

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/2022-mh/2022-mh.html>

Lezione 6: Euristiche basate su column generation

Milano, A.A. 2021/22

Programmazione Lineare (PL): forma standard

Ogni problema di Programmazione Lineare si può porre in **forma canonica**

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

che deriva dalla forma introdotta in precedenza

- trasformando le disuguaglianze in uguaglianze con **variabili di slack**

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

- trasformando le variabili libere in coppie di variabili non negative

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

Proprietà della forma canonica

Posto un problema in forma canonica

- le variabili \hat{x} si chiamano **variabili di base**
- i costi $[c^T \ 0^T]$ si chiamano **costi ridotti**

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

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

Il *tableau*

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

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

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

$-d$	c^T	0
b	A	I

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

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

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

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

Problema duale

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

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 lo stesso valore $-d$
- se $b \geq 0$, la soluzione primale è ammissibile, la duale superottima
- se $c \geq 0$, la soluzione primale è superottima, la duale ammissibile

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**)

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

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

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 simpleso più veloci** che per R
- **calcolare le variabili duali e i costi ridotti** delle variabili escluse da \bar{R}
(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

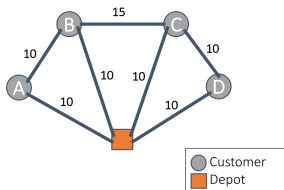
Se il risparmio nell'aggiornare il *tableau* supera il calcolo dei costi ridotti, l'algoritmo è più veloce

Esempio: Vehicle Routing Problem (VRP)

Dati

- un grafo orientato $G = (N, A)$ con un nodo deposito $d \in N$
- una funzione $c : A \rightarrow \mathbb{N}$ che assegna un costo ad ogni arco
- una funzione $w : N \rightarrow \mathbb{N}$ 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 $i \in N$
 $W = 2$

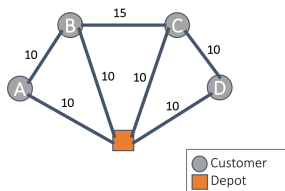
Definiamo come insieme base B quello dei circuiti basati su d e di peso $\leq W$

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

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

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

Esempio: Vehicle Routing Problem (VRP)



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

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

Esempio: Vehicle Routing Problem (VRP)

Partiamo con i soli circuiti dedicati

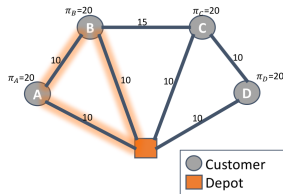
$$c \quad \begin{array}{|c|c|c|c|} \hline 20 & 20 & 20 & 20 \\ \hline \end{array} \quad y$$

$$A \quad \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \end{array} \quad \begin{array}{l} 20 \\ 20 \\ 20 \\ 20 \end{array}$$

$$x \quad 1 \quad 1 \quad 1 \quad 1$$

Il circuito che tocca i primi due nodi
ha costo ridotto $\tilde{c}_5 = 30 - 20 - 20 < 0$

\Rightarrow Va aggiunto al problema ristretto



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

Esempio: Vehicle Routing Problem (VRP)

$$c \quad \begin{array}{|c|c|c|c|c|} \hline 20 & 20 & 20 & 20 & 30 \\ \hline \end{array}$$

$$A \quad \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

$$x \quad 0 \quad 1 \quad 1 \quad 1 \quad 1$$

E così via...

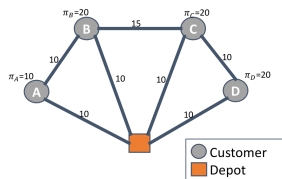
y

10

20

20

20



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

$W = 2$

Column generation per problemi esponenziali

Per problemi con moltissime variabili, non aggiornare il *tableau* è un risparmio enorme, ma ricalcolare tutti i costi ridotti rimane proibitivo

Il fatto è che **non occorre calcolarli tutti: basta trovarne uno negativo** e siccome esistono costi negativi se e solo se il più piccolo 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

- le variabili duali y_i sono note
- il costo c e i coefficienti a_j sono incogniti, ma non arbitrari: obbediscono alla definizione dell'insieme base B

Esempio: Vehicle Routing Problem (VRP)

Problema master

$$\begin{aligned} \min f(x) &= \sum_{p \in B} c_p x_p \\ \sum_{p \in B} a_{ip} x_p &= 1 \quad i \in N \\ x_p &\in \{0, 1\} \quad p \in B \end{aligned}$$

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 $i \in N$

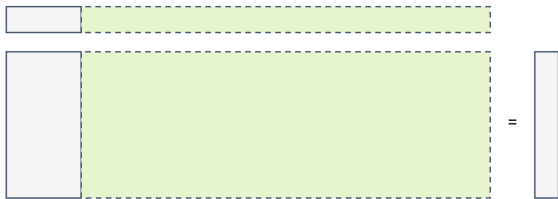
$b_{ij} = 1$: il circuito include $(i, j) \in A$

Problema di pricing

Resource-constrained Shortest Path

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

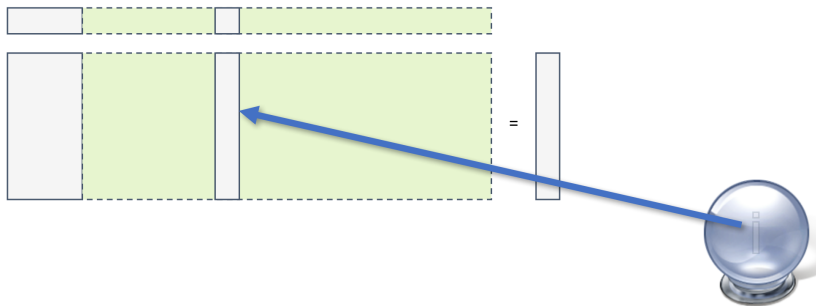
Column generation: schema



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

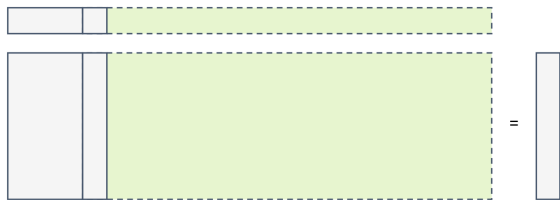
Column generation: schema



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

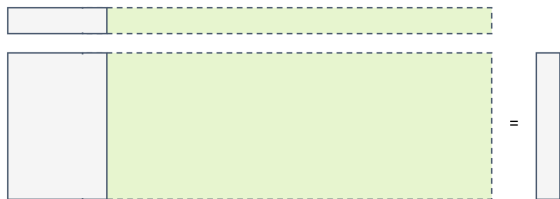
Column generation: schema



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

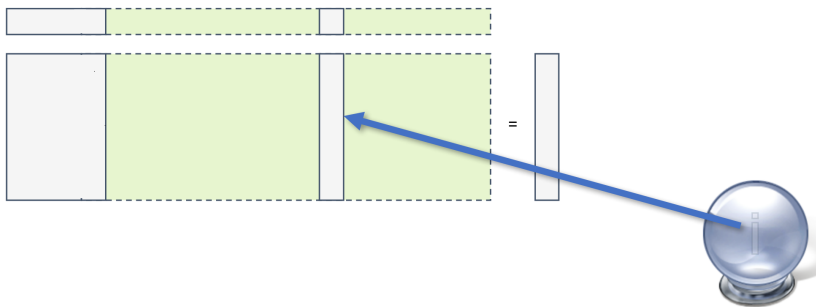
Column generation: schema



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

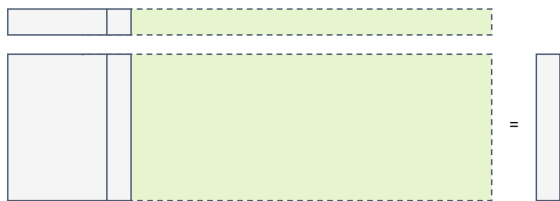
Column generation: schema



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

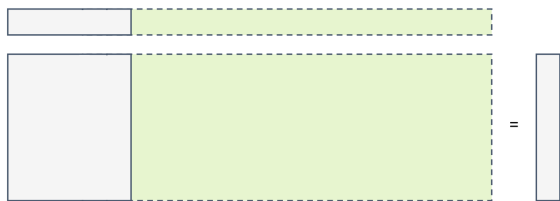
Column generation: schema



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

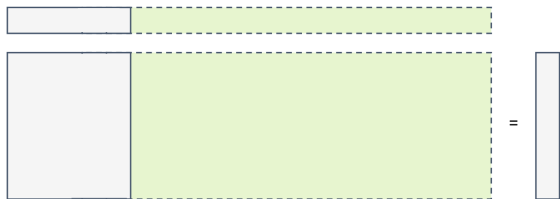
Column generation: schema



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

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
- euristiche
- chiusura del nodo, se dominato
- 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

Ovviamente, tutte le euristiche per il branch-and-bound si traducono automaticamente in euristiche per il branch-and-price

Le limitazioni del tempo, dei nodi e delle soluzioni individuate sono banali

L'uso di tolleranze impatta sul pricing oltre che sulla chiusura dei nodi

- 1 sia \bar{R} il problema master ristretto,
con ottimo $f_{\bar{R}}^*$ e costi ridotti \bar{c} (solo parzialmente noti)
- 2 sia v_{\max} il numero massimo di variabili in una soluzione ammissibile

Ogni variabile x_j fuori base può entrare in base al massimo con valore 1; se lo fa, riduce l'obiettivo di un valore massimo pari a $\bar{c}_j \leq 0$

Nel caso pessimo, le variabili della soluzione ottima sono tutte fuori base

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

La condizione di dominanza si può rilassare imponendola sul lower bound

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

che si traduce in una tolleranza sul pricing

$$c_j \geq \frac{f(\bar{x}) - f_{\bar{R}}^* - \Delta}{v_{\max}} \text{ anziché } c_j \geq 0$$

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

- 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
- si calcola un indice di confidenza ϕ_j (spesso semplicemente $1 - x_{\bar{R}j}$)
- si fissano le migliori δ variabili a 1 (tipicamente $\delta = 1$)
- si applica il presolve al sottoproblema e si risolve il rilassamento con una **nuova fase di column generation**
- se la soluzione è ammissibile, ma frazionaria, si torna al punto 1

La forza dello schema è che **le successive fasi di generazione di colonne tendono a evitare di ottenere problemi inammissibili o soluzioni cattive**

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 l

- 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}

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

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

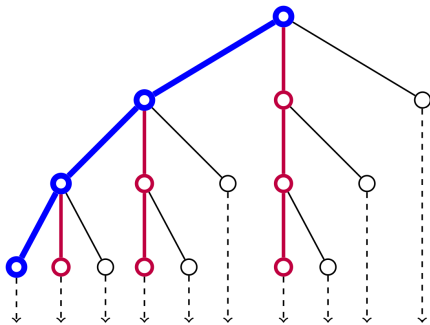
- 1 si crea una **lista di variabili vietate, inizialmente vuota** ($L := \emptyset$)
- 2 si esegue un'euristica di diving classica ottenendo \bar{x}
- 3 si risale la sequenza di variabili fissate (**backtracking**) sino al livello massimo d che abbia
 - indice $d < D_{\max}$
 - $|L| < L_{\max}$ **variabili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

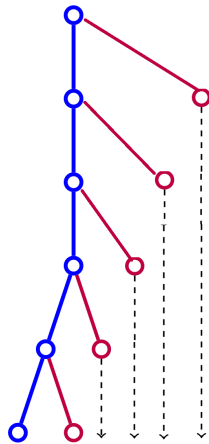
La LDS fissa variabili a 1, ma anche a 0:

- 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*)



$\text{maxDepth} = 3$

$\text{maxDiscrepancy} = 2$



$\text{maxDepth} = 5$

$\text{maxDiscrepancy} = 1$

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 ($\ell = 1, \dots, k$) 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}^ℓ is the 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}^ℓ
- $a_{iv} = 1$ se il nodo v appartiene al sottoinsieme i di \mathcal{U}^ℓ ($a_{iv} = 0$ altrimenti)

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

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

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

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

$$y_i^\ell \in \{0, 1\} \quad \ell \in V, i = 1, \dots, |\mathcal{U}^\ell| \quad (\text{Integralità})$$

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

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

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

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

$$x_\ell = 1 \quad (\text{Radici})$$

$$\bar{f}_{\ell v} = \sum_{v \in V_\ell \setminus \{\ell\}} \bar{x}_v \quad (\text{Generazione del flusso})$$

$$\sum_{(\ell, v) \in E'} \bar{f}_{\ell v} = \sum_{v \in V_\ell \setminus \{\ell\}} \bar{x}_v \quad (\text{Relazione flussi-nodi})$$

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

$$\bar{x}_v \in \{0, 1\}$$

$$v \in V_\ell$$

$$\bar{z}_a \in \{0, 1\}$$

$$a \in A$$

$$\bar{f}_{uv} \geq 0$$

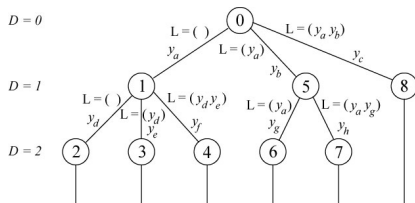
$$(u, v) \in E'$$

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**



- 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 Computing*, 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.