Università degli Studi di Padova

Dipartimento di Ingegneria
dell'Informazione
Corso di Laurea in Ingegneria Informatica

Strong Branching on Clique Constraints

Laureando:
Andrea Costalonga

Relatore:
Prof. Domenico
Salvagnin

Anno accademico 2019/2020

Indice

Elenco delle figure

Sommario

La tesi si è sviluppata attorno all'idea di una raccolta dati dalla libreria di problemi Miplib2003 concentrandosi principalmente su problemi MIP in cui sono presenti vincoli di clique. Il diverso peso dell'attribuzione dei valori 0 e 1 nei vincoli di questa tipologia suggerisce la possibilità che fare strong branching sulle variabili dei vincoli di clique sia una buona idea.

Capitolo 1

Introduzione

Nella raccolta dati creata sono stati presi in considerazione dei problemi MIP raccolti nella libreria di problemi Miplib2003; questa classe di problemi di programmazione lineare sono caratterizzati, oltre alla presenza di una funzione lineare da minimizzare e di vincoli lineari, dalla presenza di vincoli di interezza su un sottoinsieme delle variabili. Posso descrivere un problema MIP in forma compatta nella seguente formulazione:

$$
\begin{aligned}
&\min cx \\
&\quad a_i x \sim b_i \quad i = 1, \ldots, m \\
&\quad l_j \leq x_j \leq u_j \quad j = 1, \ldots, n = N \\
&\quad x_j \in \mathbb{Z} \quad \forall j \in J \subseteq N = \{1, \ldots, n\}
\end{aligned}
$$

Figura 1.1: Formulazione MIP

Si può notare che, conseguentemente all'inserimento dei vincoli di interezza delle variabili, non possa essere usato direttamente un rilassamento lineare per la soluzione di questo problema; il rilassamento ottenuto potrebbe contenere variabili con valore frazionario quando nel problema di partenza le stesse variabili appartenevano al dominio degli interi. Per la risoluzione di questo tipo di problemi viene generalmente impiegato un algoritmo chiamato "branch and cut", una versione migliorata dell'algoritmo branch and bound.

1.1 L'algoritmo b&b

Il b&b è lo strumento più importante per la soluzione di problemi MIP. Il primo passo del metodo consiste nel rilassamento lineare del problema P di partenza e nelle considerazioni successive: se il rilassamento è impossibile

allora anche il problema P è impossibile, se si è trovata una soluzione al rilassamento e questa è ammissibile per P allora la soluzione è ottima per P, se la soluzione esiste ma non è intera allora il valore f($\mathbf{x^*}$) sarà un lower bound del problema di partenza. Nell'ultimo caso (il più frequente) la soluzione del rilassamento

$$\mathbf{x}^* = ({x_1}^*, \ldots, {x_\mathrm{n}}^*)$$

non rispetta alcuni vincoli di interezza delle variabili di P ed è quindi necessaria la fase di branching. In questa fase viene considerata una strategia di branching per dividere il problema P in sottoproblemi che vadano a risolvere le conflittualità dovute ai vincoli di interezza delle variabili del problema di partenza. Verrà poi eseguita di nuovo una fase di bound per ogni nuovo nodo e così via. L'albero che si delinea può essere ridotto con una fase di pruning facendo delle considerazioni:

- Se il rilassamento del nodo è impossibile posso scartare il nodo poichè il problema rilassato ha più gradi di libertà del problema non rilassato e quindi se il problema rilassato è impossibile anche il problema non rilassato, che non è altro quindi che una restrizione del rilassamento, è impossibile;

- Se il rilassamento mi ritorna una soluzione ammissibile per P ossia

  $$\mathbf{x^*} \epsilon F(P)$$

  allora:

  - Se è la prima soluzione che trovo salvo il suo valore (incumbent);
  - Se il valore che trovo è inferiore al valore dell'incumbent già trovato allora sostituisco l'incumbent con nuovo valore e salvo la soluzione;
  - Se il valore è inferiore all'incumbent già presente non posso dedurre nulla, continuo l'esplorazione dell'albero;

- Se il rilassamento ritorna una soluzione non ammissibile per P ma il valore dell'incumbent è inferiore al valore del rilassamento in questo nodo allora posso scartare il nodo. Il nodo attuale infatti ha valore del rilassamento superione ad un candidato ad essere soluzione ottima di P, quindi i nodi successivi non porterebbero alcun miglioramento alla f.o. essendo questi una restrizione del problema a questo nodo.

Quando ho finito di esplorare tutti i nodi il valore dell'incumbent (se presente) sarà la soluzione ottima.

1.2 b&c

Il procedimento è analogo a b&b con l'aggiunta nella fase di bound di piani di taglio. Questi piani di taglio sono delle disgiunzioni lineari che riducono lo spazio delle soluzioni togliendo dal pool alcune soluzioni non ammissibili per il problema di partenza. In modo geometrico prendendo come riferimento un generico politopo posso descrivere un piano di taglio come segue:

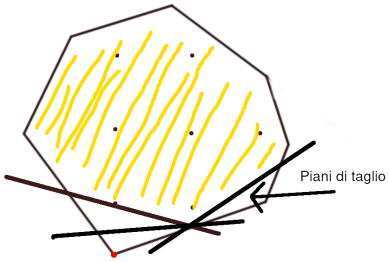

Figura 1.2: Piani di taglio.

Trovare dei piani di taglio efficaci riduce notevolmente il tempo di esecuzione e anche se non fosse possibile trovarne al nodo attuale l'algoritmo continuerebbe in stile b&b.

1.3 Strong Branching

Una delle strategie di branching applicate negli algoritmi b&b e b&c è lo strong branching: questa strategia consiste nel prendere le variabili con vincoli di interezza non soddisfatti nel rilassamento del nodo attuale, eseguire per ogni variabile il rilassamento lineare ponendo prima come Upper Bound

$$UB = \lfloor x_j{}^* \rfloor$$

e poi come Lower Bound

$$LB = \lceil x_j{}^* \rceil$$

e vedere quale di queste variabili frazionarie fornisce una separazione più importante. La separazione più forte sarà quella che verrà preferita nella creazione dei prossimi nodi.

1.4 Clique Constraints

I vincoli presi come candidati ad essere un miglioramento dello strong branching sono i vincoli di clique. Posso esprimere un vincolo di clique come segue:

$$0 \leq \sum_{i \epsilon I} x_{\mathrm{i}} \leq 1 \text{ oppure } \sum_{i \epsilon I} x_{\mathrm{i}} = 1$$
in cui I é un sottoinsieme dell'enumerazione delle variabili booleane del problema

Questi vincoli possono essere rappresentati come una cricca (clique in inglese) ossia un grafo non orientato

$$G = (E, V)$$

in cui per ogni coppia di vertici

$$x, y \, \epsilon \, V$$

esiste un arco

$$e \, \epsilon \, E$$

che li congiunge.

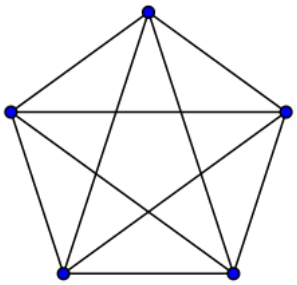

Figura 1.3: Rappresentazione grafica di una Cricca.

Si può notare che questo vincolo è molto sbilanciato quando si parla di decidere se impostare variabile a 0 o a 1: la scelta di mettere una variabile a 1 nel vincolo limita tutte le altre variabili (avranno valore 0) mentre se la stessa variabile viene assegnata a 0 le rimanenti variabili non sono univocamente definite. Su questa osservazione si basa la tesi.

Capitolo 2

Strong Branching on Clique Constraint

2.1 L'idea

I vincoli di clique, come visto nel capitolo precedente, hanno un marcato sbilanciamento quando si tratta di definire quale variabile fissare a 1; da questa considerazione è scaturita questa raccolta dati, che si prefigge il compito di raccogliere del materiale per una possibile futura implementazione di una strategia di branching sui vincoli di clique. La SBCC si propone di testare, come nel semplice Strong Branching, quale dei soggetti presi in causa porta un miglioramento nella f.o. significativo per creare un albero di sottoproblemi di grandezza ridotta. A differenza del semplice Strong Branching il problema iniziale non verrà diviso in 2 sottoproblemi diversa ma in n sottoproblemi dove n è il numero delle variabili del vincolo.
L'obiettivo rimane comunque quello di creare una separazione più forte di quella data dal singolo Strong Branching nel tentativo portare il valore della f.o. il più vicino possibile al valore dell'incumbent. L'approccio di utilizzare il miglior risultato conosciuto al momento è naturalmente un approccio euristico alla pari della strategia Strong Branching. La strategia va a creare molti più nodi per ogni separazione rispetto a S.B. ma si può presumere che molti di questi nodi non verranno mai esporati e che saranno eliminati nella fase di pruning; oltre a ciò lo Strong Branching fissa solo una variabile per ogni iterazione mentre la nuova strategia andrebbe a fissare il valore a tutte le variabili del vincolo.
Si può osservare inoltre che la strategia proposta non esclude alcuna soluzione ammissibile per il problema proprio per la struttura del vincolo di clique.

2.2 La raccolta dati

Per effettuare delle considerazioni sulla effettiva efficacia della strategia è stata effettuata una raccolta dati prendendo in esame i problemi di programmazione intera-mista(MIP) presenti nella libreria Miplib2003. L'algoritmo, scritto in C++ sfruttando le API del risolutore commerciale CPLEX 12.10 di IBM, si è concentrato inizialmente in un esame preliminare del problema P in ingresso in cui viene verificata la presenza di vincoli di clique tra i vincoli di P. Successivamente sono stati eseguiti per ogni variabile binaria il rilassamento in altro ed in basso (UB = 0 e LB = 1) e sono stati salvati i valori della funzione obiettivo. Alla fine sono stati raggruppati i risultati per ogni vincolo di clique e salvati in un file di testo. La prima parte dell'algoritmo è stata fondamentale per scartare in partenza dei MIP senza questi vincoli e quindi evitare elaborazioni inutili. Di seguito un resoconto sui problemi considerati:

Problema	Considerato	Scartato	Problema	Considerato	Scartato
10teams	X		momentum2	X	
a1c1s1		X	momentum3	X	
aflow30a	X		msc98-ip	X	
aflow40b	X		mzzv11		X
air04	X		mzzv42z		X
air05	X		net12		X
arki001		X	noswot		X
atlanta-ip	X		nsand-ipx		X
cap6000	X		nw04	X	
dano3mip	X		opt1217	X	
disctom	X		p2756		X
ds	X		pk1		X
fast0507		X	pp08a		X
fiber	X		pp08aCUTS		X
fixnet6		X	protfold	X	
gesa2		X	qiu		X
gesa2-o		X	rd-rplusc-21	X	
glass4	X		roll300		X
harp2	X		rout		X
liu		X	set1ch		X
manna81		X	seymour		X
markshare1		X	sp97ar		X
markshare2		X	stp3d	X	
mas74		X	swath	X	
mas76		X	t1717	X	
misc07	X		timtab1		X
mkc		X	timtab2		X
mod011	X		tr12-30		X
momentum1	X		vpm2		X

Oltre a verificare la presenza o meno di vincoli di clique in questa prima parte del programma vengono salvate le posizioni delle variabili binarie e dei vincoli di clique essendo dati utili per la successiva elaborazione.
Nella seconda fase vengono risolti 2 rilassamenti per ogni variabile binaria (impostando prima la variabile a 0 e poi a 1). Per ogni vincolo di $<=$ inoltre viene calcolato il valore del rilassamento con tutte le variabili a 0.
I valori ottenuti dai rilassamenti saranno poi il punto cardine nell'analisi dell'efficacia o meno della strategia di branching ipotizzata.

```
1 MIP: mod011.mps
2 LP relaxation result: -6.2122e+07
3
4
5 clique constraint 1 -> c2: 0 <= x1  + x2  + x3  + x4  + x5  + x6  <= 1
6 Var    setTo0      setTo1
7 x1[0..1] -6.2122e+07 -6.06074e+07
8 x2[0..1] -6.2122e+07 -6.0983e+07
9 x3[0..1] -6.2122e+07 -6.11242e+07
10 x4[0..1] -6.2122e+07 -6.07092e+07
11 x5[0..1] -6.2122e+07 -6.08602e+07
12 x6[0..1] -6.19754e+07 -6.05872e+07
13 All Zeroes: -6.15268e+07
14
15 clique constraint 2 -> c3: 0 <= x7  + x8  + x9  + x10  + x11  + x12  <= 1
16 Var    setTo0      setTo1
17 x7[0..1] -6.2122e+07 -6.03878e+07
18 x8[0..1] -6.2122e+07 -6.08709e+07
19 x9[0..1] -6.2122e+07 -6.09566e+07
20 x10[0..1] -6.2122e+07 -6.07246e+07
21 x11[0..1] -6.2122e+07 -6.08716e+07
22 x12[0..1] -6.19083e+07 -6.07786e+07
23 All Zeroes: -6.11865e+07
24
25 clique constraint 3 -> c4: 0 <= x13  + x14  + x15  + x16  + x17  + x18  <= 1
26 Var    setTo0      setTo1
27 x13[0..1] -6.2122e+07 -6.06545e+07
28 x14[0..1] -6.2122e+07 -6.0941e+07
29 x15[0..1] -6.2122e+07 -6.10044e+07
```

Figura 2.1: Esempio file di output.

2.2.1 Miplib2003

I problemi da cui sono state estratte le informazioni sono contenuti nella libreria Miplib2003. La libreria si configura come una collezione di 60 problemi MIP di natura varia ideali per avrebbe un insieme eterogeneo di istanze. La scelta di una libreria datata (i problemi della libreria sono stati formulati dal 1999 al 2003) è stata effettuata consapevolmente data la necessità di avere istanze già risolte (in gran parte) e per la presenza di problemi di piccola taglia utili per fare delle osservazioni dettagliate.

2.3 Software utilizzati

Per raccogliere le informazioni dalla libreria Miplib2003 è stato utilizzato il risolutore commerciale CPLEX di IBM. Il programma si distingue in questo ramo dell'informatica per le buone performance della sua implementazione dell'algoritmo del simplesso e per le molte features messe a disposizione all'utente finale.
Il programma che è stato implementato per la raccolta dati è stato scritto in C++ utilizzando le API del ramo Concert di CPLEX. Queste API, anche se non molto intuitive, si sono dimostrate efficaci per l'implementazione del codice, permettendo all'utente di modificare ed esplorare facilmente il problema di partenza. Per le istanze più onerose computazionalmente è stata

fondamentale la possibilità di iterare le variabili presenti nel vincolo di clique senza dover risolvere tutti i rilassamenti di tutte le variabili booleane (il che avrebbe fatto aumentare esponenzialmente il tempo per la raccolta dati).

Per l'elaborazione dei dati sono stati utilizzati i runner appartenenti al Cluster del Dipartimento di Ingegneria dell'Informazione tramite il sistema di somministrazione dei job SLURM: processare un numero così elevato di rilassamenti lineari non risulta possibile con soluzioni economiche e questi strumenti messi a disposizione dal DEI sono stati fondamentali per la riuscita di questo progetto.

Capitolo 3

Risultati

Di seguito alcune considerazioni sui dati ottenuti. Essendo questa un'analisi preliminare della strategia non sono stati raccolti i risultati a livello prestazionale ma semplicemente dei *numeri* utili a capire se questa strategia ha delle possibilità applicative.

3.1 Considerazioni sui dati ottenuti

Data la grande quantità di dati ottenuti risulta necessario aggregare alcuni risultati. Il primo indicatore utilizzato per verificare il potenziale della SBCC è il seguente: per ogni vincolo di clique $j \, \epsilon \, Cliques$ è stato calcolato un valore

$$s_{\mathrm{j}} = \sqrt[n]{\prod_{i=1}^{n} (R(x_{\mathrm{i}} = 1) - R_{\mathrm{ref}})}$$

in cui $R(x_{\mathrm{i}})$ è il valore del rilassamento fissando a 1 la variabile x_{i} del vincolo e R_{ref} è il valore del rilassamento del problema iniziale. Per confronto ad ogni variabile binaria $k \, \epsilon \, Binaries$ coinvolta in almeno un vincolo di clique è stato assegnato uno score

$$b_{\mathrm{k}} = \sqrt{(R(x_{\mathrm{k}} = 0) - R_{\mathrm{ref}}) * (R(x_{\mathrm{k}} = 1) - R_{\mathrm{ref}})}$$

Si può notare che nel caso uno dei valori nella produttoria è 0 allora lo score sarà 0, per ovviare a ciò è stato considerato il valore 0.01 al posto di 0.
Per ogni problema è stato raccolto il migliore s_{j} e b_{k} ed è stato fatto il rapporto tra i due valori. Se questo è > 1 allora almeno una clique del problema ha creato una separazione migliore del semplice SB sulle variabili binarie.

Di seguito il grafico ottenuto[Figura 3.1]:

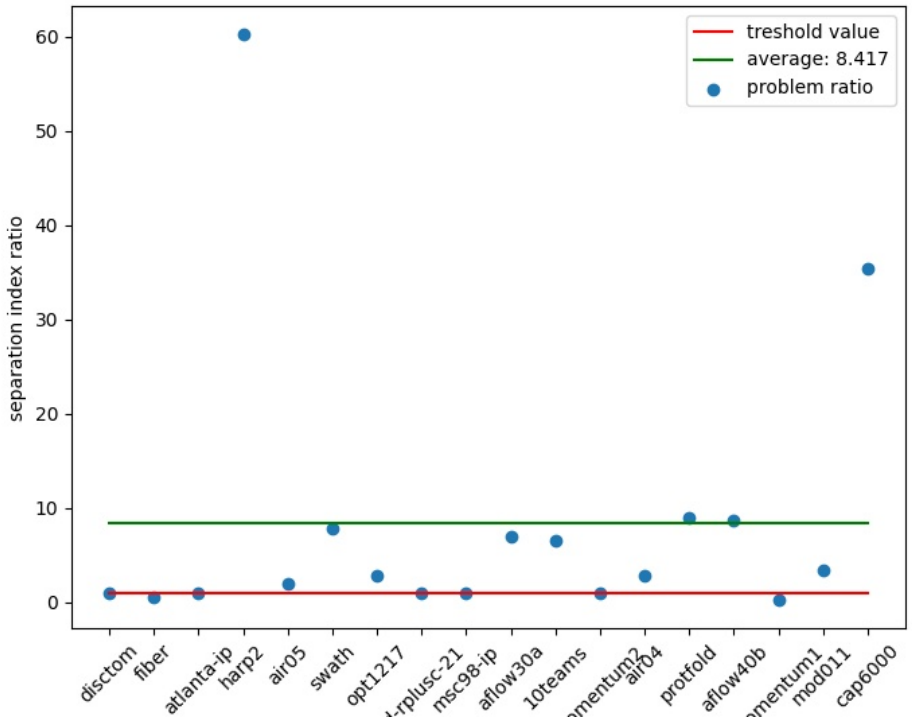

Figura 3.1: Problem ratio

Si può osservare che in molti dei problemi la separazione che va a crearsi è migliore della miglior separazione tramite Strong Branching delle variabili coinvolte nei vincoli.

3.2 Mod011.mps

Capitolo 4

Conclusioni

Capitolo 5

Bibliografia