0.00
2-38-2	32,327	32326.8	32,625.0	0.05	0.00	0.92
2-39-3	16,993	16992.6	17,889.8	0.03	0.00	5.28
2-40-2	34,641	34641.2	35,537.2	0.04	0.00	2.59
2-40-4	13,486	13486.3	14,593.7	0.04	0.00	8.21
2-40-5	7,488	7488.2	7,488.2	0.04	0.00	0.00
2-42-2	38,707	38507.3	39,571.0	0.05	0.52	2.23
2-42-3	19,255	19255.5	19,609.4	0.04	0.00	1.84
2-44-2	41,086	41085.9	41,640.5	0.04	0.00	1.35
2-44-4	14,975	14974.8	16,188.0	0.04	0.00	8.10
2-45-3	21,901	21900.5	22,972.4	0.05	0.00	4.89
2-45-5	9,656	9648.3	10,922.0	0.04	0.08	13.12
2-46-2	46,964	46964.4	47,766.9	0.07	0.00	1.71
2-48-2	49,029	49028.8	49,805.6	0.06	0.00	1.58
2-48-3	23,768	23767.66	25281.1	0.06	0.00	6.37
2-48-4	15,210	15209.97	15875.63	0.05	0.00	4.38
2-50-2	51,558	51558.3	52,352.6	0.08	0.00	1.54
2-50-5	11,810	11810.4	13,223.4	0.06	0.00	11.96
2-51-3	26,149	26148.7	26,907.5	0.07	0.00	2.90
2-52-2	63,300	63300.3	64,136.5	0.08	0.00	1.32
2-52-4	19,732	19732.4	21,293.5	0.09	0.00	7.91
2-54-2	66,217	66217.2	66,966.2	0.08	0.00	1.13
2-54-3	30,683	30683.0	32,054.7	0.07	0.00	4.47
2-55-5	14,347	14347.1	16,239.2	0.08	0.00	13.19
2-56-4	22,981	22695.9	24,869.1	0.14	1.24	8.22
2-60-5	17,794	17793.9	19,346.2	0.11	0.00	8.72
2-65-5	19,466	19465.7	21,120.2	0.13	0.00	8.50
2-70-5	23,603	23602.5	25,918.7	0.19	0.00	9.81

Tabella C.9: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S2(B2)$ al nodo radice in un tempo limite di 1 ora.

Classe $S2(B2)$	*Best node*		*Explored*	*Max*	
Istanza	#	*depth*	*nodes*	*dept*	$Time(s)$
2-30-2	0	0	1	0	0.01
2-30-3	0	0	1	0	0.01
2-30-5	0	0	1	0	0.00
2-32-2	0	0	1	0	0.02
2-32-4	0	0	1	0	0.00
2-33-3	0	0	1	0	0.04
2-34-2	0	0	1	0	0.05
2-35-5	0	0	1	0	0.00
2-36-2	0	0	1	0	0.11
2-36-3	0	0	3	1	0.16
2-36-4	0	0	1	0	0.04
2-38-2	0	0	3	1	0.30
2-39-3	0	0	9	3	0.55
2-40-2	0	0	7	2	0.62
2-40-4	0	0	7	3	0.26
2-40-5	0	0	1	0	0.04
2-42-2	7	3	15	3	1.39
2-42-3	0	0	9	3	0.39
2-44-2	0	0	15	4	0.78
2-44-4	0	0	21	5	0.95
2-45-3	0	0	41	6	2.47
2-45-5	3	2	11	4	0.48
2-46-2	0	0	25	5	1.09
2-48-2	0	0	41	6	1.82
2-48-3	0	0	131	8	9.16
2-48-4	0	0	21	5	0.86
2-50-2	0	0	21	5	0.94
2-50-5	0	0	29	4	1.17
2-51-3	0	0	15	5	0.68
2-52-2	0	0	45	7	1.91
2-52-4	0	0	65	6	3.10
2-54-2	0	0	21	5	1.05
2-54-3	0	0	75	7	3.49
2-55-5	0	0	43	5	1.89
2-56-4	13	4	85	8	4.18
2-60-5	0	0	65	6	2.97
2-65-5	0	0	57	8	3.62
2-70-5	0	0	129	11	10.19

Tabella C.10: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S2(B2)$ in un tempo limite di 1 ora.

Classe $S2(B3)$		*Root*			*Gap %*	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
3-30-2	12,193	12193.4	12,193.4	0.01	0.00	0.00
3-30-3	7,301	7301.2	7,301.2	0.01	0.00	0.00
3-30-5	5,144	5144.4	5,144.4	0.00	0.00	0.00
3-32-2	8,634	8634.1	8,634.1	0.02	0.00	0.00
3-32-4	4,724	4724.3	4,724.3	0.01	0.00	0.00
3-33-3	10,006	10005.9	10,005.9	0.05	0.00	0.00
3-34-2	10,315	10315.3	10,315.3	0.06	0.00	0.00
3-35-5	7,206	7206.3	7,206.3	0.00	0.00	0.00
3-36-2	13,091	13091.2	13,091.2	0.11	0.00	0.00
3-36-3	9,375	9374.6	10,070.6	0.03	0.00	7.42
3-36-4	8,229	8229.2	8,229.2	0.04	0.00	0.00
3-38-2	11,149	11148.6	11,491.8	0.04	0.00	3.08
3-39-3	9,977	9968.7	10,728.0	0.04	0.08	7.53
3-40-2	11,440	11440.2	12,190.9	0.04	0.00	6.56
3-40-4	9,317	9115.7	9,904.4	0.04	2.16	6.31
3-40-5	6,822	6821.7	6,821.7	0.04	0.00	0.00
3-42-2	11,682	11239.0	12,536.0	0.06	3.79	7.31
3-42-3	13,693	13692.6	13,772.5	0.05	0.00	0.58
3-44-2	14,182	14181.9	14,478.4	0.05	0.00	2.09
3-44-4	8,014	8013.5	9,111.3	0.04	0.00	13.70
3-45-3	11,242	11242.3	12,421.1	0.06	0.00	10.48
3-45-5	8,803	8043.1	9,474.8	0.04	8.63	7.63
3-46-2	16,064	16064.1	16,531.3	0.06	0.00	2.91
3-48-2	14,644	14644.0	15,344.5	0.07	0.00	4.78
3-48-3	11,926	11734.1	13,006.9	0.06	1.61	9.07
3-48-4	10,283	10019.2	11,826.8	0.06	2.57	15.01
3-50-2	14,729	14526.6	16,240.8	0.07	1.37	10.27
3-50-5	11,492	11492.4	12,252.6	0.05	0.00	6.61
3-51-3	18,177	18176.5	18,709.5	0.07	0.00	2.93
3-52-2	20,685	20351.2	22,023.0	0.07	1.62	6.47
3-52-4	10,779	10778.7	12,675.2	0.09	0.00	17.60
3-54-2	16,053	16053.0	17,369.3	0.08	0.00	8.20
3-54-3	17,214	17214.0	18,629.3	0.10	0.00	8.22
3-55-5	11,949	11602.5	14,611.1	0.07	2.90	22.27
3-56-2	19,192	18859.4	21,451.9	0.11	1.73	11.78
3-56-4	11,640	11639.8	13,146.0	0.10	0.00	12.94
3-57-3	18,421	18420.9	19,970.4	0.09	0.00	8.41
3-58-2	19,884	19708.7	20,258.8	0.10	0.88	1.89
3-60-2	27,726	27726.0	28,825.5	0.10	0.00	3.97
3-60-3	21,278	21277.9	23,086.0	0.10	0.00	8.50
3-60-4	13,928	13137.1	15,648.2	0.10	5.68	12.35
3-60-5	12,480	12480.3	13,681.7	0.10	0.00	9.63
3-63-3	21,725	21725.1	22,729.3	0.11	0.00	4.62
3-64-4	20,239	20157.5	22,483.8	0.11	0.40	11.09
3-65-5	12,459	12458.7	15,204.0	0.10	0.00	22.04
3-70-5	17,223	17039.7	20,024.5	0.15	1.06	16.27
3-75-5	19,845	19844.9	23,644.4	0.15	0.00	19.15
3-80-5	18,906	18734.9	23,016.6	0.17	0.91	21.74

Tabella C.11: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S2(B3)$ al nodo radice in un tempo limite di 1 ora.

Classe $S2(B3)$	*Best node*		*Explored*	*Max*	
Istanza	#	*depth*	*nodes*	*dept*	$Time(s)$
3-30-2	0	0	1	0	0.01
3-30-3	0	0	1	0	0.01
3-30-5	0	0	1	0	0.00
3-32-2	0	0	1	0	0.02
3-32-4	0	0	1	0	0.01
3-33-3	0	0	1	0	0.05
3-34-2	0	0	1	0	0.06
3-35-5	0	0	1	0	0.00
3-36-2	0	0	1	0	0.11
3-36-3	0	0	3	1	0.16
3-36-4	0	0	1	0	0.04
3-38-2	0	0	3	1	0.30
3-39-3	4	2	9	3	0.56
3-40-2	0	0	7	2	0.64
3-40-4	3	2	7	2	0.33
3-40-5	0	0	1	0	0.04
3-42-2	2	1	13	3	1.11
3-42-3	0	0	5	2	0.19
3-44-2	0	0	5	2	0.20
3-44-4	0	0	19	4	0.79
3-45-3	0	0	37	5	1.89
3-45-5	2	1	7	3	0.31
3-46-2	0	0	11	4	0.41
3-48-2	0	0	11	4	0.45
3-48-3	3	2	47	7	2.02
3-48-4	1	1	51	6	2.38
3-50-2	1	1	41	7	1.91
3-50-5	0	0	13	4	0.58
3-51-3	0	0	15	4	0.70
3-52-2	3	2	15	5	0.68
3-52-4	0	0	117	7	6.20
3-54-2	0	0	69	9	3.62
3-54-3	0	0	79	9	4.30
3-55-5	23	5	79	7	3.83
3-56-2	22	4	45	6	2.45
3-56-4	0	0	31	4	1.60
3-57-3	0	0	65	6	3.53
3-58-2	1	1	11	4	0.69
3-60-2	0	0	33	6	1.92
3-60-3	0	0	41	7	2.53
3-60-4	6	2	51	6	2.93
3-60-5	0	0	17	5	0.92
3-63-3	0	0	17	6	1.30
3-64-4	33	5	127	10	8.49
3-65-5	0	0	227	12	13.83
3-70-5	5	2	51	7	3.94
3-75-5	0	0	267	11	21.78
3-80-5	30	4	599	14	61.24

Tabella C.12: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S2(B3)$ in un tempo limite di 1 ora.

Classe $S2(B4)$		*Root*			*Gap* %	
Istanza	*Best Known*	*PB*	*DB*	$T(s)$	*PB*	*DB*
4-30-2	9,230	9229.7	9,229.7	0.01	0.00	0.00
4-30-3	5,348	5347.8	5,347.8	0.01	0.00	0.00
4-30-5	3,850	3850.3	3,850.3	0.00	0.00	0.00
4-32-2	7,146	7146.4	7,146.4	0.03	0.00	0.00
4-32-4	3,684	3683.6	3,683.6	0.01	0.00	0.00
4-33-3	8,443	8443.5	8,443.5	0.05	0.00	0.00
4-34-2	6,598	6598.2	6,598.2	0.07	0.00	0.00
4-35-5	3,803	3803.47	3803.47	0.01	0.00	0.00
4-36-2	10,103	10103.35	10103.35	0.15	0.00	0.00
4-36-3	5,787	5787.2	6,472.0	0.03	0.00	11.83
4-36-4	5,410	5410.1	5,410.1	0.03	0.00	0.00
4-38-2	9,898	9897.6	10,503.2	0.05	0.00	6.12
4-39-3	7,835	7834.7	8,618.5	0.04	0.00	10.00
4-40-2	11,120	11120.3	11,538.5	0.04	0.00	3.76
4-40-4	5,566	5565.9	7,300.5	0.04	0.00	31.17
4-40-5	4,523	4522.5	4,522.5	0.05	0.00	0.00
4-42-2	9,152	9152.4	9,976.0	0.06	0.00	9.00
4-42-3	9,182	9181.7	9,974.8	0.05	0.00	8.64
4-44-2	13,803	13802.9	14,077.8	0.04	0.00	1.99
4-44-4	7,071	7070.9	8,552.3	0.05	0.00	20.95
4-45-3	9,950	9950.2	11,422.5	0.05	0.00	14.80
4-45-5	5,463	5393.8	7,253.5	0.04	1.27	32.77
4-46-2	9,242	9241.8	10,130.2	0.05	0.00	9.61
4-48-2	10,747	10378.9	12,173.6	0.05	3.43	13.27
4-48-3	11,226	11226.2	12,363.5	0.06	0.00	10.13
4-48-4	8,991	8990.8	10,418.3	0.07	0.00	15.88
4-50-2	17,423	17156.9	19,178.8	0.09	1.53	10.08
4-50-5	7,751	7751.1	9,503.1	0.06	0.00	22.60
4-51-3	12,606	12605.7	12,910.3	0.09	0.00	2.42
4-52-2	16,938	16938.4	19,017.1	0.07	0.00	12.27
4-52-4	8,797	8797.3	11,149.9	0.06	0.00	26.74
4-54-2	17,442	17441.6	18,985.5	0.10	0.00	8.85
4-54-3	13,889	13888.8	15,228.0	0.07	0.00	9.64
4-55-5	10,033	10033.2	12,839.4	0.06	0.00	27.97
4-56-2	19,954	19954.1	20,736.4	0.08	0.00	3.92
4-56-4	9,349	9349.2	12,178.2	0.10	0.00	30.26
4-57-3	14,901	14901.2	16,532.0	0.08	0.00	10.94
4-58-2	19,072	19071.8	20,610.4	0.10	0.00	8.07
4-60-3	18,121	18120.7	20,287.7	0.08	0.00	11.96
4-60-4	10,652	10652.0	14,171.2	0.10	0.00	33.04
4-60-5	10,592	10592.3	12,689.8	0.06	0.00	19.80
4-64-4	15,256	15256.3	18,291.9	0.11	0.00	19.90
4-65-5	10,198	10198.2	14,061.6	0.10	0.00	37.88
4-70-5	12,915	12914.5	17,748.5	0.15	0.00	37.43
4-75-5	14,840	14839.8	18,828.2	0.15	0.00	26.88

Tabella C.13: Tabella che mostra i risultati delle simulazioni sulle singole istanze del benchmark $S2(B4)$ al nodo radice in un tempo limite di 1 ora.

Classe $S2(B4)$	*Best node*		*Explored*	*Max*	
Istanza	#	*depth*	*nodes*	*dept*	$Time(s)$
4-30-2	0	0	1	0	0.01
4-30-3	0	0	1	0	0.01
4-30-5	0	0	1	0	0.00
4-32-2	0	0	1	0	0.03
4-32-4	0	0	1	0	0.01
4-33-3	0	0	1	0	0.05
4-34-2	0	0	1	0	0.07
4-35-5	0	0	1	0	0.01
4-36-2	0	0	1	0	0.15
4-36-3	0	0	3	1	0.15
4-36-4	0	0	1	0	0.03
4-38-2	0	0	3	1	0.30
4-39-3	0	0	9	3	0.54
4-40-2	0	0	7	2	0.61
4-40-4	0	0	7	3	0.28
4-40-5	0	0	1	0	0.05
4-42-2	0	0	11	3	0.80
4-42-3	0	0	15	3	0.73
4-44-2	0	0	15	4	0.97
4-44-4	0	0	31	5	1.41
4-45-3	0	0	49	6	3.10
4-45-5	7	3	11	4	0.51
4-46-2	0	0	31	5	1.29
4-48-2	5	2	37	6	1.66
4-48-3	0	0	47	6	2.41
4-48-4	0	0	49	6	1.98
4-50-2	8	3	55	7	2.58
4-50-5	0	0	43	5	1.81
4-51-3	0	0	7	3	0.43
4-52-2	0	0	59	8	2.89
4-52-4	0	0	185	8	10.41
4-54-2	0	0	65	8	3.18
4-54-3	0	0	55	6	2.85
4-55-5	0	0	69	6	3.18
4-56-2	0	0	13	4	0.68
4-56-4	0	0	379	9	22.34
4-57-3	0	0	83	11	4.90
4-58-2	0	0	37	7	2.12
4-60-3	0	0	49	7	3.06
4-60-4	0	0	513	9	31.87
4-60-5	0	0	47	8	2.31
4-64-4	0	0	217	9	14.09
4-65-5	0	0	563	11	37.14
4-70-5	0	0	443	11	32.22
4-75-5	0	0	333	11	27.56

Tabella C.14: Tabella che mostra i tempi e le informazioni relative all'esplorazione dell'albero di ricerca sulle singole istanze del benchmark $S2(B4)$ in un tempo limite di 1 ora.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] Bahram Alidaee, Fred Glover, Gary Kochenberger, and Haibo Wang. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research*, 181(1):592–597, September 2007.

[2] Miguel F. Anjos and Miguel F. Anjos. An improved semidefinite programming relaxation for the satisfiability problem. *Math. Program*, 102:2004, 2002.

[3] Steven J. Benson and Yinyu Ye. Dsdp3: Dual-scaling algorithm for semidefinite programming. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2001.

[4] Steven J. Benson and Yinyu Ye. Dsdp5: Software for semidefinite programming. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2005.

[5] Brian Borchers. Csdp, a c library for semidefinite programming. 1997.

[6] Alberto Bosio. Un algoritmo di programmazione matematica per il "max edge weighted clique problem with multiple choice constraints". *Università degli stidi di Milano*, 2004/2005.

[7] Antonio Celebrin. Teoria della programmazione semidefinita. *Università degli studi di Padova*, 2004.

[8] Roberto Cordone. Appunti sul linguaggio di programmazione ampl. Ottobre 2001.

[9] Marcos R. Q. de Andrade, Paulo M. F. de Andrade, Simone L. Martins, and Alexandre Plastino. Grasp with path-relinking for the maximum diversity problem. In *WEA*, pages 558–569, 2005.

[10] Igor Dukanovic and Franz Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. *Math. Program.*, 109(2):345–365, 2007.

[11] Macambira E.M. The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations. *European Journal of Operational Research*, 123:346–371(26), 1 June 2000.

[12] Glover F. and Laguna M. Tabu search. *Kluwer Academic Publishers*, 1997.

[13] K. S. Dhir F. Glover. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

[14] Alberto Ghilardi. Algoritmi di branch and bound per il maximum diversity problem. *Università degli studi di Milano*, 2006/07.

[15] Hersh G McMillian C Glover F. Selecting subset of maximum diversity. *University of Colorado*, 1977.

[16] M. X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[17] M. X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[18] Didier Henrion and Jean bernard Lasserre. Gloptipoly: Global optimization over polynomials with matlab and sedumi. *ACM Trans. Math. Soft*, 29:165–194, 2002.

[19] Eun-Jong Hong and Tomas Lozano-Perez. Protein side-chain placement: probabilistic inference and integer programming methods. *MIT Computer Science and Artificial Intelligence Laboratory*.

[20] Marcel Hunting, Ulrich Faigle, and Walter Kern. A lagrangian relaxation approach to the edge-weighted clique problem. *European Journal of Operational Research*, 131:2001, 1999.

[21] G. Glover F. Kochenberger. Diversity data mining. *University of Mississippi*, 1999.

[22] Jan Koolman and Klaus Heinrich Rohm. *Testo atlante di biochimica*. 1997.

[23] Satoshi Matsuyama, Satoshi Nakamura, Katsuki Fujisawa, Kazuhide Nakata, and Masakazu Kojima. Sdpa-m (semidefinite programming algorithm in matlab) user's manual - version 1.00. 2000.

[24] Yari Melzani, Roberto Cordone, and Roberto Aringhieri. Tabu search vs. grasp for the maximum diversity problem. *4OR*, 2008.

[25] David L. Nelson and Michael M. Cox. *I principi di biochimica di Lehninger*.

[26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 1999.

[27] Stephen Prajna, Antonis Papachristodoulou, Pablo A. Parrilo, Copyright (c S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Sostools - sum of squares optimization toolbox for matlab - user's guide version 1.00, 2002.

[28] Geiza C. Silva, Marcos R. Q. de Andrade, Luiz Satoru Ochi, Simone L. Martins, and Alexandre Plastino. New heuristics for the maximum diversity problem. *J. Heuristics*, 13(4):315–336, 2007.

[29] Geiza C. Silva, Luiz Satoru Ochi, and Simone L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In *WEA*, pages 498–512, 2004.

[30] Michael M. Sørensen. New facets and a branch-and-cut algorithm for the weighted clique problem. *European Journal of Operational Research*, 154(1):57–70, 2004.

[31] Park Sungsoo, Park Kyungchul, and Lee Kyungsik. An extended formulation approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 1996.

[32] Robert Vanderbei, , Robert J. Vanderbei, Hande, and Yurttan Benson. On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. Technical report, ORFE 99-01, Dept. of Oper. Res. and Financial Eng., Princeton Univ., Princeton NJ, 1999.

[33] E. Alper Yildirim and Xiaofei Fan-Orzechowski. On extracting maximum stable sets in perfect graphs using lovász's theta function. *Comput. Optim. Appl.*, 33(2-3):229–247, 2006.