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 7: Euristiche per raffinare l'albero di branching Milano, A.A. 2021/22

Euristiche primali di inizializzazione

I metodi di branch-and-bound (e i risolutori di *PLI*) usano euristiche per

1. ottenere soluzioni ammissibili velocemente
   - per cominciare subito a chiudere nodi dominati
   - per fermarsi subito e restituire qualcosa di utile all'utente
2. raggiungere velocemente soluzioni migliori
   - per accelerare la chiusura dei nodi dominati
   - per restituire prima qualcosa di buono all'utente

Il primo aspetto è particolarmente importante quando
trovare una soluzione ammissibile è di per sé un problema difficile

Recentemente ci si è concentrati su metodi per

1. ricavare informazioni sui nodi la cui ammissibilità è dubbia
2. costruire soluzioni ammissibili a partire da rilassamenti
   anche quando non è elementare farlo

Feasibility Pump (*FP*)

L'euristica Feasibility Pump (*FP*) cerca di trovare soluzioni ammissibili nei casi in cui il semplice arrotondamento non è sufficiente

La sua idea fondamentale è che

1. la soluzione frazionaria x_R^* del rilassamento continuo R del problema
2. il vettore intero arrotondato $\bar{x}_R$ (non ammissibile!)

contengano entrambi informazione utile sulla soluzione ottima x^*

Lo schema che ne deriva è il seguente

1. si calcola $x^* := x_R^*$
2. si calcola $\bar{x}$ arrotondando x^*
3. si risolve il problema ausiliario di proiezione $PR_{\bar{x}}$

$$\begin{aligned} &\min \Delta(x, \bar{x}) \\ &Ax \leq b \end{aligned}$$

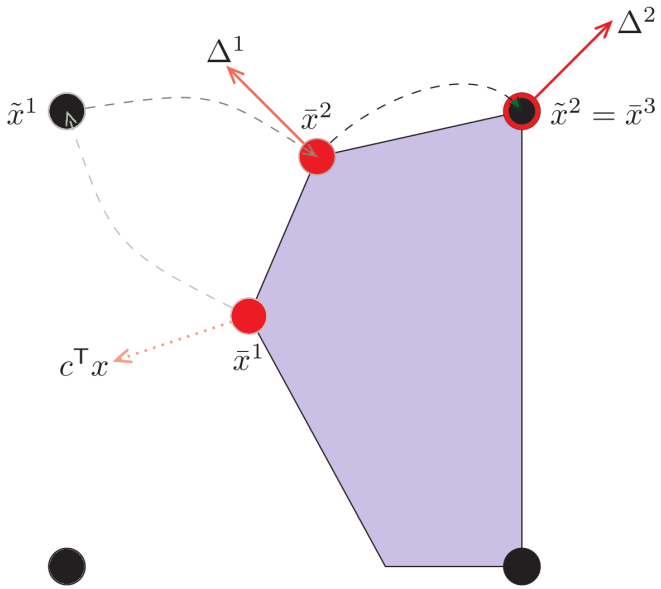
   che cerca la soluzione rilassata più vicina a quella intera arrotondata

4. si pone $x^* := x_{PR_{\bar{x}}}^*$
5. Se x^* è frazionaria, si torna al punto 2; altrimenti, si termina

Feasibility Pump (*FP*)

In breve, la Feasibility Pump alterna proiezioni e arrotondamenti

1. calcola $\bar{x}$ arrotondando x^*
2. calcola x^* proiettando $\bar{x}$



Si generano due sequenze complementari di punti in parte ammissibili

- i punti x^* rispettano i vincoli $Ax \leq b$
- i punti $\bar{x}$ rispettano i vincoli di integralità

La speranza è che convergano

Il nome "pompa" allude all'idea di compensare "pressioni" complementari

Il problema di proiezione

La natura del problema di proiezione dipende dalla definizione di distanza

La definizione più semplice da trattare è la norma ℓ_1

$$\Delta(x, \bar{x}) = \sum_{j \in B} |x_j - \bar{x}_j|$$

che rende il problema linearizzabile, dato che per vettori x e $\bar{x}$ binari

$$|x_j - \bar{x}_j| = (1 - \bar{x}_j) x_j + \bar{x}_j (1 - x_j)$$

Quindi $PR_{\bar{x}}$ è il problema di PL

$$\min \Delta(x, \bar{x}) = \sum_{j \in B} [(1 - \bar{x}_j) x_j + \bar{x}_j (1 - x_j)]$$

$$Ax \leq b$$

Un primo difetto

Le due sequenze non sempre convergono a un punto intero ammissibile:

- si possono avere cicli
- se x^* è il punto più vicino a $\bar{x}$, si va in stallo

Si può aggiungere una fase di perturbazione con un parametro T che

- arrotonda in verso opposto un numero casuale di variabili estratto da $\{T/2, \ldots, 3T/2\}$
- sceglie tali variabili in modo da minimizzare la distanza $\Delta(x^*, \bar{x})$

$x^* :=$ argmin$\{c^T x : x \in P\}$;
if x^* is integer, return(x^*);
let $\overline{x} := [x^*]$ (= rounding of x^*);
while (time < TL) do
  $x^* :=$ argmin$\{\Delta(x, \overline{x}) : x \in P\}$;
  if x^* is integer, return(x^*);
  if $\overline{x} \neq [x^*]$ then
    $\overline{x} := [x^*]$
  else
    flip the TT = rand(T/2,3T/2) entries x_j ($j \in \mathcal{B}$) with highest $|x_j^* - \overline{x}_j|$
  endif
endwhile

Ma per risolvere i cicli occorrono perturbazioni più casuali

Interrompere i comportamenti ciclici

Per interrompere i comportamenti ciclici, ogni R passi si perturba $\bar{x}$

Per ogni variabile x_j

- si genera un numero casuale uniforme $\rho \in [-0.3, 0.7]$
- si arrotonda in verso opposto a $\bar{x}_j$ se $\left|x_j^* - \bar{x}_j\right| + \rho > 0.5$

Lavori recentissimi stanno dando basi teoriche all'interazione fra

1. arrotondamento
2. proiezione
3. perturbazione

Molti risolutori di *PLI* incorporano una *feasibility pump*

- al nodo radice
- solo se altri metodi più veloci non funzionano

In GUROBI, il parametro `PumpPasses` fissa il numero di passi del metodo

Un secondo difetto

La proiezione considera la distanza, ma non la qualità delle soluzioni: quindi rischia di peggiorarle fortemente

La Objective Feasibility Pump rimedia combinando distanza e obiettivo

$$\min \Delta'_\alpha(x) = (1-\alpha)\frac{\Delta(x,\bar{x})}{\sqrt{|B|}} + \alpha\frac{f(x)}{||c||_2}$$

dove

- $\alpha \in [0,1]$ pesa l'obiettivo rispetto alla distanza
- α decresce gradualmente: $\alpha^{(t+1)} := \rho\alpha^{(t)}$
  in modo da favorire prima l'ottimalità poi l'ammissibilità
- i fattori $\sqrt{|B|}$ e $||c||_2$ normalizzano i due termini (*con una norma* ℓ_2)

Cambiare l'obiettivo aiuta anche a evitare comportamenti ciclici

Raffinamenti dello schema base (arrotondamento)

Diverse proposte mirano ad avvicinare x^* e $\bar{x}$ il più possibile, cercando di tener conto in una delle due fasi delle esigenze dell'altra

Per raffinare l'arrotondamento si è proposto di

- calcolare $\bar{x}$ arrotondando una variabile x_j^* per volta e applicando il presolve per limitare gli arrotondamenti inammissibili
  (*una specie di diving ridotto all'essenziale*)
- arrotondare alcune variabili e risolvere il problema di *PLI* residuo
  (*ovvero un diving con sub-MIP*)
- anziché x^*, arrotondare il centro analitico x^{ac} del poliedro $Ax \leq b$, cioè il punto di massima soddisfazione dei vincoli

$$x^{ac} = \arg \max_{x:Ax \leq b} \prod_{i=1}^{m} \left(b_i - A^{(i)} x \right) = \arg \min_{x:Ax \leq b} \sum_{i=1}^{m} - \log \left(b_i - A^{(i)} x \right)$$

  oppure arrotondare una combinazione convessa di x^* e x^{ac}
  (*partendo da dentro il poliedro, è più facile che $\bar{x}$ sia ammissibile*)

Raffinamenti dello schema base (proiezione)

Diverse proposte mirano ad avvicinare x^* e $\bar{x}$ il più possibile,
cercando di tener conto in una delle due fasi delle esigenze dell'altra

Per raffinare la proiezione si è proposto di

- usare una distanza pesata con coefficienti w_j che crescono al calare della frazionarietà della variabile x_j

$$\Delta_{x^*}(x, \bar{x}) = \sum_{j \in B} w_j\left(x_j^*\right) |x_j - \bar{x}_j|$$

(*così che* $\Delta_{x^*}(x, \bar{x})$ *approssimi il numero di variabili frazionarie*)

- generare piani di taglio per migliorare l'integralità di x^*
(*contrasta i comportamenti ciclici perché* x^* *cambia ad ogni passo*)

Il problema di proiezione diventa una sequenza di problemi di *PL*
con obiettivo o con vincoli leggermente diversi (aggiornabili passo passo)

Euristiche primali di miglioramento

I metodi di branch-and-bound (e i risolutori di *PLI*) usano euristiche per

1. ottenere soluzioni ammissibili velocemente
   - per cominciare subito a chiudere nodi dominati
   - per fermarsi subito e restituire qualcosa di utile all'utente
2. raggiungere velocemente soluzioni migliori
   - per accelerare la chiusura dei nodi dominati
   - per restituire prima qualcosa di buono all'utente

Il secondo aspetto è particolarmente importante quando si ritiene che la miglior soluzione nota possa essere ulteriormente migliorata

Recentemente ci si è concentrati su metodi per

1. guidare il processo di branch-and-bound verso soluzioni migliori della soluzione corrente $\bar{x}$
2. sfruttare regole di branching non standard per esplorare meglio lo spazio delle soluzioni

Local Branching (LB)

La strategia Local Branching (LB) è una regola di branching basata su $\bar{x}$, anziché sulla soluzione frazionaria del rilassamento; l'idea è che

- soluzioni vicine a $\bar{x}$ sono più promettenti,
  e vanno quindi esplorate per prime
- o eventualmente il contrario (secondo il problema)

Si può quindi partizionare il problema in due sottoproblemi contenenti:

- le soluzioni a distanza di Hamming $\leq k$ da $\bar{x}$

$$\sum_{j \in B} [\bar{x}_j \, (1 - x_j) + (1 - \bar{x}_j) \, x_j \,] \leq k$$

- le soluzioni a distanza di Hamming $> k$ da $\bar{x}$

$$\sum_{j \in B} [\bar{x}_j \, (1 - x_j) + (1 - \bar{x}_j) \, x_j \,] \geq k + 1$$

Problemi a cardinalità fissata

In molti problemi di Ottimizzazione Combinatoria
le soluzioni ammissibili hanno tutte la stessa cardinalità

Quindi la distanza di Hamming fra due soluzioni è sempre pari

$$\left|\bar{\xi} \setminus \xi\right| = \left|\xi \setminus \bar{\xi}\right| \Leftrightarrow \sum_{j \in B} \bar{x}_j \, (1 - x_j) = \sum_{j \in B} (1 - \bar{x}_j) \, x_j$$

e si può dividere il problema in

1. soluzioni x con al massimo k elementi non contenuti in $\bar{x}$

$$\sum_{j \in B} (1 - \bar{x}_j) \, x_j = \sum_{j \in B \setminus \bar{x}} x_j \leq k$$

2. soluzioni x con almeno $k + 1$ elementi non contenuti in $\bar{x}$

$$\sum_{j \in B} (1 - \bar{x}_j) \, x_j = \sum_{j \in B \setminus \bar{x}} x_j \geq k + 1$$

dove k è la metà della distanza di Hamming limite

Local Branching e strategia di visita

Generati i due nodi

- se si vuole intensificare la ricerca perché si considera $\bar{x}$ promettente, si esplora prima il nodo con le soluzioni vicine a $\bar{x}$
- se si vuole diversificare la ricerca perché si considera $\bar{x}$ già sfruttata, si esplora prima il nodo con le soluzioni lontane da $\bar{x}$

La scelta più frequente è la prima

Limitato al nodo radice, il Local Branching si può vedere come una forma molto flessibile di diving con sub-MIP che

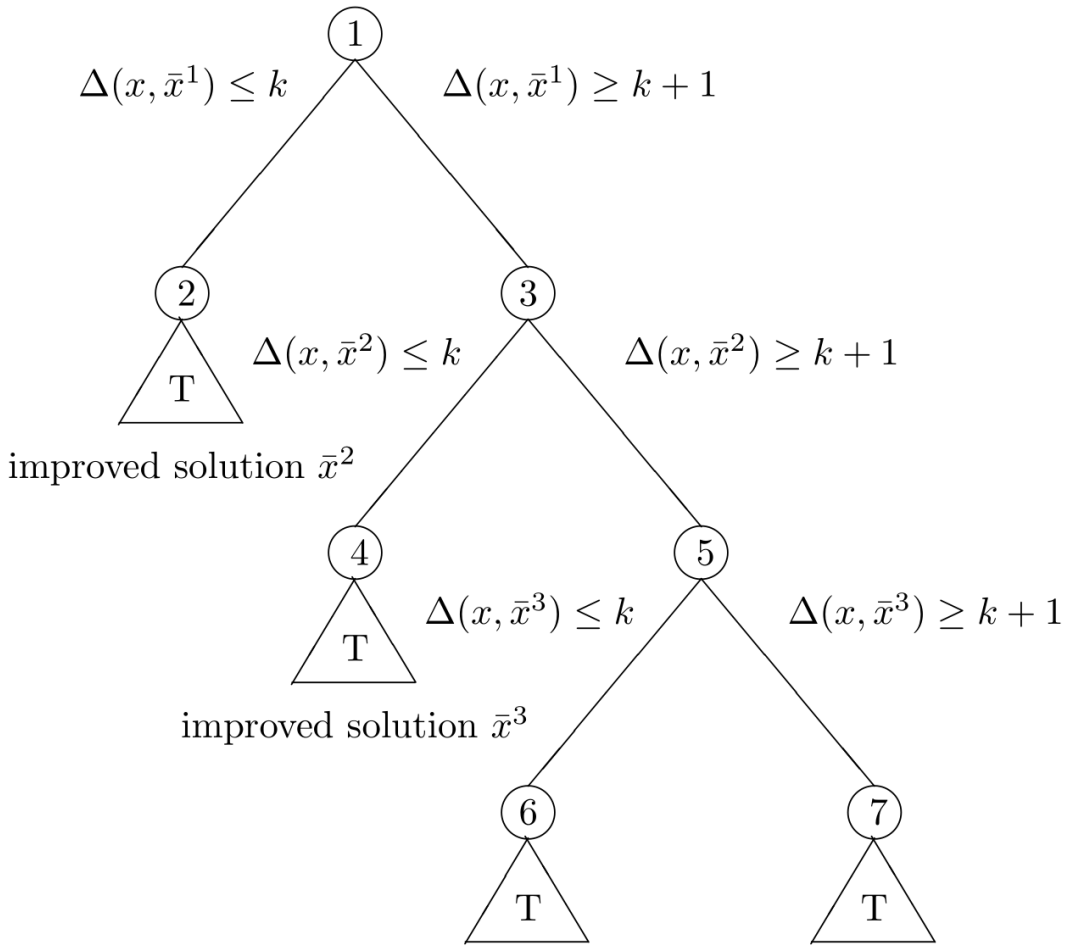
- sceglie quante variabili fissare
- anziché quali variabili fissare

prima di dedicarsi a risolvere il problema ridotto

Applicato senza limitazioni, il Local Branching è un metodo esatto

Schema di branching

Ogni operazione di branching richiede una diversa soluzione $\bar{x}$



E se non si ha una soluzione ammissibile? Vedremo poi

Gestione del parametro k

Se si visita per primo il nodo con le soluzioni vicine a $\bar{x}$,
la distanza limite di Hamming k va fissata tenendo conto che

1. valori bassi riducono il numero di soluzioni del nodo
   e quindi riducono la qualità
2. valori alti aumentano il numero di soluzioni del nodo
   e quindi aumentano il tempo di calcolo

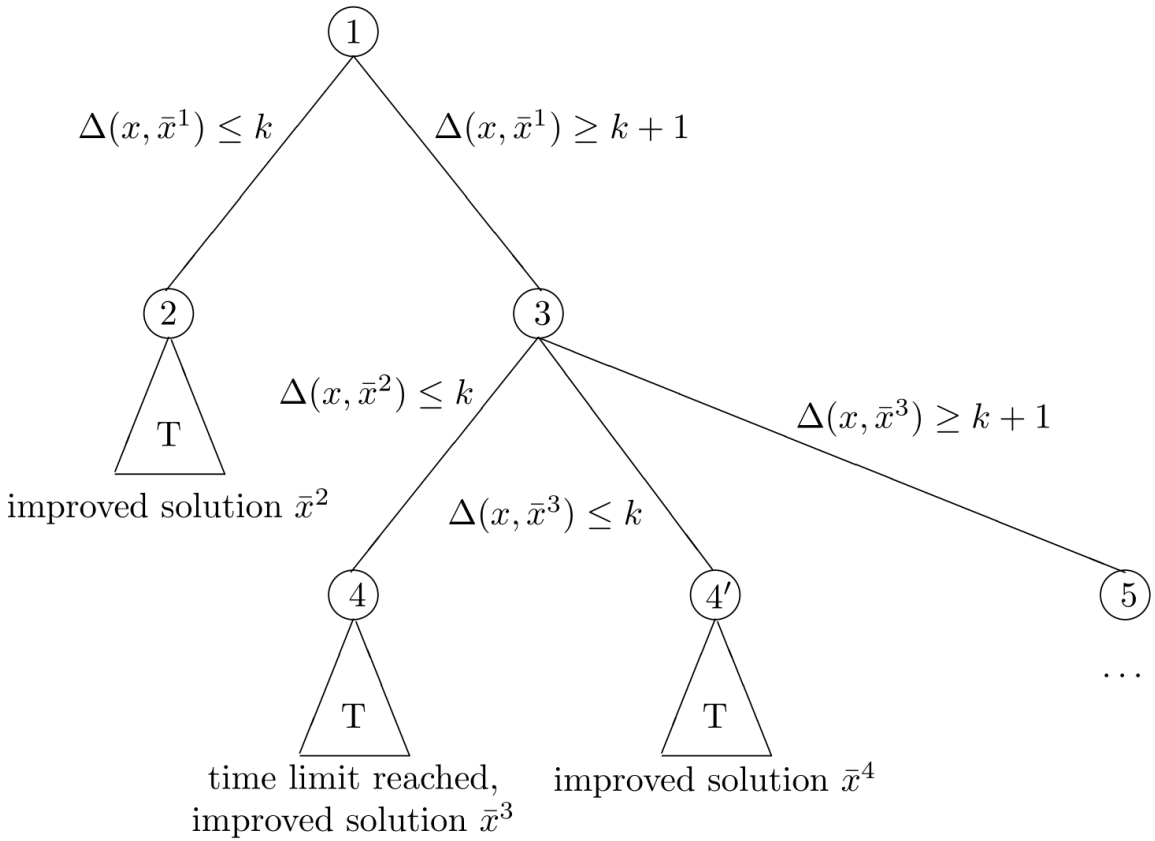
Ovviamente il valore corretto di k cambia nodo per nodo

È possibile seguire un approccio adattivo, cioè

- fissare k piuttosto alto
- ma limitare il tempo di elaborazione del nodo
- se il nodo viene risolto nel tempo assegnato, chiuderlo regolarmente;
  altrimenti, tornare al nodo padre e provare un branching diverso
  1. se si è ottenuta una soluzione $\bar{x}'$ migliorante $(f(\bar{x}') < f(\bar{x}))$,
     allora si partiziona il nodo padre in base a $\bar{x}'$ con distanza k
  2. altrimenti, si partiziona il nodo padre in base a $\bar{x}$ con distanza $k' < k$
     (*ad esempio, si può porre* $k' = \lfloor k/\alpha \rfloor$ *con* $\alpha > 1$)

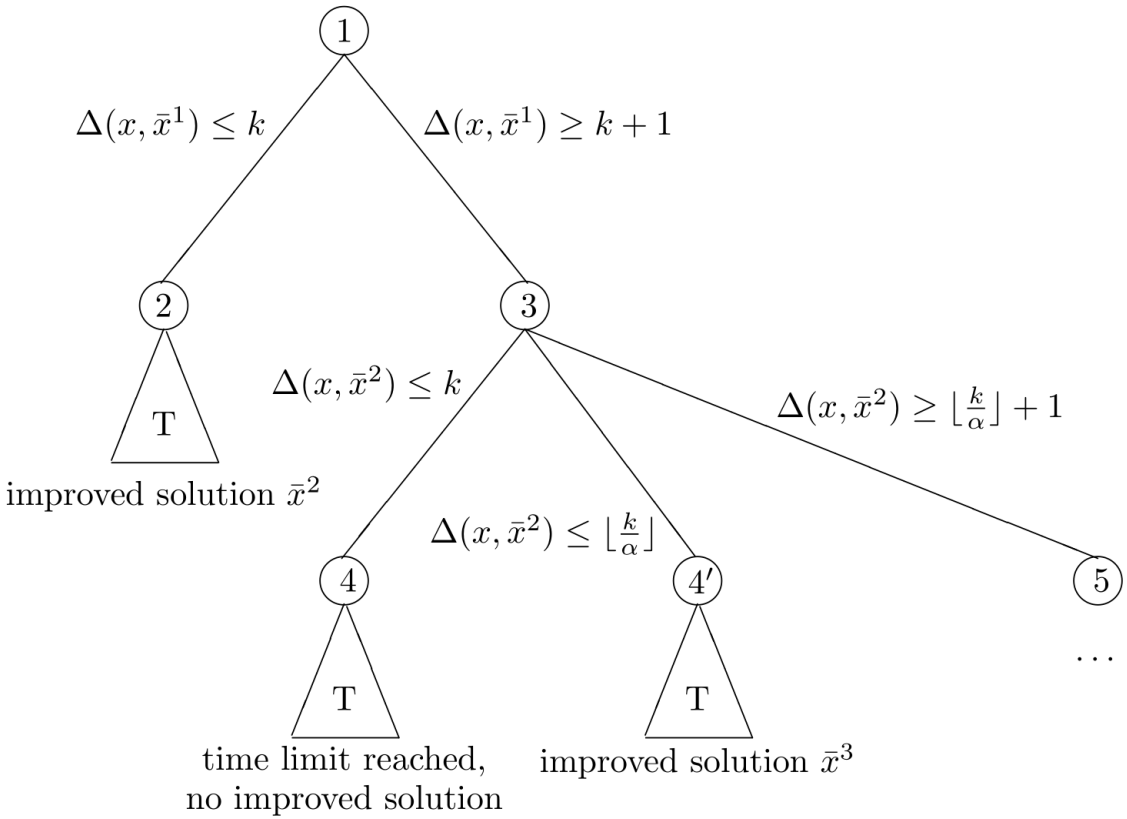
Gestione del parametro k

Se si ottiene $\bar{x}'$ con $f(\bar{x}') < f(\bar{x})$,
si partiziona il nodo padre in base a $\bar{x}'$ con distanza k



Gestione del parametro k

Se non si ottengono soluzioni migliori di $\bar{x}$,
si partiziona il nodo padre in base a $\bar{x}$ con distanza $k' < k$ (*ad es.*, k/α)



Combinare Feasibility Pump e Local Branching

Ogni nodo richiede una diversa soluzione ammissibile $\bar{x}$

E se non se ne conosce una?

Qualsiasi vettore intero $\bar{x}$ fornisce una partizione lecita, salvo che uno inammissibile è meno promettente per la ricerca di soluzioni buone

L'alternativa è riparare soluzioni inammissibili, purché

1. di buona qualità
2. vicine all'ammissibilità

Un modo di farlo è combinare Feasibility Pump e Local Branching

Supponiamo che

- la Feasibility Pump abbia restituito una soluzione $\bar{x}$ non ammissibile
- l'insieme $T_{\bar{x}}$ dei vincoli violati in $\bar{x}$ sia piuttosto piccolo

Sembra promettente l'idea di esplorare un intorno di $\bar{x}$ per trovare $\bar{x}' \in X$

Combinare Feasibility Pump e Local Branching

Per ricondurre il vettore $\bar{x}$ all'ammissibilità con modifiche minime

1. riformuliamo la regione ammissibile

   da $A^{(i)} x \leq b_i$ per $i = 1, \ldots, m$ a
   $$\begin{cases} A^{(i)} x \leq b_i & \text{per } i \notin T_{\bar{x}} \\ A^{(i)} x - s_i \leq b & \text{per } i \in T_{\bar{x}} \\ s_i \geq 0 & \text{per } i \in T_{\bar{x}} \\ s_i \leq M y_i & \text{per } i \in T_{\bar{x}} \\ y_i \in \{0, 1\} & \text{per } i \in T_{\bar{x}} \end{cases}$$

   dove le variabili binarie y_i indicano i vincoli violati
2. sostituiamo l'obiettivo originale $f(x) = c^T x$ con
   $$\min v(x, y) = \sum_{i \in T_{\bar{x}}} y_i$$

   che è il numero dei vincoli violati
3. $(\bar{x}, \mathbf{1})$ è ammissibile per il nuovo problema e vale $v(\bar{x}, \mathbf{1}) = |T|$

Combinare Feasibility Pump e Local Branching

$$\min v(x,y) = \sum_{i \in T} y_i$$

$$\begin{aligned} A^{(i)} x &\leq b_i && i \notin T \\ A^{(i)} x - s_i &\leq b && i \in T \\ s_i &\geq 0 && i \in T \\ s_i &\leq M y_i && i \in T \\ y_i &\in \{0,1\} && i \in T \\ x_j &\in \{0,1\} && j \in B \end{aligned}$$

4. applichiamo la strategia LB al problema di minimizzare il numero dei vincoli violati usando $\bar{x}$ come soluzione di riferimento
5. se si trova una soluzione di valore nullo, si torna al problema iniziale: le variabili x_j formano una soluzione ammissibile

Relaxation Induced Neighbourhood Search (*RINS*)

L'euristica Relaxation Induced Neighbourhood Search (*RINS*) si basa sull'idea di combinare l'informazione della miglior soluzione corrente $\bar{x}$ e della soluzione del rilassamento x^*

Si tratta di

1. calcolare una soluzione euristica $\bar{x}$
2. calcolare la soluzione ottima x^* del rilassamento continuo
3. confrontare x^* e $\bar{x}$ e calcolare

$$J_{x^*,\bar{x}} = \left\{ j \in B : x_j^* = \bar{x}_j \right\}$$

   cioè l'insieme delle variabili che hanno ugual valore in x^* e $\bar{x}$
4. fissare $x_j = \bar{x}_j$ per ogni $j \in J_{x^*,\bar{x}}$, vale a dire
   fidarsi delle informazioni su cui euristica e rilassamento concordano
5. risolvere il problema ridotto col branch-and-bound (magari limitato);
   se si trova una soluzione migliore, aggiornare $\bar{x}$ e tornare al punto 3

Relaxation Induced Neighbourhood Search

Anche se può sembrare a prima vista, la *RINS* non è un sub-MIP

- in genere $\bar{x}$ non rispetta i vincoli di branching imposti su x^*, dato che non appartiene allo stesso sottoalbero di branching

La *RINS* è spesso molto efficace, per diverse ragioni

1. esplora regioni diverse dello spazio delle soluzioni, perché x^* cambia in ogni nodo
2. non richiede di generare continuamente nuove soluzioni euristiche $\bar{x}$ perché ogni nodo ha una diversa soluzione rilassata x^*
3. in genere il problema ridotto è piccolo e veloce da risolvere, dato che i vincoli aggiuntivi riducono il numero di variabili
   (*purché x^* e $\bar{x}$ non siano completamente diverse*)
4. spesso il problema ridotto cambia natura, perché poche decisioni chiave lo semplificano brutalmente (per esempio, si decompone in sottoproblemi indipendenti)

I risolutori di *PLI* spesso consentono di applicare la *RINS* (in *GUROBI* il parametro `RINS` indica ogni quanti nodi farlo)

(*non si fa ad ogni nodo perché le x^* sono troppo simili*)

Bibliografia

- M. Fischetti, F. Glover, A. Lodi, (2005). The feasibility pump. *Mathematical Programming, Series A*, 104, 91–104.
- T. Berthold, A. Lodi, D. Salvagnin, (2019). Ten years of feasibility pump, and counting. *EURO Journal on Computational Optimization*, 7 (1), 1–14.
- M. Fischetti, A. Lodi, (2003). Local branching. *Mathematical Programming*, 98 (1–3), 23–47.
- M. Fischetti, A. Lodi, D. Salvagnin, (2010). Just MIP it! Chap. 2, 39–70 in: V. Maniezzo, T. Stützle, S. Voß. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. Springer.
- E. Danna, E. Rothberg, C. Le Pape, (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102 (1), 71–90.