Matheuristics for Combinatorial Optimization problems

PhD in Computer Science

Roberto Cordone

DI - Università degli Studi di Milano

Phone nr.: 02 503 16235

E-mail: roberto.cordone@unimi.it

Web page: https://homes.di.unimi.it/cordone/courses/2024-mhco/2024-mhco.html

Lezione 6: Euristiche basate su column generation Milano, A.A. 2023/24

Programmazione Lineare (PL): forma standard

Ogni problema di Programmazione Lineare si può porre in forma canonica

$$\begin{aligned} \min f = c^T x \quad &+ \quad d \qquad (\Pi) \\ Ax + I\hat{x} \quad &= \quad b \\ x, \hat{x} \quad &\geq \quad 0 \end{aligned}$$

che deriva dalla forma introdotta in precedenza

- trasformando le disuguaglianze in uguaglianze con variabili di slack

$$a_i^T x \leq b_i \Leftrightarrow \begin{cases} a_i^T x + \hat{x}_i = b_i \\ \hat{x}_i \geq 0 \end{cases} \qquad \text{per } i = 1, \ldots, m$$

- trasformando le variabili libere in coppie di variabili non negative

$$\begin{cases} x_j \to x_j^+ - x_j^- \\ x_j^+, x_j^- \geq 0 \end{cases} \qquad \text{per ogni } x_j \in \mathbb{R}$$

Proprietà della forma canonica

Posto un problema in forma canonica

- le variabili $\hat{x}$ si chiamano variabili di base
- i costi $\begin{bmatrix} c^T & 0^T \end{bmatrix}$ si chiamano costi ridotti

$$\begin{array}{rcll} \min f = c^T x & + & d & (\Pi) \\ Ax + I\hat{x} & = & b & \\ x & \geq & 0 & \end{array}$$

La forma canonica è definita da due proprietà fondamentali

1. le colonne relative alle variabili di base $\hat{x}$ danno una matrice identità
2. i costi ridotti delle variabili di base sono nulli

e rende facilissimo trovare una soluzione associata (soluzione di base):

- variabili fuori base nulle: $x_j = 0$ per ogni $j = 1, \ldots, n$
- variabili in base pari ai termini noti: $\hat{x}_i = b_i$ per ogni $i = 1, \ldots, m$

Il *tableau*

Ogni forma canonica corrisponde a un *tableau*, cioè una matrice dove

- le righe $1, \ldots, m$ corrispondono ai vincoli
- le colonne $1, \ldots, n$ alle variabili
- la riga 0 alla funzione obiettivo, nella forma $c^T x = f(x) - d$, con $f(x)$ implicita: $a_{00} := -d$ e $a_{0j} := c_j$ $(j = 1, \ldots, n)$
- la colonna 0 al vettore dei termini noti: $a_{i0} := b_i$ $(i = 1, \ldots, m)$

$$\begin{aligned} c^T x &= f(x) - d \\ A x + I \hat{x} &= b \\ x, \hat{x} &\geq 0 \end{aligned}$$

$-d$	c^T	0
b	A	I

$$\begin{aligned} \min f = 3x_1 - x_2 - 4x_3 &\quad - 8 \\ 3x_1 + x_2 - 2x_3 + x_4 &= 2 \\ -x_1 + x_3 + x_5 &= 2 \\ 6x_1 + x_2 - 5x_3 + x_6 &= 3 \\ x_1, \ldots, x_6 &\geq 0 \end{aligned}$$

8	3	−1	−4	0	0	0
2	3	1	−2	1	0	0
2	−1	0	1	0	1	0
3	6	1	−5	0	0	1

L'algoritmo del simplesso

L'algoritmo del simplesso esplora diverse forme canoniche del problema

- equivalenti fra loro (uguali soluzioni ammissibili con uguali valori)
- espresse rispetto a variabili di base diverse

Ad ogni iterazione

- sceglie una variabile x fuori base da far entrare in base
- determina (univocamente) una variabile $\hat{x}$ in base da far uscire
- aggiorna le celle del *tableau* per ottenere la nuova forma canonica

Si dimostra che

1. quando $b \geq 0$ la soluzione di base $(x, \hat{x}) = (0, b)$ è ammissibile
2. quando $c \geq 0$ la soluzione $(x, \hat{x}) = (0, b)$ è ottima

L'algoritmo del simplesso introduce nell'ordine le due condizioni

Dualità

Ogni problema di *PL* ha un problema duale, costruito meccanicamente con gli stessi elementi in una diversa struttura:

- il segno dell'ottimizzazione si inverte
- righe e colonne si scambiano
  - variabili e vincoli
  - costi e termini noti dei vincoli
- i segni delle disuguaglianze e i limiti sulle variabili seguono regole dipendenti dal segno dell'ottimizzazione

Problema primale	Problema duale
$\min f(x) = c^T x + d$	$\max \phi(y) = b^T y + d$
$Ax \leq b$	$A^T y \leq c$
$x \geq 0$	$y \leq 0$

In genere si pone $Ax \geq b$ *e* $y \geq 0$, *ma questa forma è più utile nel seguito*

Dualità e *tableau*

Il *tableau* rappresenta entrambi i problemi in forma standard, a patto di ricordare che il duale è un problema di massimo con $y \leq 0$, ma $\hat{y} \geq 0$

Problema primale

$$\begin{aligned} \min f(x) = c^T x &+ d \\ Ax + I\hat{x} &= b \\ x, \hat{x} &\geq 0 \end{aligned}$$

Problema duale

$$\begin{aligned} \max \phi(y) = b^T y &+ d \\ A^T y + I\hat{y} &= c \\ y \leq 0, \hat{y} &\geq 0 \end{aligned}$$

$-d$	c^T	0
b	A	I
0	I	

Le soluzioni di base delle due forme canoniche sono strettamente legate

- hanno sempre lo stesso valore d: $\begin{cases} (x, \hat{x}) = (0, b) \Rightarrow f(x, \hat{x}) = d \\ (y, \hat{y}) = (0, c) \Rightarrow \phi(y, \hat{y}) = d \end{cases}$
- se $b \geq 0$, la soluzione primale è ammissibile, la duale superottima
- se $c \geq 0$, la soluzione primale è superottima, la duale ammissibile

Scarti complementari

Le soluzioni di base delle due forme canoniche sono strettamente legate

- variabile primale fuori base $(x) \leftrightarrow$ variabile duale in base $(\hat{y})$
- variabile primale in base $(\hat{x}) \leftrightarrow$ variabile duale fuori base (y)

Il loro prodotto è nullo per ogni coppia (scarto complementare)

$$\begin{cases} x_j\, \hat{y}_j = 0 \text{ per ogni } j = 1, \ldots, n \\ \hat{x}_i\, y_i = 0 \text{ per ogni } i = 1, \ldots, m \end{cases}$$

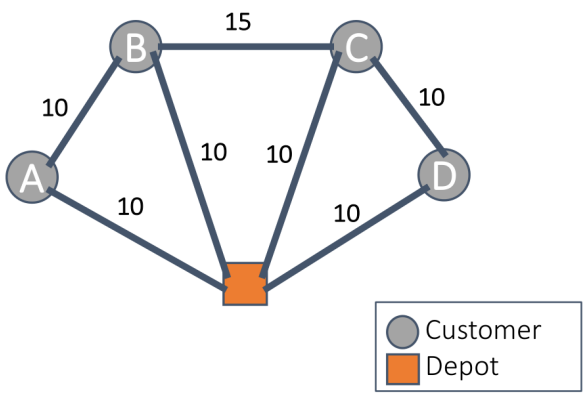
Si ricavano le une dalle altre risolvendo un sistema lineare di equazioni

Esempio: Vehicle Routing Problem (*VRP*)

Dati

- un grafo orientato $G = (N, A)$ con un nodo deposito $d \in N$
- una funzione $c : A \to \mathbb{N}$ che assegna un costo ad ogni arco
- una funzione $w : N \to \mathbb{N}$ che assegna un peso ad ogni nodo
- un numero $W \in \mathbb{N}$ che descrive una capacità massima

il *VRP* cerca l'insieme di circuiti di costo minimo che toccano tutti i nodi e ciascuno dei quali visita d e ha peso totale $\leq W$



$w_i = 1$ per ogni $i \in N$
$W = 2$

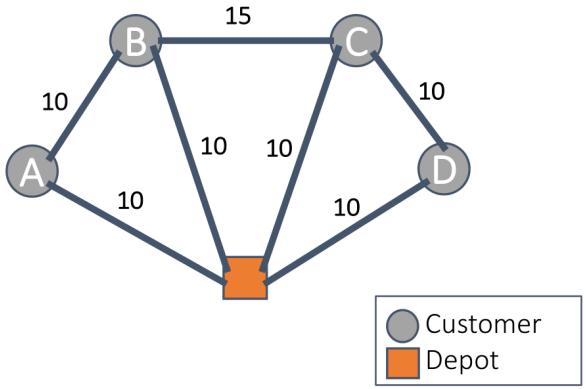
Definiamo come insieme base B quello dei circuiti basati su d e di peso $\leq W$

$$\min f(x) = \sum_{p \in B} c_p x_p$$

$$\sum_{p \in B} a_{ip} x_p = 1 \qquad i \in N$$

$$x_p \in \{0, 1\} \qquad p \in B$$

Esempio: Vehicle Routing Problem (*VRP*)



c

20	20	20	20	30	30	35

A

1	0	0	0	1	0	0
0	1	0	0	1	0	1
0	0	1	0	0	1	1
0	0	0	1	0	1	0

$w_i = 1$ per ogni $i \in N$

$W = 2$

$$
\begin{aligned}
\min f(x) &= 20x_1 + 20x_2 + 20x_3 + 20x_4 + \\
&+ 30x_5 + 30x_6 + 35x_7 \\
x_1 + x_5 &= 1 \\
x_2 + x_5 + x_7 &= 1 \\
x_3 + x_6 + x_7 &= 1 \\
x_4 + x_6 &= 1 \\
x_i &\geq 0 \quad \forall i
\end{aligned}
$$

$$
\begin{aligned}
\min \phi(y) &= y_8 + y_9 + y_{10} + y_{11} \\
y_8 &\leq 20 \\
y_9 &\leq 20 \\
y_{10} &\leq 20 \\
y_{11} &\leq 20 \\
y_8 + y_9 &\leq 30 \\
y_{10} + y_{11} &\leq 30 \\
y_9 + y_{10} &\leq 35 \\
y_i &\geq 0 \quad \forall i
\end{aligned}
$$

$$x^* = (0, 0, 0, 0, 1, 1, 0)$$
$$\hat{x}^* = (0, 0, 0, 0)$$
$$f^* = 60$$

$$\hat{y}^* = (10, 0, 5, 5, 0, 0, 0)$$
$$y^* = (10, 20, 15, 15)$$
$$\phi^* = 60$$

Column generation

Se le variabili primali (colonne) sono molte, si può pensare di

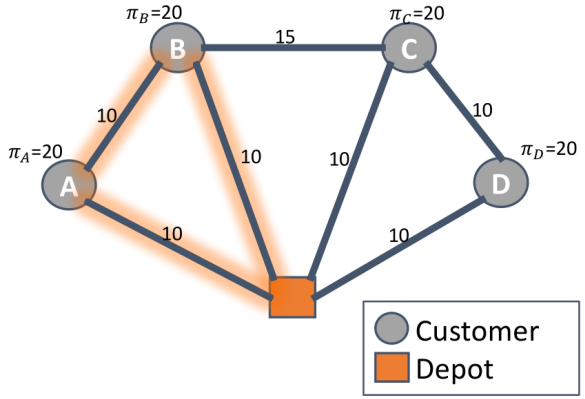
- fissare a 0 alcune variabili primali, ottenendo un problema ristretto $\bar{R}$
  Questo corrisponde a rilassare alcuni vincoli duali
- risolvere $\bar{R}$ con iterazioni di simplesso più veloci che per R
- calcolare i costi ridotti delle variabili escluse da $\bar{R}$ (variabili duali) spendendo un po' di tempo aggiuntivo
- se vi sono costi ridotti negativi (cioè il duale non è ammissibile), aggiungere a $\bar{R}$ le variabili corrispondenti e tornare al punto 2;
  se i costi ridotti sono non negativi, la soluzione è ottima per R

Se il risparmio nell'aggiornare il *tableau* supera il calcolo dei costi ridotti, l'algoritmo è più veloce

Esempio: Vehicle Routing Problem (*VRP*)

Partiamo con i soli circuiti dedicati

c	20	20	20	20	y
A	1	0	0	0	20
	0	1	0	0	20
	0	0	1	0	20
	0	0	0	1	20



$w_i = 1$ per ogni $i \in N$
$W = 2$

$$\begin{aligned} \min f(x) &= 20x_1 + 20x_2 + 20x_3 + 20x_4 \\ x_1 &= 1 \\ x_2 &= 1 \\ x_3 &= 1 \\ x_4 &= 1 \\ x_i &\geq 0 \quad \forall i \end{aligned}$$

$$\begin{aligned} \min \phi(y) &= y_8 + y_9 + y_{10} + y_{11} \\ y_8 &\leq 20 \\ y_9 &\leq 20 \\ y_{10} &\leq 20 \\ y_{11} &\leq 20 \\ y_i &\geq 0 \quad \forall i \end{aligned}$$

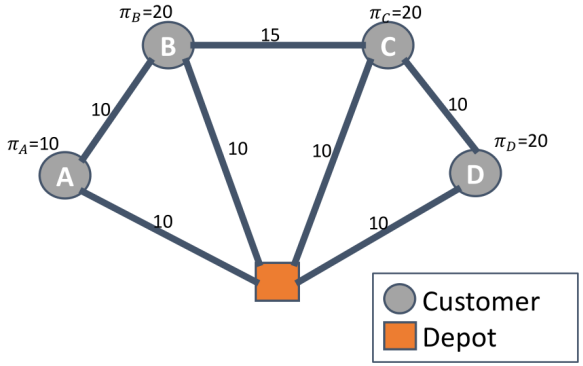
$$x = (1, 1, 1, 1, 0, 0, 0)$$
$$\hat{x} = (0, 0, 0, 0)$$
$$f = 80$$

$$\hat{y} = (0, 0, 0, 0, -10, -10, -5)$$
$$y = (20, 20, 20, 20)$$
$$\phi = 80$$

Il circuito x_5 ha costo ridotto $\tilde{c}_5 = \hat{y}_5 = 30 - 20 - 20 < 0 \Rightarrow$ va aggiunto!

Esempio: Vehicle Routing Problem (*VRP*)

c	20	20	20	20	30	y
A	1	0	0	0	1	10
	0	1	0	0	1	20
	0	0	1	0	0	20
	0	0	0	1	0	20



$w_i = 1$ per ogni $i \in N$
$W = 2$

$$\begin{aligned} \min f(x) &= 20x_1 + 20x_2 + 20x_3 + 20x_4 + 30x_5 \\ x_1 + x_5 &= 1 \\ x_2 + x_5 &= 1 \\ x_3 &= 1 \\ x_4 &= 1 \\ x_i &\geq 0 \quad \forall i \end{aligned}$$

$$\begin{aligned} \min \phi(y) &= y_8 + y_9 + y_{10} + y_{11} \\ y_8 &\leq 20 \\ y_9 &\leq 20 \\ y_{10} &\leq 20 \\ y_{11} &\leq 20 \\ y_8 + y_9 &\leq 30 \\ y_i &\geq 0 \quad \forall i \end{aligned}$$

$$x = (0, 0, 1, 1, 1, 0, 0)$$
$$\hat{x} = (0, 0, 0, 0)$$
$$f = 70$$

$$\hat{y} = (0, 0, 0, 0, 0, -10, -5)$$
$$y = (10, 20, 20, 20)$$
$$\phi = 70$$

E così via...

Column generation per problemi esponenziali

Se le variabili sono molte, non aggiornare il *tableau* è un forte risparmio, ma ricalcolare tutti i costi ridotti rimane proibitivo

Tuttavia, non occorre calcolarli tutti: basta trovarne uno negativo
ed esistono costi negativi se e solo se il costo minimo è negativo

$$\exists j \in B : \bar{c}_j < 0 \quad \Leftrightarrow \quad \min_{j \in B} \bar{c}_j = c_j - \sum_{i=1}^{m} a_{ij} y_i < 0$$

questo equivale a risolvere un problema di pricing in cui

- y_i hanno valori noti (variabili duali ottime correnti)
- c e a_i sono incognite che descrivono un elemento dell'insieme base B

Questo è un problema di Ottimizzazione Combinatoria

Esempio: Vehicle Routing Problem (*VRP*)

Problema master

$$\min f(x) = \sum_{p \in B} c_p x_p$$

$$\sum_{p \in B} a_{ip} x_p = 1 \qquad i \in N$$

$$x_p \in \{0,1\} \qquad p \in B$$

L'insieme base B enumera i circuiti basati su d e di peso $\leq W$

$a_i = 1$: il circuito include $i \in N$

q_i è il carico alla partenza da $i \in N$

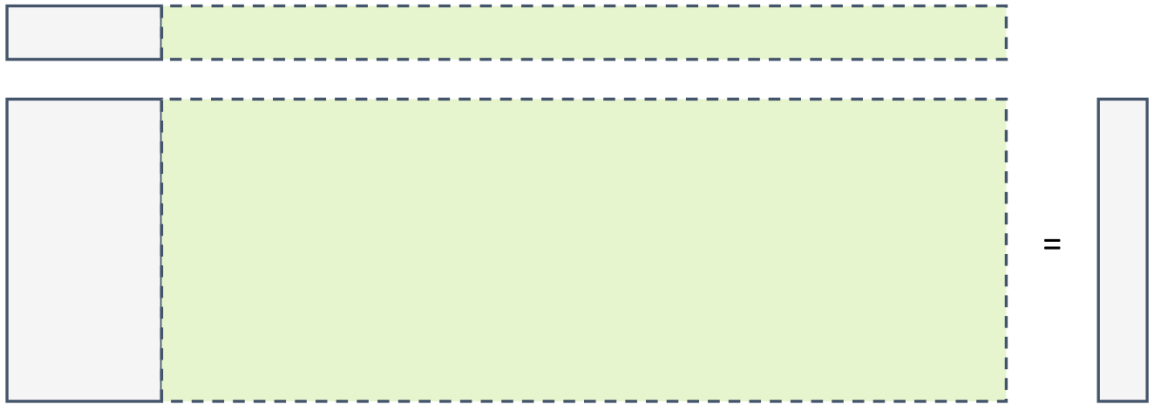
$b_{ij} = 1$: il circuito include $(i,j) \in A$

Problema di pricing

Resource-constrained Shortest Path

$$\begin{aligned}
\min \bar{c}(x) &= \sum_{(i,j) \in A} c_{ij} b_{ij} - \sum_{i \in N} y_i a_i \\
\sum_{j \in N \setminus \{d\}} b_{dj} &= 1 \\
\sum_{j \in N} b_{ji} &= \sum_{j \in N} b_{ij} && i \in N \\
q_i + w_j &\leq q_j + W(1 - b_{ij}) && i \in N \\
a_i &= \sum_{i \in N} b_{ij} && i \in N \\
0 \leq q_i &\leq W\, a_i && i \in N \\
a_i &\in \{0,1\} && i \in N \\
b_{ij} &\in \{0,1\} && (i,j) \in A
\end{aligned}$$

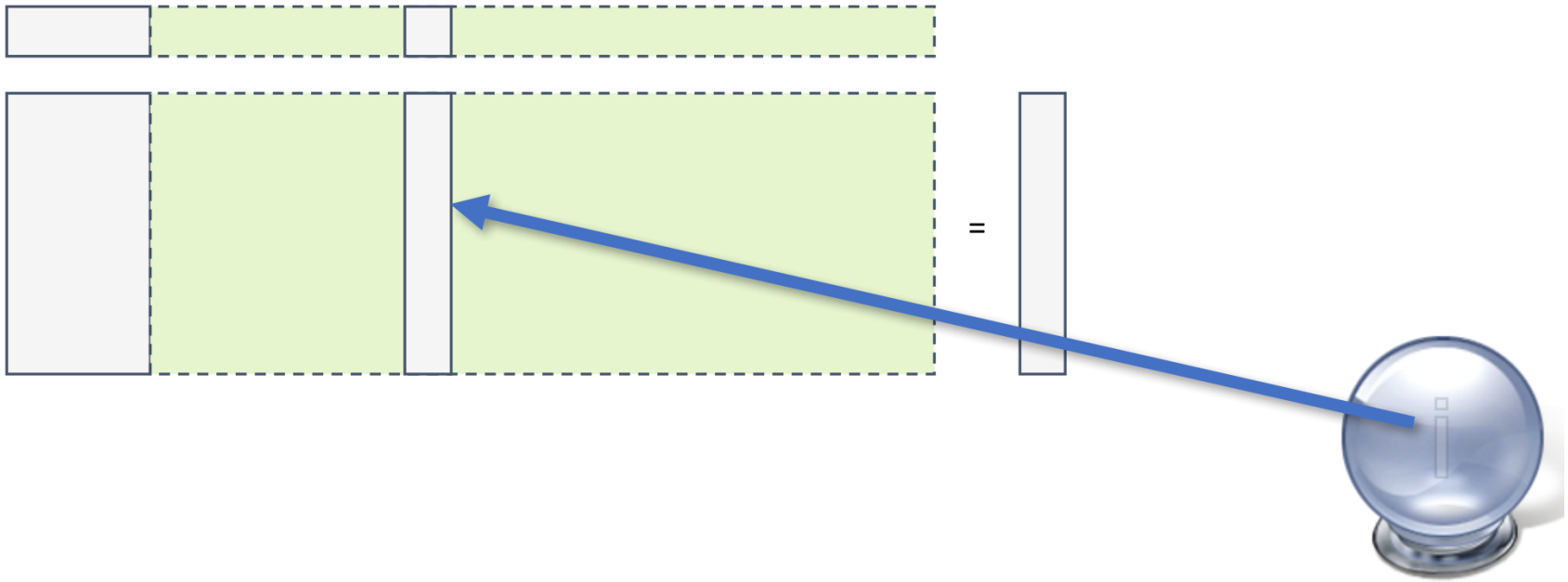
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

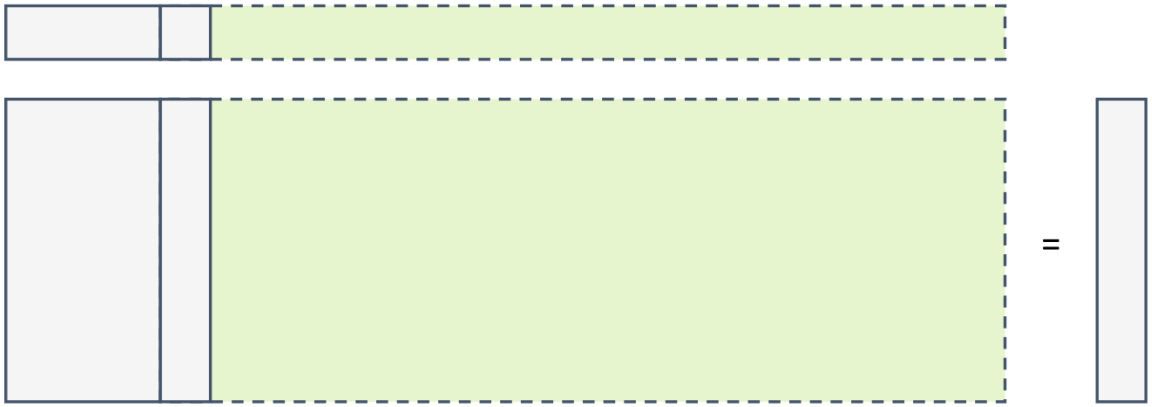
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

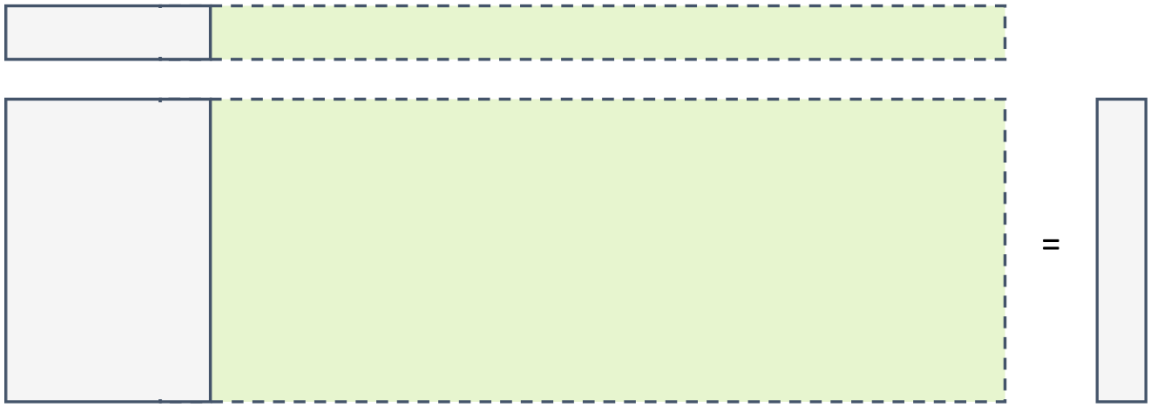
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

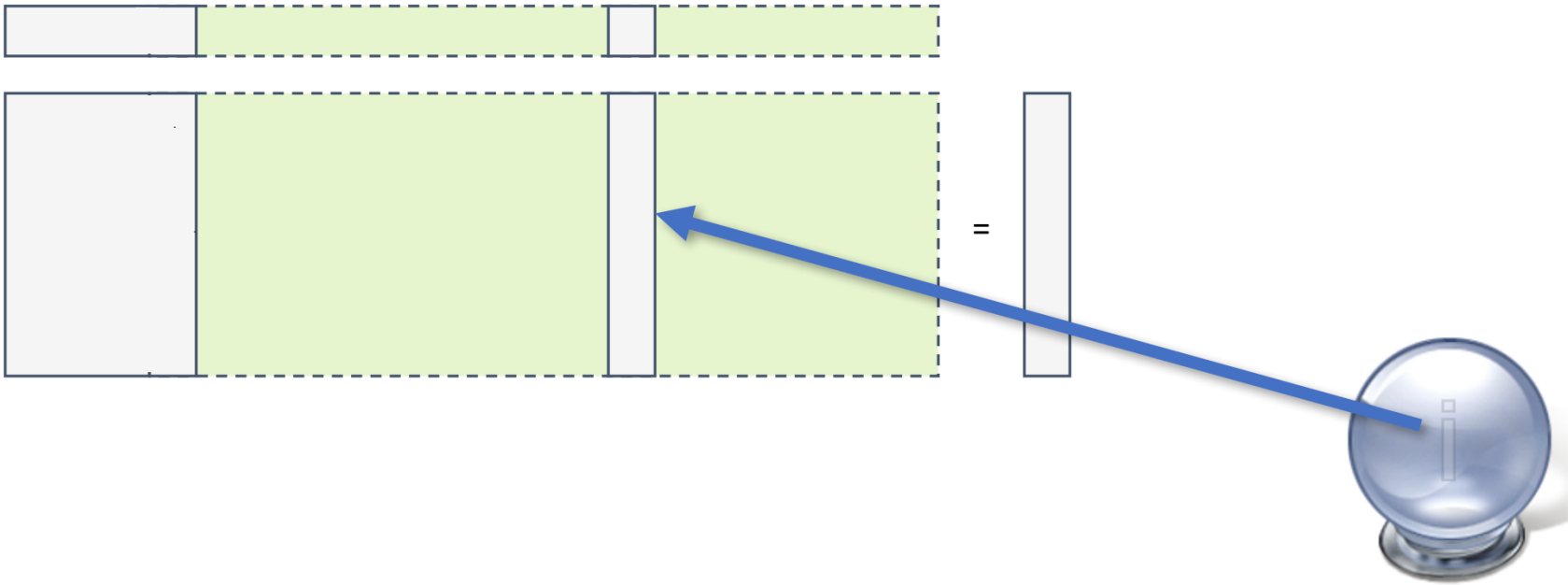
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

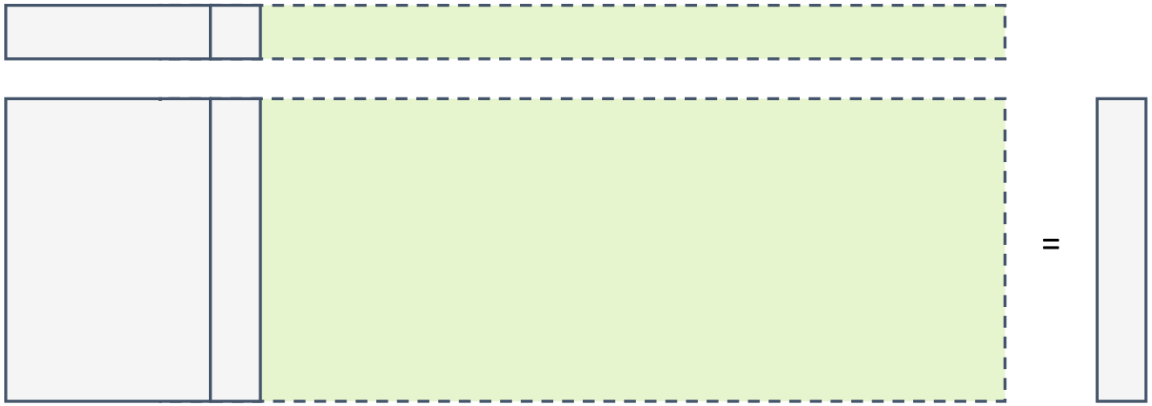
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

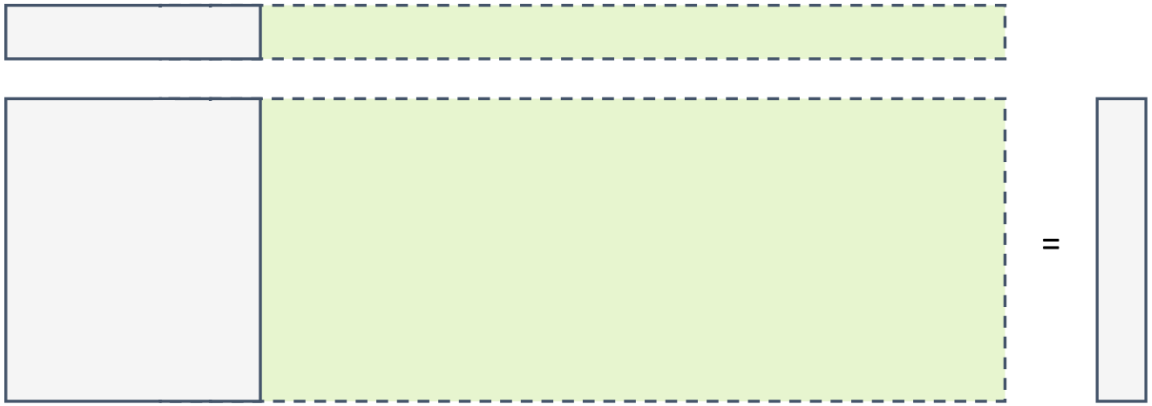
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

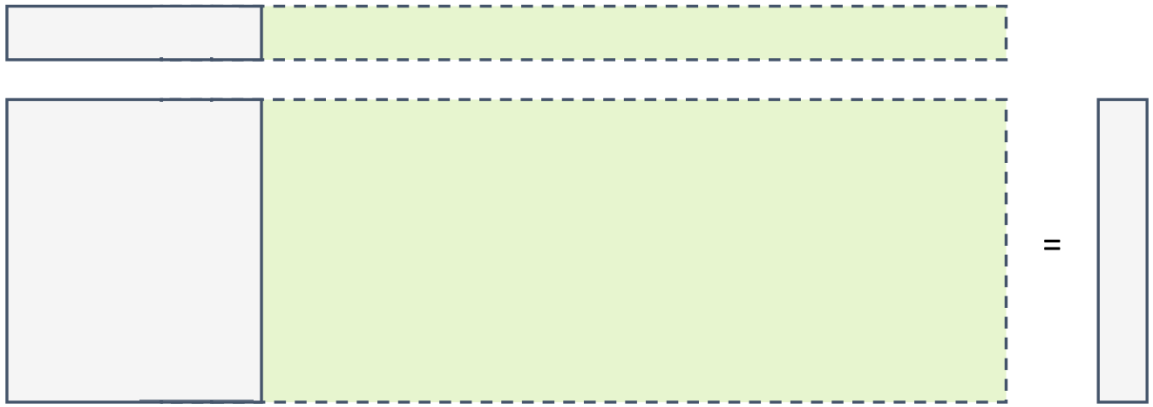
Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

Column generation: schema



Riassumendo, si attraversano le seguenti fasi

1. si crea il master problem ristretto $\bar{R}$ con un sottoinsieme di colonne
2. si risolve il pricing problem, individuando una colonna di costo $\bar{c}_j < 0$ o dimostrando che non ne esistono (nel qual caso, si termina)
3. si aggiunge la colonna al master problem ristretto
4. si riottimizza il master problem ristretto con le sue variabili duali e si torna al punto 2

Branch-and-price

Terminata la column generation, il rilassamento continuo R è risolto, cioè è risolto il nodo radice dell'albero di branching

Il seguito avviene come nei branch-and-bound classici

- chiusura del nodo, se inammissibile
- chiusura del nodo, se dominato
- ricerca di soluzioni euristiche a partire da quella ottima rilassata
- chiusura del nodo, se ha soluzione ottima intera
- branching, con produzione di più nodi da affrontare come sopra, compresa una nuova column generation per ogni nodo

Il branching interagisce in modo delicato con la generazione delle colonne, perché a volte cambia la struttura del problema

Per brevità, parleremo solo dell'impatto sulle euristiche

Limitazioni e tolleranze nella column generation

Qualsiasi euristica per il branch-and-bound si estende al branch-and-price

- limitare il tempo
- limitare il numero di nodi di branching
- limitare il numero di soluzioni individuate

ma alcune euristiche hanno proprietà aggiuntive

Dato un problema P e il suo rilassamento continuo R, sia

- $\bar{R}$ il master problem ristretto tratto da R
- $f^*_{\bar{R}}$ l'ottimo di $\bar{R}$
- $\bar{c}_{\bar{R}}$ i suoi costi ridotti

Se i costi ridotti sono tutti non negativi ($\bar{c}_{\bar{R}} \geq 0$)

1. la soluzione di $\bar{R}$ è ottima per il rilassamento R: $f^*_{\bar{R}} = f^*_R$
2. il suo valore stima per difetto l'ottimo del problema dato: $f^*_{\bar{R}} \leq f^*$
3. consente di chiuder nodi con la condizione di dominanza: $f^*_R \geq f(\bar{x})$

Se esistono costi ridotti negativi, in generale $f^*_{\bar{R}} \geq f^*_R$ e non si può usare: variabili attualmente nulle potrebbero entrare in base e ridurre $f^*_{\bar{R}}$

Di quanto?

Limitazioni e tolleranze nella column generation

Il calo dovuto ai costi ridotti negativi ha un limite intrinseco

- ogni variabile $x_j = 0$ può crescere al massimo di $\epsilon \leq 1$
- se $\bar{c}_j < 0$ e $x_j = 0$ cresce di ϵ, l'obiettivo varia di $\delta f \geq \bar{c}_j \epsilon$
- se una soluzione ammissibile ha al massimo v_{max} variabili positive, le variabili nulle che possono entrare in base sono $\leq v_{max}$

Nel caso pessimo (variabili ottime tutte fuori base e di costo ridotto $\min_{j \in B} c_j$)

$$f_{\bar{R}}^* + v_{\max} \min_{j \in B} c_j \leq f_R^* \leq f_{\bar{R}}^*$$

che può sostituire l'ancora ignota f_R^* nella condizione di dominanza

$$f_{\bar{R}}^* + v_{\max} \min_{j \in B} c_j \geq f(\bar{x})$$

Non occorre risolvere all'ottimo il problema di pricing, ma basta mostrare che

$$c_j \geq \frac{f(\bar{x}) - f_{\bar{R}}^*}{v_{\max}} \; \forall j \in B$$

che è più facile di $c_j \geq 0$ perché la dominanza è possibile solo se $f_{\bar{R}}^* > f(\bar{x})$

Una tolleranza euristica sull'ottimo si traduce in un'accelerazione del pricing

$$f_{\bar{R}}^* + v_{\max} \min_{j \in B} c_j \geq f(\bar{x}) - \Delta \Leftrightarrow c_j \geq \frac{f(\bar{x}) - f_{\bar{R}}^* - \Delta}{v_{\max}} < z \forall j \in B$$

Restricted master heuristic

La Restricted master heuristic si basa sull'idea di considerare molto promettenti le colonne generate al nodo radice

1. si risolve il rilassamento continuo generando le colonne necessarie
2. si forzano a 0 tutte le altre variabili
3. si risolve il problema di *PLI* ristretto alle variabili generate

Il metodo non è esatto perché alcune colonne ottime vengono generate solo dopo le opportune operazioni di branching

Lo svantaggio fondamentale è che il problema ristretto

- potrebbe non essere ammissibile
- se è ammissibile perché inizializzato con soluzioni euristiche, potrebbe non fornire soluzioni miglioranti

Questo suggerisce l'idea di

- rilassare parzialmente il problema master fino a renderlo ammissibile e riparare euristicamente la soluzione rilassata così ottenuta
- arricchire il master problem con colonne euristiche
- entrambe le cose (ad es., aggiungendo colonne euristiche ricavate dalla riparazione di una soluzione rilassata del problema master)

Column generation e diving

Anche le euristiche di diving si estendono con qualche osservazione

Tipicamente, avendo risolto il rilassamento al nodo radice

1. si considera come valore promettente solo $\tilde{x}_j = 1$ perché
   - fissare le variabili a 1 riduce fortemente il problema master
     Quindi si hanno al massimo $v_{\max}$ iterazioni
   - fissare le variabili a 0 introduce un vincolo complicante nel pricing
     senza ridurre sensibilmente il master
2. si calcola un indice di confidenza ϕ_j (spesso semplicemente $1 - x_{\bar{R}_j}$)
3. si fissano le migliori δ variabili a 1 (tipicamente $\delta = 1$)
4. si applica il presolve al sottoproblema
   e si risolve il rilassamento con una nuova fase di column generation
5. se la soluzione è ammissibile, ma frazionaria, si torna al punto 1

La forza dello schema è che le successive fasi generano colonne utili a evitare di ottenere problemi inammissibili o soluzioni cattive

Diving con restart

L'euristica di diving con restart itera il procedimento del diving in modo da intensificare la ricerca attorno alla soluzione trovata dal diving

Prima esegue un'euristica di diving classica ottenendo la soluzione $\bar{x}$
Quindi, per un dato numero di iterazioni I

1. sceglie un sottoinsieme casuale $\tilde{J}$ di $\alpha |x|$ colonne appartenenti a $\bar{x}$
   (*ovviamente* α *è un parametro critico da tarare*)
2. pone $x_j = 1$ per ogni $j \in \tilde{J}$
3. costruisce il sottoproblema risultante $\Pi_{\tilde{J}}$
   e ne risolve il rilassamento $R_{\tilde{J}}$ con la column generation
4. se il problema è ammissibile e frazionario, torna al punto 1;
   se è intero, aggiorna eventualmente la miglior soluzione nota $\bar{x}$

Strong diving

L'euristica di strong diving si basa sull'idea di scegliere la variabile di diving in base alle conseguenze di ciascuna alternativa al passo successivo

1. sceglie un sottoinsieme $\tilde{J}$ di colonne promettenti
2. per ogni $j \in \tilde{J}$
   - pone temporaneamente $x_j = 1$
   - costruisce il sottoproblema risultante Π_j, ne risolve il rilassamento R_j con la column generation e calcola il valore ottimo $f^*_{R_j}$
3. tornata al problema di partenza, pone $x_{j^*} = 1$ con

$$j^* = \arg\min_{j \in \tilde{J}} f^*_{R_j}$$

perché fissare j^* ha deteriorato il bound il meno possibile

Nota: la tecnica di "strong branching" fa una cosa simile per scegliere la variabile di branching, ma opera al contrario (massimizza il bound) e prova tutti i valori, perché non mira a trovare soluzioni euristiche buone, ma ad alzare il bound

Diving con sub-MIP

L'euristica di diving con sub-MIP si basa sull'idea di

- eseguire il diving finché la dimensione del problema supera una soglia
  (*numero di variabili, vincoli, variabili frazionarie, ecc. . .*)
- altrimenti, risolvere il problema ridotto con il branch-and-bound

Questo consente di

- proseguire l'esplorazione dell'albero di branching oltre la prima foglia
- controllare il tempo di calcolo attraverso la dimensione sotto la quale si esegue il branch-and-bound
  (*eventualmente limitando anche il tempo, i nodi o le soluzioni*)

La Restricted Master Heuristic è una specie di diving con sub-MIP in cui

- il diving si interrompe al nodo radice
- le colonne non generate al nodo radice vengono tutte fissate a 0
- il problema ridotto è formato dalle colonne generate

Limited Discrepancy Search

La Limited Discrepancy Search è un'euristica per branch-and-bound generici, usata soprattutto in quelli basati sulla column generation

L'idea essenziale è che

- il diving fallisce a causa di un piccolo numero di variabili sbagliate
- l'errore è commesso ai primi passi

Quindi, violeremo l'indicazione dell'indice di confidenza

- in un piccolo numero di casi
- nei livelli più alti dell'albero di branching

L'euristica ha due parametri fondamentali

1. la massima profondità $D_{\max}$, cioè il livello oltre il quale si rispettano le indicazioni dell'indice di confidenza ϕ_j
2. la massima discrepanza $L_{\max}$, cioè il massimo numero di variabili che ϕ_j consiglia di porre a 1 e che vengono invece poste a 0

Limited Discrepancy Search

Si procede quindi così

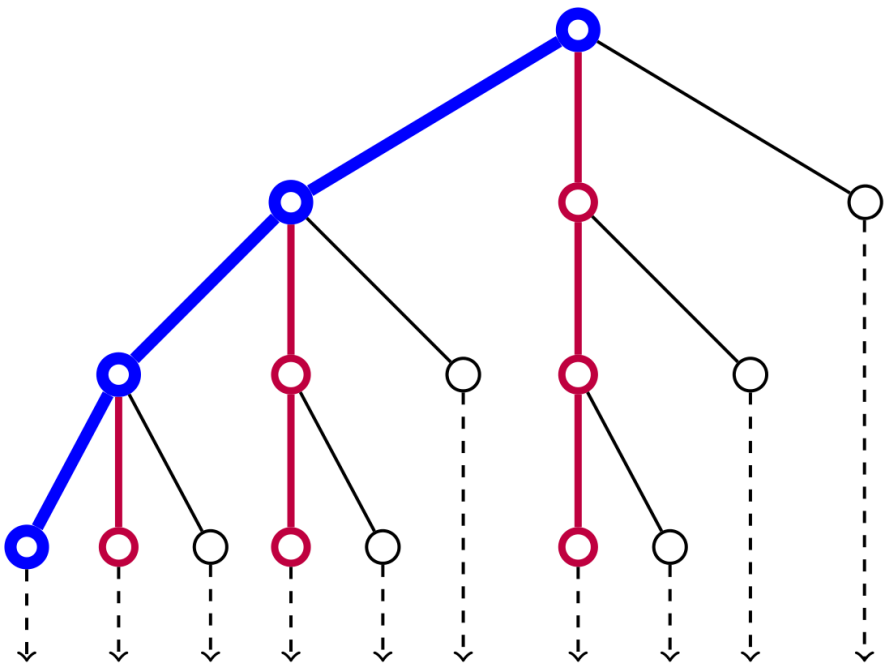
1. si crea una lista di variabili vietate, inizialmente vuota ($L := \emptyset$)
2. si esegue un'euristica di diving classica ottenendo $\bar{x}$
3. si risale la sequenza di variabili fissate (backtracking) sino al livello massimo d che abbia
   - indice $d < D_{\max}$
   - $|L| < L_{\max}$ variabili vietate

   Il backtracking libera sia le variabili fissate sia quelle vietate
4. se $|L| < L_{\max}$, sceglie la prima variabile j indicata da ϕ e non vietata, pone $x_j = 1$ e riapplica l'euristica di diving con la column generation

La *LDS* fissa variabili a 1, ma anche a 0:

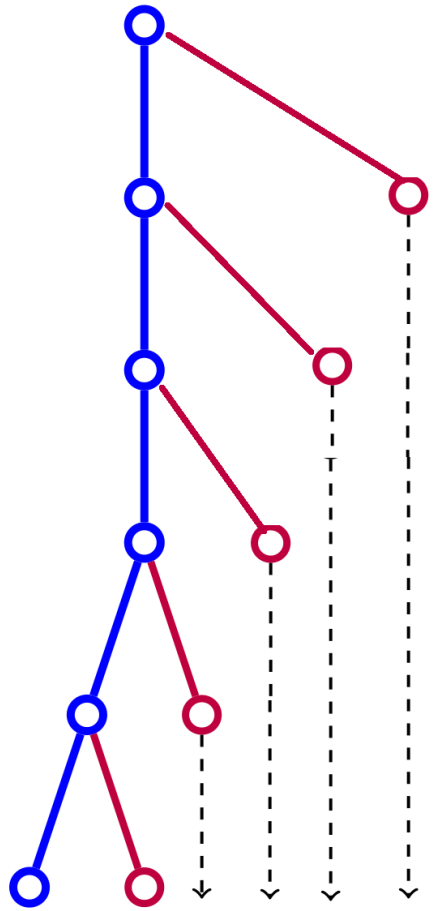
- il pricing va modificato in modo che non rigeneri le variabili vietate
- se le variabile vietate sono poche, si può fare senza complicazioni (*con una verifica esplicita della lista*)

Esempi



$maxDepth = 3$

$maxDiscrepancy = 2$



$maxDepth = 5$

$maxDiscrepancy = 1$

LDS per l'Homogeneous Areas Problem

In Italia le province coordinano i comuni che devono interagire per attività sovracomunali: parchi, strade, eventi, incentivi economici,...

La provincia offre un help desk che

- aiuta direttamente i funzionari comunali
- li indirizza agli uffici provinciali specializzati

Siccome l'help desk deve essere esperto di molte attività diverse l'efficienza suggerisce di

- raggruppare in aree omogenee i comuni coinvolti nelle stesse attività
- assegnare ogni area a una squadra di funzionari provinciali

Le aree devono essere connesse

- è più semplice organizzare riunioni per il coordinamento
- aree sconnesse sono politicamente inaccettabili

Le diverse squadre devono avere un carico di lavoro simile

Un problema di Ottimizzazione Combinatoria

- il grafo non orientato $G = (V, E)$ descrive l'adiacenza geografica
- A_v è il sottoinsieme di attività che coinvolge $v \in V$ $(A = \bigcup_{v \in V} A_v)$
- k è il numero delle squadre e W il massimo carico di lavoro
- q_a è il carico di lavoro richiesto dall'attività $a \in A$

Il carico di lavoro richiesto dai comuni dell'area $U \subseteq V$ è

$$w_U = \sum_{a \in A_U} q_a \quad \text{dove } A_U = \bigcup_{v \in U} A_v$$

Il problema

Partizionare V in al massimo k sottoinsiemi U_ℓ $(\ell = 1, \ldots, k)$ tali che

- U_ℓ induca un sottografo connesso: ogni area è connessa
- $w_{U_\ell} \leq W$: il carico di lavoro di ogni squadra è ammissibile
- $\min \phi = \sum_{\ell=1}^{k} \sum_{a \in A_{U_\ell}} q_a - \sum_{a \in A} q_a$: il carico di lavoro duplicato è minimo

Formulazione estesa

- $\mathcal{U}^\ell$ è la collezione dei sottoinsiemi di V radicati in ℓ e ammissibili (contenenti ℓ, connessi e con $w_U \leq W$)
- ϕ_i^ℓ è il carico richiesto dal sottoinsieme i di $\mathcal{U}^\ell$
- $a_{iv} = 1$ se il nodo v appartiene al sottoinsieme i di $\mathcal{U}^\ell$ ($a_{iv} = 0$ altrimenti)

Sia $y_i^\ell = 1$ se la soluzione usa il sottoinsieme i di $\mathcal{U}^\ell$ ($y_i^\ell = 0$ altrimenti)

$$\min \phi = \sum_{\ell \in V} \sum_{i=1}^{|\mathcal{U}^\ell|} \phi_i^\ell y_i^\ell$$

$$\sum_{\ell \in V} \sum_{i=1}^{|\mathcal{U}^\ell|} a_{iv} y_i^\ell = 1 \qquad v \in V \qquad \text{(partizione)}$$

$$\sum_{\ell \in V} \sum_{i=1}^{|\mathcal{U}^\ell|} y_i^\ell \leq k \qquad \text{(cardinalità)}$$

$$y_i^\ell \in \{0,1\} \qquad \ell \in V, i = 1, \ldots, \left|\mathcal{U}^\ell\right| \qquad \text{(integralità)}$$

Il problema di pricing

Le variabili duali ottime

- π_v per i vincoli di partizione imposti su ogni comune $v \in V$ forniscono premi sui comuni
- π_0 per i vincoli di cardinalità forniscono un offset fisso

Un sottoinsieme ammissibile per una data radice ℓ è descritto fissando

- $\bar{x}_v = 1$ se il comune $v \in V$ è nel sottoinsieme ($\bar{x}_v = 0$ altrimenti)
- $\bar{z}_a = 1$ se il sottoinsieme include comuni coinvolti nell'attività $a \in A$ ($\bar{z}_a = 0$ altrimenti)
- $\bar{f}_{uv}$ è il flusso sull'arco orientato $(u, v) \in E'$ emesso dalla radice

con i seguenti obiettivo e vincoli

Il problema di pricing

$$\min \bar{\phi}^{\ell} = \sum_{a \in A} q_a \, \bar{z}_a - \sum_{v \in V} \pi_v \, \bar{x}_v - \pi_0 \qquad \text{(costo ridotto)}$$

$$\sum_{a \in A} q_a \bar{z}_a \leq W \qquad \text{(carico)}$$

$$\bar{x}_v \leq \bar{z}_a \qquad v \in V_\ell, \; a \in A_v \qquad \text{(attivazione)}$$

$$\bar{x}_\ell = 1 \qquad \text{(radici)}$$

$$\sum_{(\ell,v) \in E'} \bar{f}_{\ell v} = \sum_{v \in V_\ell \setminus \{\ell\}} \bar{x}_v \qquad \text{(generazione del flusso)}$$

$$\sum_{(u,v) \in E'} \bar{f}_{uv} - \sum_{(v,w) \in E'} \bar{f}_{vw} = \bar{x}_v \qquad v \in V_\ell \setminus \{\ell\} \qquad \text{(conservazione del flusso)}$$

$$\sum_{(u,v) \in E'} \bar{f}_{\ell v} \leq (|V_\ell| - 1) \, \bar{x}_v \qquad v \in V_\ell \setminus \{\ell\} \qquad \text{(relazione flussi-nodi)}$$

$$\bar{x}_v \in \{0,1\} \qquad v \in V_\ell$$

$$\bar{z}_a \in \{0,1\} \qquad a \in A$$

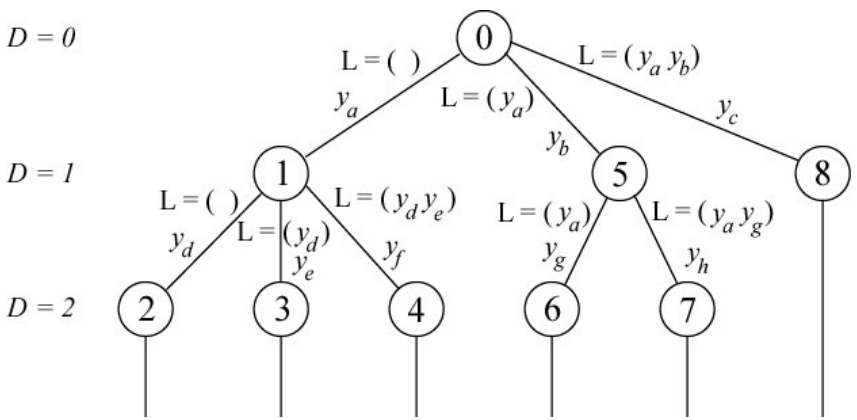
$$\bar{f}_{uv} \geq 0 \qquad (u,v) \in E'$$

più vincoli logici ottenuti combinando connessione e carico massimo di lavoro

Euristica di column generation

La Least Discrepancy Search per l'*HAP* procede come segue

- mantiene una lista L di variabili provate a 1 e attualmente poste a 0
- fissa a 1 la variabile y_i^ℓ più vicina a 1
- riottimizza il RMP ed (euristicamente) il PP^ℓ per ogni $\ell \in V$
- diving: fissa e riottimizza finché possibile
- fa backtrack finché le variabili fissate a 1 sono $< D_{\max}$ e
  le variabili fissate a 0 sono $< L_{\max}$
  - fissa a 0 e sposta in L la variabile precedentemente fissata a 1
  - fissa a 1 la variabile seguente
- quando il backtracking non è più possibile, termina
  e risolve il master ristretto con tutte le variabili generate nel processo



Bibliografia

- J. Desrosiers, M. E. Lübbecke, (2005). A Primer in Column Generation. Chap. 1 in: G. Desaulniers, J. Desrosiers, M. M. Solomon, Column Generation, Kluwer
- R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, E. Uchoa, (2019). Primal Heuristics for Branch-and-Price: The Assets of Diving Methods. *INFORMS Journal on Coputing*, 31 (2), 251–267.
- F. Colombo, R. Cordone, M. Trubian, (2014). Column-generation based bounds for the Homogeneous Areas Problem. *European Journal of Operational Research*, 236, 695–705.