

Euristiche MIP per i problemi classici

Indice

Perché le euristiche MIP per i problemi classici?

Euristiche basate sulla PLI

I problemi e i loro modelli MIP

Arrotondamento

Diving

La feasibility pump

RINS, local branching e proximity search

RINS

Local branching

Proximity search

Relax-and-fix e fix-and-optimize sul lot sizing

Le euristiche dentro il solver

Che cosa danno le euristiche basate sulla PLI

Errori tipici

Implementazione in Python

In sintesi

Obiettivi

Alla fine di questo capitolo saremo in grado di:

- ▶ scrivere i modelli MIP dei problemi classici nella forma richiesta dalle euristiche basate sulla PLI e calcolarne i rilassamenti lineari;
- ▶ eseguire a mano arrotondamento, diving, feasibility pump, RINS, local branching e proximity search sul problema dello zaino 0/1, sul bin packing e su $P || C_{\max}$;
- ▶ eseguire relax-and-fix e fix-and-optimize su un'istanza di lot sizing;
- ▶ spiegare perché ciascun metodo riesce o fallisce su ciascun problema;
- ▶ calcolare il gap di una soluzione euristica quando è disponibile un bound duale.

Prerequisiti (1/2)

- ▶ Dal modulo *Programmazione lineare intera*, il capitolo «**Euristiche MIP**»: i metodi di questo capitolo (arrotondamento, diving, feasibility pump, RINS, local branching, proximity search, relax-and-fix e fix-and-optimize), i loro pseudocodici e le condizioni in cui funzionano.
- ▶ Da questo modulo, i capitoli sulle formulazioni dei problemi classici (parte «Formulazioni e loro proprietà»): i modelli del problema dello zaino 0/1, del problema del bin packing, del problema del commesso viaggiatore, di $P \parallel C_{\max}$ e del lot sizing.

Prerequisiti (2/2)

- ▶ Da questo modulo, i capitoli sugli algoritmi euristici, da «Algoritmi euristici: definizioni» a «Metaeuristiche»: le euristiche costruttive (*GR*, *FF*, *NN*, *LPT*), la ricerca locale e le metaeuristiche, le cui soluzioni sono i punti di partenza di questo capitolo.

Perché le euristiche MIP per i problemi classici?

Introduzione: Perché le euristiche MIP per i problemi classici?

- ▶ Abbiamo già incontrato due modi molto diversi di cercare buone soluzioni
- ▶ In questo capitolo li mettiamo insieme: i metodi basati su un modello MIP sono applicati ai problemi classici di questo modulo
- ▶ Confrontiamo che cosa i due approcci sanno dire sulla qualità delle loro soluzioni

Dai metodi ai problemi classici (1/2)

- ▶ Gli algoritmi euristici dei capitoli precedenti (euristiche costruttive, ricerca locale e metaeuristiche) lavorano direttamente su una rappresentazione delle soluzioni:
 - ▶ un vettore binario,
 - ▶ un vettore *bin*,
 - ▶ un giro,
 - ▶ un vettore *mach*.
- ▶ Non guardano mai un modello matematico del problema, e non usano mai un solver.

Dai metodi ai problemi classici (2/2)

- ▶ Le euristiche basate sulla PLI del capitolo «**Euristiche MIP**» del modulo *Programmazione lineare intera* partono invece dal modello del problema.
- ▶ Usano il suo rilassamento lineare e un solver MIP.
- ▶ In questo capitolo le applichiamo ai problemi classici, con le istanze di riferimento dei capitoli precedenti.

Due famiglie di euristiche:

- ▶ euristiche costruttive, ricerca locale, metaeuristiche: lavorano su una rappresentazione delle soluzioni, senza un modello e senza un solver
- ▶ euristiche basate sulla PLI: lavorano sul modello, con il suo rilassamento lineare e un solver MIP

Questo capitolo:

- ▶ le euristiche basate sulla PLI sui problemi classici e sulle loro istanze di riferimento

Euristiche basate sulla PLI

Introduzione: Euristiche basate sulla PLI

- ▶ Definiamo che cos'è un'euristica basata sulla PLI
- ▶ Richiamiamo la notazione del capitolo «Euristiche MIP» del modulo *Programmazione lineare intera*, che useremo sempre
- ▶ Diciamo come sono calcolati gli esempi di questo capitolo

Definizione 1: Euristică basata sulla PLI

*Un algoritmo euristico per un problema con un modello MIP è **basato sulla PLI** (una **matheuristica**) se usa il rilassamento lineare del modello e un solver MIP per cercare una soluzione ammissibile.*

- ▶ Le euristiche basate sulla PLI sono definite nel modulo *Programmazione lineare intera*, capitolo «[Euristiche MIP](#)».
- ▶ In questo capitolo le applichiamo ai problemi classici, con le istanze di riferimento dei capitoli precedenti.

Notazione

La notazione è quella del capitolo «Euristiche MIP»:

- ▶ un MIP con variabili intere $\mathcal{D}$;
- ▶ $\tilde{\mathbf{x}}$ è una soluzione ottima del rilassamento lineare;
- ▶ $\bar{\mathbf{x}}$ è una soluzione ammissibile (la soluzione incumbent);
- ▶ $\lceil v \rceil = \lfloor v + \frac{1}{2} \rfloor$ è l'arrotondamento all'intero più vicino;
- ▶ $\Delta(\mathbf{x}, \bar{\mathbf{x}}) = \sum_{j \in \mathcal{D}} |x_j - \bar{x}_j|$ è la distanza da un punto intero.

Un problema di massimo, come il KP, si tratta scambiando min e max, $\leq$ e $\geq$ nei confronti fra valori obiettivo.

Come sono calcolati gli esempi

- ▶ Gli esempi sono risolti con il solver MIP Gurobi, usato da Python tramite `gurobipy`.
- ▶ Quando un LP ha più soluzioni ottime, il testo riporta quella restituita dal solver.
- ▶ I codici sono nell'ultima sezione.

Rappresentazione o modello

Le euristiche costruttive, la ricerca locale e le metaeuristiche lavorano su una rappresentazione delle soluzioni;
un'euristica basata sulla PLI lavora sul modello del problema, attraverso il suo rilassamento lineare e un solver MIP.

Euristiche basate sulla PLI:

- ▶ una matheuristica usa il rilassamento lineare del modello e un solver MIP
- ▶ sono definite nel modulo *Programmazione lineare intera*

Notazione:

- ▶ $\tilde{x}$ soluzione LP, $\bar{x}$ incumbent, $[v]$ intero più vicino, $\Delta(x, \bar{x})$ distanza
- ▶ gli esempi sono risolti con Gurobi da Python

I problemi e i loro modelli MIP

Introduzione: I problemi e i loro modelli MIP

- ▶ Richiamiamo i quattro problemi classici, le loro istanze di riferimento e i valori trovati dalle euristiche dei capitoli precedenti
- ▶ Richiamiamo i modelli MIP su cui lavorano i metodi di questo capitolo
- ▶ Guardiamo i rilassamenti lineari sulle istanze di riferimento: sono il punto di partenza di ogni metodo

I quattro problemi

- ▶ Le euristiche sono illustrate sui quattro problemi dei capitoli precedenti:
 - ▶ il problema dello zaino 0/1 (KP),
 - ▶ il problema del bin packing (BPP),
 - ▶ il problema del commesso viaggiatore (TSP),
 - ▶ $P || C_{\max}$.

Le loro definizioni e ipotesi sono nelle rispettive pagine.

- ▶ Le istanze di riferimento sono richiamate nella tabella seguente, con i valori calcolati nei capitoli precedenti sugli algoritmi euristici.

Le istanze di riferimento

problema	istanza di riferimento	valori
KP	$p = (10, 7, 6, 4),$ $w = (5, 4, 3, 3),$ $c = 9$	$GR: \{1, 3\}$, profitto 16; $z(I) = 17, \{1, 2\}$
BPP	$w = (4, 4, 5, 3, 2, 3),$ $c = 7$	$FF: 4$ contenitori; $z(I) = 3$
TSP	5 città, le distanze di riferimento	NN dalla città 1: 25; $z(I) = 18$
$P \parallel C_{\max}$	$t = (5, 5, 4, 4, 3, 3, 3),$ $m = 3$	$LPT: J_1 = \{1, 5, 7\},$ $J_2 = \{2, 6\}, J_3 = \{3, 4\},$ 11; $z(I) = 9$

I modelli MIP

- ▶ I modelli MIP sono quelli della parte «Formulazioni e loro proprietà»:
 - ▶ il modello ILP_{KP} del capitolo «Formulazioni dei problemi di zaino»,
 - ▶ il modello ILP_{BPP} del capitolo «Formulazioni dei problemi di packing»,
 - ▶ il modello ILP_{TSP}^{DFJ} del capitolo «Formulazioni dei problemi di routing»,
 - ▶ il modello $ILP_{P||C_{max}}$ del capitolo «Formulazioni dei problemi di scheduling».
- ▶ Richiamiamo i tre modelli usati negli esempi, con le variabili e la notazione di quei capitoli.

Il problema dello zaino 0/1

- La variabile binaria x_j vale 1 se l'oggetto j è selezionato nello zaino, e 0 altrimenti, per ogni $j \in \mathcal{J}$.

Usando queste variabili, un modello PLI per il problema dello zaino 0/1 (KP) è il seguente:

$$\begin{aligned} \max \quad & \sum_{j \in \mathcal{J}} p_j x_j \\ \text{soggetto a} \quad & \sum_{j \in \mathcal{J}} w_j x_j \leq c, \\ & x_j \in \{0, 1\}, \quad \forall j \in \mathcal{J}. \end{aligned}$$

Questo modello PLI si chiama modello ILP_{KP}.

Il problema del bin packing: variabili

- ▶ Dato un upper bound m sul numero di contenitori usati in una soluzione ottima, e l'insieme $\mathcal{B} = \{1, 2, \dots, m\}$ dei contenitori disponibili:
 - ▶ la variabile binaria x_{jb} vale 1 se l'oggetto j è inserito nel contenitore b ,
 - ▶ la variabile binaria y_b vale 1 se il contenitore b è usato,per ogni $j \in \mathcal{I}$ e $b \in \mathcal{B}$.
- ▶ Il contenuto del contenitore b è lo schema di riempimento $P_b = \{j \in \mathcal{I} : x_{jb} = 1\}$.
- ▶ Il numero n di oggetti è sempre un upper bound m valido; sull'istanza di riferimento usiamo $m = n = 6$.

Il problema del bin packing: modello

Usando queste variabili, un modello PLI per il problema del bin packing (BPP) è il seguente:

$$\begin{aligned} \min \quad & \sum_{b \in \mathcal{B}} y_b \\ \text{soggetto a} \quad & \sum_{b \in \mathcal{B}} x_{jb} = 1, & \forall j \in \mathcal{I}, \\ & - \sum_{j \in \mathcal{I}} w_j x_{jb} + c y_b \geq 0, & \forall b \in \mathcal{B}, \\ & x_{jb} \in \{0, 1\}, & \forall j \in \mathcal{I}, \forall b \in \mathcal{B}, \\ & y_b \in \{0, 1\}, & \forall b \in \mathcal{B}. \end{aligned}$$

Questo modello PLI si chiama modello ILP_{BPP} .

Il problema del commesso viaggiatore

- ▶ Il modello $\text{ILP}_{\text{TSP}}^{\text{DFJ}}$ ha:
 - ▶ una variabile binaria x_{ij} per ogni arco $(i, j) \in A$,
 - ▶ i vincoli sugli archi uscenti da ogni città ed entranti in ogni città,
 - ▶ i vincoli di eliminazione dei sottogiri sui sottoinsiemi $S \subseteq \mathcal{V} \setminus \{1\}$ con $|S| \geq 2$.
- ▶ Sull'istanza di riferimento, con $n = 5$ città, ha 20 variabili e $5 + 5 + 11 = 21$ vincoli.

Il problema di scheduling $P || C_{\max}$: variabili

- ▶ Dato l'insieme $\mathcal{H} = \{1, 2, \dots, m\}$ delle macchine:
 - ▶ la variabile binaria x_{jh} vale 1 se il lavoro j è assegnato alla macchina h , per ogni $j \in \mathcal{J}$ e $h \in \mathcal{H}$,
 - ▶ la variabile continua non negativa $C_{\max}$ è il makespan.
- ▶ $C_{\max}$ è l'unica variabile continua dei quattro modelli: dato l'assegnamento, il completamento la pone uguale al carico massimo $L_h = \sum_{j \in \mathcal{J}} t_j x_{jh}$.

Il problema di scheduling $P \parallel C_{\max}$: modello

Usando queste variabili, un modello PLI per $P \parallel C_{\max}$ è il seguente:

$$\begin{aligned} \min \quad & C_{\max} \\ \text{soggetto a} \quad & \sum_{h \in \mathcal{H}} x_{jh} = 1, & \forall j \in \mathcal{J}, \\ & - \sum_{j \in \mathcal{J}} t_j x_{jh} + C_{\max} \geq 0, & \forall h \in \mathcal{H}, \\ & x_{jh} \in \{0, 1\}, \quad \forall j \in \mathcal{J}, \quad \forall h \in \mathcal{H}, \\ & C_{\max} \geq 0. \end{aligned}$$

Questo modello PLI si chiama modello ILP $_{P \parallel C_{\max}}$.

Con i modelli a disposizione,
guardiamo i loro rilassamenti lineari
sulle istanze di riferimento.

I rilassamenti lineari sulle istanze di riferimento

- La tabella seguente riporta la dimensione dei quattro modelli sulle istanze di riferimento, il valore $z(LP)$ dei loro rilassamenti lineari, calcolato con Gurobi, e il valore ottimo $z(I)$.

problema	variabili	vincoli	$z(LP)$	$z(I)$	soluzione LP intera?
KP	4	1	17.75	17	no
BPP	42	12	3	3	no
TSP	20	21	18	18	sì
$P \parallel C_{\max}$	22	10	9	9	no

Rilassamenti forti

- ▶ Su tutte e quattro le istanze il valore LP, arrotondato nel verso del bound (per difetto per il KP, per eccesso per i tre problemi di minimo), è già il valore ottimo.
- ▶ Su queste istanze i rilassamenti sono *forti*: la difficoltà è trovare una soluzione che raggiunga il bound, non dimostrarlo.
- ▶ È la situazione tipica in cui le euristiche basate sulla PLI sono utili.

L'istanza di riferimento del TSP

- Sull'istanza di riferimento del TSP la soluzione LP del modello ILP_{TSP}^{DFJ} è essa stessa il giro ottimo $(1, 3, 5, 2, 4)$, con

$$\tilde{x}_{13} = \tilde{x}_{35} = \tilde{x}_{52} = \tilde{x}_{24} = \tilde{x}_{41} = 1 :$$

ogni metodo di questo capitolo lo trova subito.

- Il rilassamento lineare del modello ILP_{TSP}^{MTZ} è più debole: il suo valore è 17.
- Per questo il TSP non è usato negli altri esempi di questo capitolo.

Esempio: le soluzioni LP delle istanze di riferimento (1/3)

KP.

- ▶ La soluzione LP è

$$\tilde{\mathbf{x}} = (1, \frac{1}{4}, 1, 0).$$

- ▶ Gli oggetti sono presi per efficienza non crescente, e l'oggetto critico 2 è preso per la frazione $\frac{9-8}{4} = \frac{1}{4}$ del suo peso, quindi

$$z(\text{LP}) = 10 + \frac{7}{4} + 6 = 17.75.$$

Esempio: le soluzioni LP delle istanze di riferimento (2/3)

$P \parallel C_{\max}$. La soluzione LP restituita dal solver è (zeri omissi)

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
$\tilde{x}_{j1}$		$\frac{3}{5}$				1	1
$\tilde{x}_{j2}$			$\frac{1}{2}$	1	1		
$\tilde{x}_{j3}$	1	$\frac{2}{5}$	$\frac{1}{2}$				

con carichi

$$L_1 = 3 + 5 \cdot \frac{3}{5} + 3 = 9, \quad L_2 = 4 \cdot \frac{1}{2} + 4 + 3 = 9, \quad L_3 = 5 + 5 \cdot \frac{2}{5} + 4 \cdot \frac{1}{2} = 9$$

e $\tilde{C}_{\max} = 9 = \frac{27}{3}$. Il LP ha infinite soluzioni ottime: ogni assegnamento frazionario con tre carichi uguali a 9 è ottimo.

Esempio: le soluzioni LP delle istanze di riferimento (3/3)

BPP. La soluzione LP restituita dal solver:

- ▶ mette l'oggetto 1 nel contenitore 6, l'oggetto 2 nel contenitore 4,
- ▶ l'oggetto 3 per $\frac{4}{5}$ nel contenitore 3 e per $\frac{1}{5}$ nel contenitore 4,
- ▶ l'oggetto 4 nel contenitore 6, l'oggetto 5 nel contenitore 1, l'oggetto 6 nel contenitore 3,
- ▶ e apre i contenitori per le frazioni

$$\tilde{\mathbf{y}} = \left(\frac{2}{7}, 0, 1, \frac{5}{7}, 0, 1 \right),$$

appena sufficienti a contenerne i carichi: 2, 0, 7, 5, 0, 7.

Il suo valore è $\frac{21}{7} = 3$.

I modelli:

- ▶ ILP_{KP} , ILP_{BPP} , ILP_{TSP}^{DFJ} e $ILP_{P||C_{max}}$, dalla parte «Formulazioni e loro proprietà»
- ▶ C_{max} è l'unica variabile continua

I rilassamenti lineari sulle istanze di riferimento:

- ▶ forti: il valore LP arrotondato è già il valore ottimo
- ▶ la difficoltà è trovare una soluzione che raggiunga il bound
- ▶ sul TSP la soluzione LP è già il giro ottimo, quindi il TSP non è usato in seguito

Arrotondamento

Introduzione: Arrotondamento

- ▶ L'arrotondamento è il modo più semplice di trasformare la soluzione LP in un punto intero
- ▶ Lo applichiamo al KP, a $P \parallel C_{\max}$ e al BPP
- ▶ Vediamo quando funziona e perché fallisce sui problemi con vincoli di assegnamento

Arrotondamento

- ▶ L'*arrotondamento* arrotonda la soluzione LP al punto intero più vicino e completa le variabili continue; fallisce se il punto arrotondato viola un vincolo.
- ▶ L'arrotondamento per difetto è sempre ammissibile quando tutti i coefficienti di A sono non negativi e $b \geq 0$ (l'osservazione sull'arrotondamento per difetto del capitolo «[Euristiche MIP](#)» del modulo *Programmazione lineare intera*).
- ▶ Un vincolo di uguaglianza, come i vincoli di assegnamento del BPP e di $P \parallel C_{\max}$, è quasi sempre violato da arrotondamenti indipendenti.

Esempio: arrotondamento sulle istanze di riferimento (1/3)

KP.

- ▶ La soluzione LP $\tilde{x} = (1, \frac{1}{4}, 1, 0)$ è arrotondata a

$$\bar{x} = (1, 0, 1, 0) : \quad \text{peso } 8 \leq 9, \quad \text{profitto } 16.$$

- ▶ È la soluzione di *GR* del capitolo «Euristiche costruttive».
- ▶ Arrotondare invece $\tilde{x}_2 = \frac{1}{4}$ per eccesso darebbe $(1, 1, 1, 0)$, di peso $12 > 9$: inammissibile.

Esempio: arrotondamento sulle istanze di riferimento (2/3)

$P \parallel C_{\max}$.

- ▶ Nella soluzione LP dell'istanza di riferimento, $\left[\frac{3}{5}\right] = 1$ e $\left[\frac{2}{5}\right] = 0$ mandano il lavoro 2 sulla macchina 1.
- ▶ Ma $\left[\frac{1}{2}\right] = 1$ assegna il lavoro 3 a *entrambe* le macchine 2 e 3:

$$\bar{x}_{31} + \bar{x}_{32} + \bar{x}_{33} = 0 + 1 + 1 = 2 \neq 1.$$

- ▶ Il vincolo di assegnamento del lavoro 3 è violato:
l'arrotondamento fallisce.

Esempio: arrotondamento sulle istanze di riferimento (3/3)

BPP.

- ▶ Arrotondando la soluzione LP dell'istanza di riferimento otteniamo $\bar{\mathbf{y}} = (0, 0, 1, 1, 0, 1)$ e mettiamo l'oggetto 3 nel contenitore 3.
- ▶ Due vincoli di capacità sono violati:
 - ▶ il contenitore 1 contiene l'oggetto 5 ma $\bar{y}_1 = 0$ ($-2 + 7 \cdot 0 \geq 0$ è falso),
 - ▶ il contenitore 3 contiene gli oggetti 3 e 6 ($-(5 + 3) + 7 \cdot 1 \geq 0$ è falso).
- ▶ L'arrotondamento fallisce.

Tre esiti

- ▶ I tre esiti riassumono il metodo.
- ▶ Sul KP, con un solo vincolo di packing, l'arrotondamento è sicuro (l'osservazione sull'arrotondamento per difetto del capitolo «[Euristiche MIP](#)» del modulo *Programmazione lineare intera*).
- ▶ Sui problemi con vincoli di assegnamento fallisce, perché il LP distribuisce un lavoro o un oggetto su più macchine o contenitori, e l'arrotondamento non può decidere dove vada.

Arrotondamento:

- ▶ arrotonda $\tilde{x}$ al punto intero più vicino e completa le variabili continue
- ▶ sicuro con un solo vincolo di packing: sul KP dà la soluzione di GR , di profitto 16

Vincoli di assegnamento:

- ▶ gli arrotondamenti indipendenti non sommano quasi mai a 1
- ▶ su $P \parallel C_{\max}$ e sul BPP l'arrotondamento fallisce

Diving

Introduzione: Diving

- ▶ Il diving non arrotonda tutte le variabili insieme: le fissa una alla volta e risolve di nuovo il LP
- ▶ Eseguiamo il fractional diving sul KP e su $P \parallel C_{\max}$, e il guided diving su $P \parallel C_{\max}$
- ▶ Vediamo perché un diving può prendere una decisione sbagliata, e perché fallisce sul BPP

Diving

- ▶ Il *diving* fissa una variabile frazionaria alla volta e risolve di nuovo il LP:
 - ▶ il fractional diving fissa la variabile meno frazionaria al suo intero più vicino,
 - ▶ il guided diving fissa la variabile più vicina alla soluzione incumbent al valore dell'incumbent;fallisce quando il LP diventa inammissibile.
- ▶ I pareggi si risolvono a favore della variabile che viene prima nell'ordine delle variabili ($x_{11}, x_{12}, \dots$ per i modelli di assegnamento, poi $C_{\max}$ o $y_1, y_2, \dots$).

Esempio: fractional diving sul KP (1/3)

Passo 1. La soluzione LP è $\tilde{x} = (1, \frac{1}{4}, 1, 0)$, di valore 17.75.
L'unica variabile frazionaria è x_2 , arrotondata a $\lfloor \frac{1}{4} \rfloor = 0$: fissare $x_2 := 0$.

Passo 2. Senza l'oggetto 2, il LP prende gli oggetti 1 e 3 (peso 8) e la frazione $\frac{1}{3}$ dell'oggetto 4:

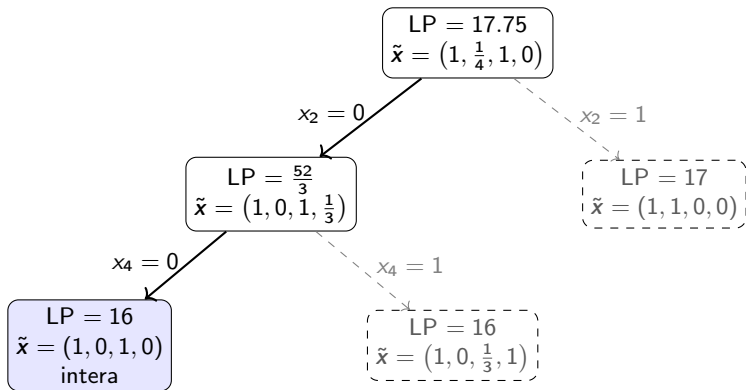
$$\tilde{x} = (1, 0, 1, \frac{1}{3}), \quad \text{di valore } 10 + 6 + \frac{4}{3} = \frac{52}{3} \approx 17.33.$$

Fissare $x_4 := \lfloor \frac{1}{3} \rfloor = 0$.

Passo 3. La soluzione LP $\tilde{x} = (1, 0, 1, 0)$, di valore 16, è intera: il diving restituisce $\{1, 3\}$, di profitto 16.

Esempio: fractional diving sul KP (2/3)

Il diving è il cammino disegnato a linee continue nell'albero di branch-and-bound qui sotto; i nodi tratteggiati sono i rami che non visita mai, con il valore dei loro rilassamenti lineari.



Esempio: fractional diving sul KP (3/3)

- ▶ Il ramo $x_2 = 1$, abbandonato al primo passo, contiene la soluzione ottima $\{1, 2\}$ di profitto 17.
- ▶ $\{1, 2\}$ è una soluzione ottima del rilassamento lineare di quel ramo.
- ▶ La prima decisione del diving, $\left\lceil \frac{1}{4} \right\rceil = 0$, era quella sbagliata.

Sul KP il diving richiede tre LP.

Su $P \parallel C_{\max}$ il LP ha molte soluzioni ottime:
seguiamo il diving passo per passo.

Esempio: fractional diving su $P \parallel C_{\max}$ (1/3)

- ▶ Il diving parte dalla soluzione LP dell'istanza di riferimento.
- ▶ La tabella riporta, per ogni passo, il valore del LP, le sue variabili frazionarie e il fissaggio.
- ▶ A parità di frazionarietà $|\tilde{x}_j - [\tilde{x}_j]|$ fra due variabili decide l'ordine $x_{11}, x_{12}, x_{13}, x_{21}, \dots$

Esempio: fractional diving su $P \parallel C_{\max}$ (2/3)

passo	valore LP	variabili frazionarie	fissaggio
1	9	$x_{21} = \frac{3}{5}, x_{23} = \frac{2}{5}, x_{32} = \frac{1}{2}, x_{33} = \frac{1}{2}$	$x_{21} := 1$
2	9	$x_{11} = \frac{1}{5}, x_{13} = \frac{4}{5}, x_{32} = \frac{3}{4}, x_{33} = \frac{1}{4}$	$x_{11} := 0$
3	9	$x_{32} = \frac{3}{4}, x_{33} = \frac{1}{4}, x_{41} = \frac{1}{4}, x_{43} = \frac{3}{4}$	$x_{32} := 1$
4	9	$x_{41} = \frac{1}{4}, x_{43} = \frac{3}{4}, x_{52} = \frac{2}{3}, x_{53} = \frac{1}{3}$	$x_{41} := 0$
5	9	$x_{61} = \frac{1}{3}, x_{62} = \frac{2}{3}$	$x_{61} := 0$
6	9	$x_{12} = \frac{3}{5}, x_{13} = \frac{2}{5}, x_{71} = \frac{1}{3}, x_{72} = \frac{2}{3}$	$x_{71} := 0$

Esempio: fractional diving su $P \parallel C_{\max}$ (3/3)

passo	valore LP	variabili frazionarie	fissaggio
7	9.5	$x_{42} = \frac{1}{8}, x_{43} = \frac{7}{8}$	$x_{42} := 0$
8	9.5	$x_{62} = \frac{1}{6}, x_{63} = \frac{5}{6}$	$x_{62} := 0$
9	9.5	$x_{12} = \frac{1}{2}, x_{13} = \frac{1}{2}$	$x_{12} := 1$
10	9.5	$x_{72} = \frac{1}{6}, x_{73} = \frac{5}{6}$	$x_{72} := 0$
11	10	nessuna	—

- L'ultima soluzione LP è intera: $J_1 = \{2, 5\}$, $J_2 = \{1, 3\}$, $J_3 = \{4, 6, 7\}$, con carichi 8, 9, 10 e makespan 10.
- Il diving migliora l'arrotondamento, che era fallito, e LPT (11), ma non raggiunge l'ottimo 9.

Un valore LP piatto

- ▶ Per sei passi il valore LP resta 9: il LP di $P \parallel C_{\max}$ ha un insieme enorme di soluzioni ottime, e fissare una variabile sposta solo il LP su un'altra di esse.
- ▶ Il valore LP non dice nulla sulla qualità delle decisioni finché il carico di una macchina non è forzato sopra 9, al passo 7, e allora è troppo tardi per annullare le decisioni che lo hanno causato.
- ▶ Gli obiettivi min-max sono piatti per il LP quanto lo sono per la ricerca locale.

Esempio: guided diving su $P \parallel C_{\max}$ (1/3)

- La soluzione incumbent è la soluzione di LPT , $J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$, di makespan 11, ossia

$$\bar{x}_{11} = \bar{x}_{51} = \bar{x}_{71} = \bar{x}_{22} = \bar{x}_{62} = \bar{x}_{33} = \bar{x}_{43} = 1$$

e tutte le altre $\bar{x}_{jh} = 0$.

- Il guided diving fissa, a ogni passo, la variabile frazionaria più vicina alla soluzione incumbent al suo valore nell'incumbent.

Esempio: guided diving su $P \parallel C_{\max}$ (2/3)

Passo 1. Nella soluzione LP dell'istanza di riferimento le variabili frazionarie e le loro distanze dall'incumbent sono

$$\begin{aligned} |\tilde{x}_{21} - \bar{x}_{21}| &= \frac{3}{5}, & |\tilde{x}_{23} - \bar{x}_{23}| &= \frac{2}{5}, \\ |\tilde{x}_{32} - \bar{x}_{32}| &= \frac{1}{2}, & |\tilde{x}_{33} - \bar{x}_{33}| &= \frac{1}{2}. \end{aligned}$$

La più vicina è x_{23} : fissare $x_{23} := \bar{x}_{23} = 0$.

Passo 2. La nuova soluzione LP, di valore 9, ha le variabili frazionarie $x_{12} = \frac{2}{5}$, $x_{13} = \frac{3}{5}$, $x_{31} = \frac{1}{2}$, $x_{32} = \frac{1}{2}$, a distanze $\frac{2}{5}$, $\frac{3}{5}$, $\frac{1}{2}$, $\frac{1}{2}$ dall'incumbent. Fissare $x_{12} := \bar{x}_{12} = 0$.

Esempio: guided diving su $P \parallel C_{\max}$ (3/3)

Passo 3. La soluzione LP è intera, di valore 9:

$$J_1 = \{5, 6, 7\}, \quad J_2 = \{2, 4\}, \quad J_3 = \{1, 3\},$$

tutti i carichi uguali a 9.

Il guided diving trova una soluzione **ottima** dopo due LP.

Che cosa decide l'incumbent

- ▶ Guidato da una soluzione di makespan 11, il diving termina in una soluzione di makespan 9.
- ▶ L'incumbent decide solo i *valori* delle due variabili fissate, entrambi uguali a 0, mentre il LP ricostruisce il resto dell'assegnamento.

Diving sul BPP

- ▶ Sull'istanza di riferimento del BPP, il fractional diving esegue 8 fissaggi, tutti con valore LP 3.
- ▶ Il nono, $x_{12} := 1$, mette l'oggetto 1 nel contenitore 2, che contiene già l'oggetto 2 per un fissaggio precedente, e $4 + 4 > 7$ rende il LP inammissibile: **il diving fallisce**.
- ▶ I fissaggi $x_{jb} = 1$ assegnano gli oggetti ai contenitori uno alla volta, e il LP del BPP è così debole (3 a ogni passo) da non dare alcun segnale prima del vicolo cieco.

Fractional diving:

- ▶ sul KP restituisce $\{1, 3\}$, di profitto 16: il primo fissaggio $x_2 := 0$ era sbagliato
- ▶ su $P \parallel C_{\max}$ arriva a makespan 10: il valore LP resta piatto a 9 per sei passi
- ▶ sul BPP fallisce dopo 8 fissaggi

Guided diving:

- ▶ su $P \parallel C_{\max}$, guidato da LPT , trova l'ottimo 9 dopo due LP

La feasibility pump

Introduzione: La feasibility pump

- ▶ La feasibility pump cerca una soluzione ammissibile quando l'arrotondamento e il diving falliscono
- ▶ La eseguiamo su $P \parallel C_{\max}$ e sul BPP
- ▶ Vediamo che trova bene soluzioni ammissibili, ma non si cura della loro qualità

La feasibility pump

- ▶ La *feasibility pump* alterna:
 - ▶ la proiezione di un punto intero sulla regione ammissibile del LP (un LP che minimizza Δ),
 - ▶ l'arrotondamento della proiezione,con un flip quando l'arrotondamento restituisce lo stesso punto intero e un riavvio casuale quando la pump cicla, finché la distanza è 0.
- ▶ I numeri casuali sono quelli del generatore *minimal standard* di Park e Miller con i semi 1, 2, ..., come nel capitolo «Metaeuristiche».

Esempio: la feasibility pump su $P \parallel C_{\max} (1/2)$

Iterazione 0. La soluzione LP dell'istanza di riferimento è arrotondata come nell'esempio dell'arrotondamento: il lavoro 3 è assegnato alle macchine 2 e 3, e $\bar{x}^0$ è inammissibile.

Iterazione 1. La proiezione minimizza $\Delta(x, \bar{x}^0)$, che è lineare perché tutte le variabili intere sono binarie.

- ▶ Il solver restituisce il punto di $\mathcal{L}$ che mantiene tutti gli assegnamenti di $\bar{x}^0$ e toglie il lavoro 3 dalla macchina 2, ossia porta x_{32} da 1 a 0: $\Delta = 1$.
- ▶ Togliere il lavoro 3 dalla macchina 3 sarebbe altrettanto vicino.

Esempio: la feasibility pump su $P \parallel C_{\max}$ (2/2)

Iterazione 1 (continua). Il risultato è intero:

$$J_1 = \{2, 6, 7\}, \quad J_2 = \{4, 5\}, \quad J_3 = \{1, 3\},$$

con carichi 11, 7, 9, e il suo arrotondamento è il nuovo punto intero $\bar{x}^1$.

Iterazione 2. $\bar{x}^1 \in \mathcal{L}$, la distanza è 0, e la pump restituisce una soluzione di makespan 11, dopo due proiezioni.

Ammissibilità, non qualità

- ▶ La soluzione ha lo stesso makespan di LPT .
- ▶ La proiezione si occupa dei vincoli di assegnamento e ignora i carichi, poiché $C_{\max}$ è continua e non entra in Δ .
- ▶ La pump risponde alla domanda «esiste una soluzione ammissibile?», che per $P \parallel C_{\max}$ non era una domanda difficile.

Esempio: la feasibility pump sul BPP (1/3)

- ▶ Sull'istanza di riferimento del BPP l'arrotondamento e il diving falliscono.
- ▶ La pump è eseguita con i semi $1, 2, \dots, 10$ dei numeri casuali.
- ▶ La tabella riporta:
 - ▶ il numero di proiezioni, di flip e di riavvii,
 - ▶ il numero di contenitori aperti dalla soluzione trovata ($\sum_b \bar{y}_b$, il valore della soluzione),
 - ▶ il numero di contenitori non vuoti.

Esempio: la feasibility pump sul BPP (2/3)

seme	1	2	3	4	5	6	7	8	9	10
proiezioni	15	12	9	9	6	6	19	5	10	20
flip	8	5	3	4	3	3	8	2	5	8
riavvii	4	3	3	2	1	1	5	1	2	4
contenitori aperti	4	4	4	5	5	4	3	4	4	5
contenitori non vuoti	4	4	4	3	5	4	3	4	4	4

Esempio: la feasibility pump sul BPP (3/3)

- ▶ La pump trova una soluzione ammissibile con ogni seme, ma sempre dopo qualche riavvio: senza di essi ciclerebbe.
- ▶ Solo con il seme 7 la soluzione è ottima: $\{3, 5\}$, $\{2, 4\}$, $\{1, 6\}$, di valore 3.
- ▶ Con il seme 4 la pump apre 5 contenitori, due dei quali restano vuoti: il riempimento $\{3, 5\}$, $\{1, 6\}$, $\{2, 4\}$ è ottimo, ma la soluzione ha valore 5, perché nulla nella funzione distanza chiede di chiudere i contenitori vuoti.

Una prima soluzione ammissibile

La pump è un metodo per trovare una *prima* soluzione ammissibile, e lo fa bene.

La qualità della soluzione è compito dei metodi di miglioramento, che partono da una soluzione incumbent.

La feasibility pump:

- ▶ proiezione sulla regione ammissibile del LP, arrotondamento, flip e riavvii casuali
- ▶ dopo il primo LP ignora l'obiettivo

Sulle istanze di riferimento:

- ▶ $P \parallel C_{\max}$: una soluzione di makespan 11 dopo due proiezioni, come *LPT*
- ▶ BPP: una soluzione ammissibile con ogni seme, ottima solo con il seme 7; possono restare aperti contenitori vuoti

RINS, local branching e proximity search

Introduzione: RINS, local branching e proximity search

- ▶ Sono metodi di miglioramento: partono da una soluzione incumbent e cercano una soluzione migliore con un sotto-MIP
- ▶ Eseguiamo ciascuno di essi sul KP e su $P \parallel C_{\max}$
- ▶ Vediamo il legame fra il local branching e gli intorni della ricerca locale, e come la proximity search può dimostrare l'ottimalità

Tre metodi con un sotto-MIP

- ▶ I tre metodi partono da una soluzione incumbent $\bar{\mathbf{x}}$ e risolvono un sotto-MIP con il vincolo di cutoff

$$f(\mathbf{x}) \leq f(\bar{\mathbf{x}}) - \theta,$$

che qui ha $\theta = 1$ poiché i valori obiettivo sono interi.

- ▶ RINS fissa le variabili intere su cui la soluzione LP e l'incumbent concordano.
- ▶ Il local branching aggiunge $\Delta(\mathbf{x}, \bar{\mathbf{x}}) \leq k$ e aumenta k quando non trova nulla.
- ▶ La proximity search minimizza $\Delta(\mathbf{x}, \bar{\mathbf{x}})$.

Cominciamo con RINS:
quali variabili fissa,
e quanto è grande il sotto-MIP che resta?

Esempio: RINS sul KP (1/2)

- ▶ La soluzione incumbent è la soluzione $\bar{x} = (1, 0, 1, 0)$ di GR , di profitto 16.
- ▶ La soluzione LP è $\tilde{x} = (1, \frac{1}{4}, 1, 0)$.
- ▶ Concordano su $x_1 = 1$, $x_3 = 1$, $x_4 = 0$, che vengono fissate.
- ▶ Il sotto-MIP, con il vincolo di capacità e il vincolo di cutoff, è

$$\begin{array}{llllll} \max & 16 & + & 7x_2 & & \\ \text{soggetto a} & 8 & + & 4x_2 & \leq & 9 \\ & 16 & + & 7x_2 & \geq & 17 \\ & & & x_2 & \in & \{0, 1\} \end{array}$$

Esempio: RINS sul KP (2/2)

- ▶ Il sotto-MIP è inammissibile: la capacità richiede $x_2 = 0$, il cutoff $x_2 = 1$.
- ▶ RINS restituisce l'incumbent.
- ▶ La soluzione ottima $\{1, 2\}$ ha $x_3 = 0$, ma x_3 è fissata a 1 perché il LP e l'incumbent concordano su di essa.

Esempio: RINS su $P \parallel C_{\max}$ (1/3)

- ▶ La soluzione incumbent è la soluzione di LPT ($J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$, makespan 11), e $\tilde{x}$ è la soluzione LP dell'istanza di riferimento.
- ▶ La tabella confronta, per ogni lavoro, la macchina dell'incumbent con la soluzione LP (lavori 1–4):

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
macchina in $\tilde{x}$	1	2	3	3
macchine in $\tilde{x}$	3	$1 (\frac{3}{5}), 3 (\frac{2}{5})$	$2 (\frac{1}{2}), 3 (\frac{1}{2})$	2
variabili fissate	$x_{12} = 0$	—	$x_{31} = 0$	$x_{41} = 0$

Esempio: RINS su $P \parallel C_{\max}$ (2/3)

Il confronto per i lavori 5–7:

	$j = 5$	$j = 6$	$j = 7$
macchina in $\bar{x}$	1	2	1
macchine in $\tilde{x}$	2	1	1
variabili fissate	$x_{53} = 0$	$x_{63} = 0$	$x_{71} = 1, x_{72} = x_{73} = 0$

- ▶ Solo il lavoro 7 è sulla stessa macchina in entrambe le soluzioni.
- ▶ Per gli altri lavori, le due soluzioni concordano solo su alcune macchine che il lavoro *non* usa.
- ▶ Otto delle 21 variabili binarie sono fissate.

Esempio: RINS su $P \parallel C_{\max}$ (3/3)

Il sotto-MIP con il cutoff $C_{\max} \leq 10$ è risolto da Gurobi, che restituisce

$$J_1 = \{5, 6, 7\}, \quad J_2 = \{2, 3\}, \quad J_3 = \{1, 4\},$$

con tutti i carichi uguali a 9.

RINS trova una soluzione ottima.

La dimensione dell'intorno di RINS

- ▶ Sul KP la soluzione incumbent e la soluzione LP sono troppo simili, e l'intorno di RINS ha una sola variabile libera.
- ▶ Su $P \parallel C_{\max}$ sono molto diverse, e l'intorno è abbastanza grande da contenere l'ottimo.

Local branching e ricerca locale

- ▶ Con $k = 1$ l'intorno del local branching è l'intorno flip del KP del capitolo «Algoritmi euristici: definizioni».
- ▶ Con $k = 2$ contiene anche l'intorno swap.
- ▶ Per le variabili di assegnamento del BPP e di $P \parallel C_{\max}$ la distanza ha un significato semplice.

Distanza fra due assegnamenti

Osservazione 1

Siano $\mathbf{x}$ e $\bar{\mathbf{x}}$ due assegnamenti ammissibili di $P \parallel C_{\max}$. Allora $\Delta(\mathbf{x}, \bar{\mathbf{x}}) = 2r$, dove r è il numero di lavori assegnati a macchine diverse in $\mathbf{x}$ e in $\bar{\mathbf{x}}$.

Dimostrazione

- ▶ Un lavoro sulla stessa macchina in $\mathbf{x}$ e $\bar{\mathbf{x}}$ contribuisce 0 a Δ .
- ▶ Un lavoro sulla macchina h in $\bar{\mathbf{x}}$ e sulla macchina $h' \neq h$ in $\mathbf{x}$ contribuisce

$$1 - x_{jh} = 1 \quad \text{e} \quad x_{jh'} = 1,$$

e 0 per le altre macchine.



Intorni move e swap

- ▶ Quindi, per $P \parallel C_{\max}$:
 - ▶ l'intorno $\Delta(\mathbf{x}, \bar{\mathbf{x}}) \leq 2$ è l'intorno **move** del capitolo «Ricerca locale» (un lavoro cambia macchina),
 - ▶ $\Delta(\mathbf{x}, \bar{\mathbf{x}}) \leq 4$ contiene l'intorno **swap** (due lavori cambiano macchina).
- ▶ Lo stesso vale per le variabili x_{jb} del BPP.

Esempio: local branching sul KP (1/2)

Partiamo da $\bar{\mathbf{x}}^1 = (1, 0, 1, 0)$, di profitto 16, con $k = 1$, $k_{\max} = 3$, $k_{\text{step}} = 1$ e $\theta = 1$.

$k = 1$. I vicini a distanza 1 sono i flip di $\bar{\mathbf{x}}^1$; con il cutoff $\sum_j p_j x_j \geq 17$ nessuno è ammissibile. Aggiungere $\Delta(\mathbf{x}, \bar{\mathbf{x}}^1) \geq 2$ e porre $k = 2$.

$k = 2$. Il sotto-MIP con $2 \leq \Delta(\mathbf{x}, \bar{\mathbf{x}}^1) \leq 2$ trova lo swap $(1, 1, 0, 0)$, di profitto 17: la nuova soluzione incumbent $\bar{\mathbf{x}}^2$. Aggiungere $\Delta(\mathbf{x}, \bar{\mathbf{x}}^1) \geq 3$.

Esempio: local branching sul KP (2/2)

$k = 2$, **attorno a $\bar{x}^2$** . Nessuna soluzione di profitto almeno 18: aggiungere $\Delta(\mathbf{x}, \bar{\mathbf{x}}^2) \geq 3$ e porre $k = 3$.

$k = 3$. Nemmeno ora c'è una soluzione di profitto almeno 18; k diventa $4 > k_{\max}$ e il metodo termina con $\{1, 2\}$, di profitto 17: l'ottimo.

Esempio: local branching su $P \parallel C_{\max}$ (1/3)

Partiamo dalla soluzione $\bar{x}^1$ di LPT , di makespan 11, con $k = 2$, $k_{\max} = 6$, $k_{step} = 2$ e $\theta = 1$, in modo che il cutoff sia $C_{\max} \leq 10$, poi 9, e così via.

$k = 2$. Per l'osservazione sulla distanza fra due assegnamenti, l'intorno è l'intorno move, e LPT è un ottimo locale per esso (capitolo «Ricerca locale»): il sotto-MIP è inammissibile. Aggiungere $\Delta(\mathbf{x}, \bar{x}^1) \geq 3$ e porre $k = 4$.

Esempio: local branching su $P \parallel C_{\max}$ (2/3)

$k = 4$. La soluzione migliore con due lavori riassegnati ha makespan 10; il solver restituisce lo swap dei lavori 1 e 4, $\bar{x}^2$ nella figura seguente. Aggiungere $\Delta(\mathbf{x}, \bar{x}^1) \geq 5$.

$k = 4$, **attorno a $\bar{x}^2$** . Il solver restituisce lo swap dei lavori 4 e 6, di makespan 9: $\bar{x}^3$. Aggiungere $\Delta(\mathbf{x}, \bar{x}^2) \geq 5$.

$k = 4$ e $k = 6$, **attorno a $\bar{x}^3$** . Nessuna soluzione di makespan 8: k diventa $8 > k_{\max}$, e il metodo termina con makespan 9, l'ottimo.

Esempio: local branching su $P \parallel C_{\max}$ (3/3)

$\bar{x}^1$ (LPT), $C_{\max} = 11$

M_1	1	5	7
M_2	2	6	
M_3	3	4	

$\bar{x}^2$, $C_{\max} = 10$

M_1	4	5	7
M_2	2	6	
M_3	1	3	

$\bar{x}^3$, $C_{\max} = 9$

M_1	5	6	7
M_2	2	4	
M_3	1	3	

La linea tratteggiata è il lower bound 9; i lavori scambiati per ottenere la soluzione successiva sono colorati.

Il local branching come variable neighbourhood search

- ▶ Nel linguaggio del capitolo «Ricerca locale», il local branching esegue una ricerca locale con un intorno la cui dimensione è controllata da k , esplorato in modo esatto dal solver.
- ▶ Allarga l'intorno quando si blocca.
- ▶ È esattamente l'idea della variable neighbourhood search, con il solver che esegue l'esplorazione.

Proximity search e ottimalità

- ▶ Un sotto-MIP di proximity inammissibile con $\theta = 1$ dimostra che la soluzione incumbent è ottima, quando l'obiettivo assume valori interi.
- ▶ È la proposizione del capitolo «**Euristiche MIP**» del modulo *Programmazione lineare intera*.

Esempio: proximity search sul KP

- ▶ Da $\bar{x} = (1, 0, 1, 0)$, di profitto 16: la soluzione più vicina di profitto almeno 17 è $(1, 1, 0, 0)$, a distanza 2 (esce l'oggetto 3, entra l'oggetto 2), di profitto 17.
- ▶ Attorno a essa nessuna soluzione ha profitto almeno 18.
- ▶ Per la condizione di ottimalità della proximity search (capitolo «**Euristiche MIP**» del modulo *Programmazione lineare intera*), $\{1, 2\}$ è **ottima**.

Esempio: proximity search su $P \parallel C_{\max}$ (1/2)

Dalla soluzione di LPT , di makespan 11:

iterazione	cutoff	min Δ	soluzione restituita	$C_{\max}$
1	$C_{\max} \leq 10$	4	$J_1 = \{5, 6, 7\},$ $J_2 = \{1, 2\},$ $J_3 = \{3, 4\}$	10
2	$C_{\max} \leq 9$	4	$J_1 = \{5, 6, 7\},$ $J_2 = \{1, 4\},$ $J_3 = \{2, 3\}$	9
3	$C_{\max} \leq 8$	—	inammissibile	—

Esempio: proximity search su $P || C_{\max}$ (2/2)

- ▶ Per l'osservazione sulla distanza fra due assegnamenti, distanza 4 significa due lavori riassegnati: i lavori 1 e 6 alla prima iterazione, i lavori 2 e 4 alla seconda.
- ▶ Nessuna soluzione a distanza 2 migliora il makespan, come per il local branching con $k = 2$.
- ▶ Alla terza iterazione il sotto-MIP è inammissibile, e la condizione di ottimalità della proximity search (capitolo «Euristiche MIP» del modulo *Programmazione lineare intera*) dimostra che il makespan 9 è ottimo.

La dimostrazione di ottimalità viene dal solver

Il solver dimostra l'inammissibilità dell'ultimo sotto-MIP
con i propri bound duali:

la proximity search, usata con $\theta = 1$,
è un metodo esatto che migliora
la soluzione incumbent strada facendo.

Riepilogo (1/2)

RINS:

- ▶ sul KP l'intorno ha una sola variabile libera e il sotto-MIP è inammissibile
- ▶ su $P \parallel C_{\max}$ otto delle 21 variabili binarie sono fissate, e RINS trova l'ottimo 9

Local branching:

- ▶ $\Delta \leq 2$ è l'intorno move, $\Delta \leq 4$ contiene l'intorno swap
- ▶ trova l'ottimo sul KP (17) e su $P \parallel C_{\max}$ (9)

Riepilogo (2/2)

Proximity search:

- ▶ a ogni iterazione trova la soluzione migliorante più vicina
- ▶ sul KP e su $P \parallel C_{\max}$ l'ultimo sotto-MIP inammissibile dimostra l'ottimalità

Il vincolo di cutoff:

- ▶ $f(\mathbf{x}) \leq f(\bar{\mathbf{x}}) - \theta$ con $\theta = 1$, poiché i valori obiettivo sono interi

Relax-and-fix e fix-and-optimize sul lot sizing

Introduzione: Relax-and-fix e fix-and-optimize sul lot sizing

- ▶ Relax-and-fix e fix-and-optimize lavorano su blocchi di variabili intere
- ▶ Il lot sizing, con i suoi periodi, è la loro applicazione naturale
- ▶ Eseguiamo relax-and-fix su due istanze, e fix-and-optimize su quella capacitata

Blocchi di periodi

- ▶ Relax-and-fix e fix-and-optimize decompongono il MIP in blocchi di variabili intere.
- ▶ Il lot sizing è un'applicazione naturale, perché le sue variabili sono indicizzate dai periodi $t \in T = \{1, 2, \dots, m\}$ di un orizzonte di pianificazione, e le decisioni di un periodo interagiscono soprattutto con quelle dei periodi vicini.
- ▶ I blocchi sono le variabili di setup di periodi consecutivi.

Il modello del lot sizing: variabili

- Il modello è il modello ILP_{CLSP} del capitolo «[Formulazioni dei problemi di lot sizing](#)», con il valore forte di big-M

$$M_t = \min\{cap_t, D_{t,m}\}, \quad D_{t,m} = \sum_{k=t}^m d_k;$$

per un'istanza senza capacità, $cap_t = D_{t,m}$.

- Le sue variabili sono la produzione x_t , la scorta s_t alla fine del periodo t e il setup y_t , per ogni periodo $t \in T$.

Il modello del lot sizing

Usando queste variabili, un modello PLI per il **problema di lot sizing capacitato (CLS)** è il seguente:

$$\begin{aligned} \min \quad & \sum_{t \in T} (sc_t y_t + vc_t x_t + hc_t s_t) \\ \text{soggetto a} \quad & s_{t-1} + x_t = d_t + s_t, \quad \forall t \in T, \\ & x_t \leq M_t y_t, \quad \forall t \in T, \\ & x_t \leq cap_t, \quad \forall t \in T, \\ & s_0 = s_m = 0, \\ & x_t \in \mathbb{Z}_{\geq 0}, \quad \forall t \in T, \\ & s_t \in \mathbb{Z}_{\geq 0}, \quad \forall t \in T, \\ & y_t \in \{0, 1\}, \quad \forall t \in T. \end{aligned}$$

Questo modello PLI si chiama modello ILP_{CLSP}.

Le decisioni difficili

- ▶ Fissati i setup y , resta un LP con dati interi e una matrice di rete, che ha una soluzione ottima intera.
- ▶ Le decisioni difficili sono le m variabili binarie: come nei codici, le euristiche trattano x_t e s_t come variabili continue, con $x_t \geq 0$ e $s_t \geq 0$ al posto di $x_t \in \mathbb{Z}_{\geq 0}$ e $s_t \in \mathbb{Z}_{\geq 0}$.
- ▶ Il rilassamento lineare è debole, perché una frazione $y_t = x_t/M_t$ di setup basta per produrre x_t .

Esempio: relax-and-fix su un'istanza di lot sizing senza capacità

- ▶ L'istanza del modulo *Modellazione MIP*, capitolo «[Euristiche costruttive](#)», ha $m = 5$ periodi, $d = (20, 10, 30, 40, 10)$, $sc_t = 50$, $vc_t = 0$ e $hc_t = 1$ per ogni $t \in T$, e nessuna capacità.
- ▶ Il suo rilassamento lineare ha valore $\frac{36985}{396} \approx 93.40$, l euristica costruttiva del costo unitario minimo di quel capitolo dà 200, e il valore ottimo è 170.
- ▶ Relax-and-fix con un periodo per blocco, $B_t = \{y_t\}$, trova una soluzione di valore 170: **ottima**.

Esempio: relax-and-fix su un'istanza capacitata (1/3)

- ▶ Consideriamo $m = 6$ periodi, $d = (50, 30, 10, 10, 40, 10)$, $sc_t = 40$, $vc_t = 0$, $hc_t = 1$ e $cap_t = 80$ per ogni $t \in T$.
- ▶ Il rilassamento lineare ha valore $\frac{2150}{21} \approx 102.38$.
- ▶ Il valore ottimo è 160, con setup nei periodi 1, 2, 5.
- ▶ Eseguiamo relax-and-fix con $B_t = \{y_t\}$.

Esempio: relax-and-fix su un'istanza capacitata (2/3)

passo r	valore del MIP del passo r	y della soluzione	fissaggio
1	$\frac{2465}{21} \approx 117.38$	$(1, 0.375, 0.143, 0.167, 1, 0)$	$y_1 := 1$
2	$\frac{2780}{21} \approx 132.38$	$(1, 0, 0.143, 0.167, 1, 0)$	$y_2 := 0$
3	$\frac{500}{3} \approx 166.67$	$(1, 0, 1, 0.167, 1, 0)$	$y_3 := 1$
4	170	$(1, 0, 1, 0, 1, 0)$	$y_4 := 0$
5	170	$(1, 0, 1, 0, 1, 0)$	$y_5 := 1$
6	170	$(1, 0, 1, 0, 1, 0)$	$y_6 := 0$

(i valori frazionari sono arrotondati a tre decimali)

Esempio: relax-and-fix su un'istanza capacitata (3/3)

- ▶ Il risultato ha setup nei periodi 1, 3, 5: produzione 80, 20, 50, scorte 30, 10, 10 alla fine dei periodi 1, 3, 5, costo $3 \cdot 40 + 50 = 170$.
- ▶ La decisione sbagliata è presa al passo 2: con il futuro rilassato, $y_2 = 0$ dà 132.38 e $y_2 = 1$ dà 142.38, quindi relax-and-fix risparmia il setup del periodo 2.
- ▶ Ma poi la capacità impone un setup nel periodo 3, che il rilassamento aveva pagato solo per una frazione 0.143.

Esempio: fix-and-optimize sull'istanza capacitata (1/2)

Partiamo dalla soluzione di relax-and-fix dell'esempio capacitato, $\bar{y} = (1, 0, 1, 0, 1, 0)$ di costo 170, con le finestre di due periodi consecutivi $W_t = \{y_t, y_{t+1}\}$, $t = 1, 2, \dots, 5$.

finestra	incumbent dopo la finestra	
$\{y_1, y_2\}$	170	nessun miglioramento
$\{y_2, y_3\}$	160	$\bar{y} = (1, 1, 0, 0, 1, 0)$
$\{y_3, y_4\}, \{y_4, y_5\},$ $\{y_5, y_6\}$	160	nessun miglioramento
secondo passaggio, tutte le finestre	160	nessun miglioramento: stop

Esempio: fix-and-optimize sull'istanza capacitata (2/2)

- ▶ La finestra $\{y_2, y_3\}$ corregge la decisione sbagliata di relax-and-fix: il setup passa dal periodo 3 al periodo 2.
- ▶ Il piano produce 50, 50, 50 nei periodi 1, 2, 5, con scorte 20 e 10 alla fine dei periodi 2, 3 e 10 alla fine del periodo 5.
- ▶ Costo $3 \cdot 40 + 40 = 160$, l'**ottimo**.

Relax-and-fix:

- ▶ un blocco di setup per periodo, il futuro rilassato
- ▶ istanza senza capacità: 170, ottima; istanza capacitata: 170, contro l'ottimo 160
- ▶ è miope: la decisione economica $y_2 := 0$ impone un setup nel periodo 3

Fix-and-optimize:

- ▶ finestre di due periodi consecutivi, riottimizzate una alla volta
- ▶ la finestra $\{y_2, y_3\}$ corregge la decisione sbagliata e raggiunge l'ottimo 160

Le euristiche dentro il solver

Introduzione: Le euristiche dentro il solver

- ▶ Un solver MIP esegue le proprie euristiche durante il branch-and-bound
- ▶ Misuriamo il loro effetto su un'istanza media del BPP, cambiando i parametri di Gurobi
- ▶ Vediamo che, quando il bound è forte, trovare la soluzione è tutta la difficoltà

Le euristiche del solver

- ▶ Le euristiche di Gurobi, i loro parametri e il MIP start sono descritti nei capitoli «[Euristiche MIP](#)» e «[Il solver MIP: che cosa fanno](#)» del modulo *Programmazione lineare intera*.
- ▶ Una soluzione euristica calcolata fuori dal solver (con *FFD*, *LPT*, una ricerca locale) si può passare al solver come MIP start (l'attributo *Start* delle variabili).

Esempio: le euristiche di Gurobi su un'istanza media del BPP (1/6)

L'istanza ha $n = 60$ oggetti, con pesi $w_j = 20 + (s_j \bmod 41)$, dove $s_1, s_2, \dots$ è la sequenza di Park–Miller dal seme 1, e capacità $c = 100$:

58	29	22	22	55	45	38	40	34	32
43	57	20	57	53	32	40	57	50	33
29	49	37	20	27	22	55	40	48	40
52	24	21	48	31	36	38	46	30	20
31	58	40	28	54	50	26	31	27	54
26	26	22	30	33	55	27	50	47	26

Esempio: le euristiche di Gurobi su un'istanza media del BPP (2/6)

- ▶ Il peso totale è 2271, quindi $LB = \lceil 2271/100 \rceil = 23$.
- ▶ *FFD* usa 24 contenitori, e il modello ILP_{BPP} è costruito con $m = 24$ contenitori.
- ▶ Gurobi 13.0.3 è eseguito su un thread con un limite di 10 *work unit* (una misura deterministica dello sforzo di calcolo, all'incirca un secondo ciascuna), con parametri diversi.

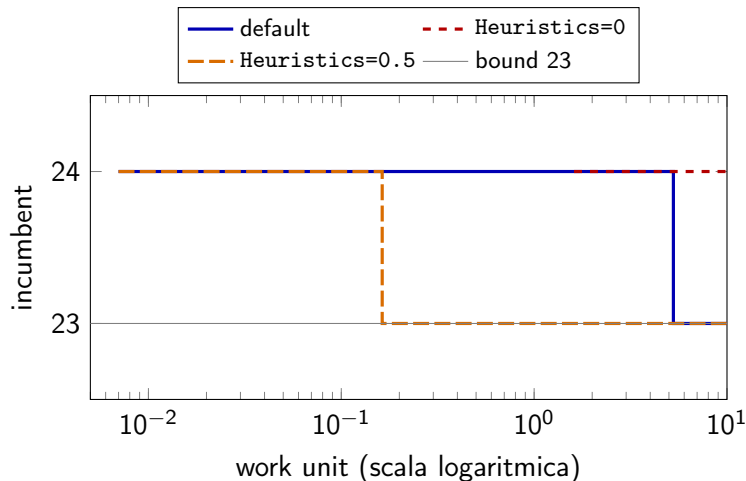
Esempio: le euristiche di Gurobi su un'istanza media del BPP (3/6)

parametri	incumbent	bound	nodì	work
default	23	23	2533	5.27
Heuristics=0	24	23	4809	10.00
Heuristics=0.5	23	23	1	0.16
MIPFocus=1	23	23	1	0.16
NoRelHeurWork=1	23	23	2533	6.48
RINS=0	23	23	2959	5.86

Esempio: le euristiche di Gurobi su un'istanza media del BPP (4/6)

parametri	incumbent trovate (work: valore)
default	0.007: 24; 5.273: 23
Heuristics=0	1.610: 24
Heuristics=0.5	0.007: 24; 0.163: 23
MIPFocus=1	0.007: 24; 0.160: 23
NoRelHeurWork=1	0.007: 24; 6.477: 23
RINS=0	0.007: 24; 5.861: 23

Esempio: le euristiche di Gurobi su un'istanza media del BPP (5/6)



Esempio: le euristiche di Gurobi su un'istanza media del BPP (6/6)

- ▶ Con i parametri di default, una soluzione con 24 contenitori è trovata subito, e quella ottima solo dopo 2533 nodi.
- ▶ Dedicando più sforzo alle euristiche ($\text{Heuristics}=0.5$ o $\text{MIPFocus}=1$) si trova 23 alla radice, circa 30 volte più velocemente.
- ▶ Senza euristiche la prima soluzione arriva dopo 1.61 work unit, e l'ottimo non è trovato entro il limite.
- ▶ Poiché il bound vale 23 già alla radice, tutta la difficoltà di questa istanza è primale: trovare la soluzione, non dimostrarne l'ottimalità.

Una lezione qualitativa

- ▶ I numeri dipendono dall'istanza, dalla versione del solver e dalla macchina; una sola istanza non dimostra nulla (capitolo «Algoritmi euristici: definizioni»).
- ▶ La lezione è qualitativa: sulle istanze il cui bound è forte, come il BPP con $LB = z(I)$, il tempo di soluzione è il tempo per trovare una buona soluzione.
- ▶ Lo decidono le euristiche del solver, o un buon MIP start.

Le euristiche dentro il solver:

- ▶ Gurobi esegue le proprie euristiche; una soluzione trovata fuori si può passare come MIP start
- ▶ sull'istanza media del BPP il bound 23 è noto già alla radice

L'effetto dei parametri:

- ▶ più sforzo nelle euristiche trova 23 alla radice, circa 30 volte più velocemente
- ▶ senza euristiche l'ottimo non è trovato entro 10 work unit
- ▶ quando il bound è forte, la difficoltà è primale

Che cosa danno le euristiche basate sulla PLI

Introduzione: Che cosa danno le euristiche basate sulla PLI

- ▶ Confrontiamo le euristiche dei capitoli precedenti con le euristiche basate sulla PLI su tutte le istanze del capitolo
- ▶ La differenza chiave è il bound duale: con un modello MIP il gap di una soluzione è noto
- ▶ Un gap uguale a 0 certifica che la soluzione è ottima

Bound primali e duali

- ▶ Il capitolo «Algoritmi euristici: definizioni» si chiudeva con un avvertimento: un'euristica dà solo un lato del gap, e senza un bound duale non si può dire nulla sulla qualità della sua soluzione.
- ▶ Le euristiche dei capitoli precedenti danno *solo* bound primali.

L'euristica sa quanto è lontana dall'ottimo

- ▶ Un'euristica basata sulla PLI lavora con un modello MIP, e il modello porta con sé il suo rilassamento lineare.
- ▶ Eseguita dentro un solver MIP, porta con sé anche il bound duale del branch-and-bound.
- ▶ Per la prima volta, l'euristica **sa quanto è lontana dall'ottimo** mentre è in esecuzione: il gap del capitolo «Algoritmi euristici: definizioni» si calcola in ogni momento, e quando vale 0 la soluzione è *dimostrata* ottima.

I gap delle due famiglie (1/2)

Euristiche dei capitoli precedenti:

istanza	soluzione	bound duale	gap
KP	GR : 16	LP: 17	6.25%
BPP	FF : 4	$\lceil 21/7 \rceil = 3$	33.3%
TSP	NN : 25	due lati: 17	47.1%
$P \parallel C_{\max}$	LPT : 11	9	22.2%
lot sizing, capacitato	RF: 170	LP: 102.38	66.0%
BPP medio	FFD : 24	23	4.3%

I gap delle due famiglie (2/2)

Euristiche basate sulla PLI:

istanza	soluzione	bound duale	gap
KP	LB, PS: 17	PS: 17	0%
BPP	pump (seme 7): 3	LP: 3	0%
TSP	arrotondamento: 18	LP: 18	0%
$P \parallel C_{\max}$	RINS, LB, PS: 9	LP: 9	0%
lot sizing, capacitato	FO: 160	solver: 160	0%
BPP medio	Gurobi: 23	23	0%

Da dove vengono i bound duali

- ▶ Sul KP l'arrotondamento intero del valore LP è $\lfloor 17.75 \rfloor = 17$, e la proximity search dimostra da sola l'ottimalità di 17, per la condizione di ottimalità della proximity search (capitolo «Euristiche MIP» del modulo *Programmazione lineare intera*).
- ▶ Sull'istanza di lot sizing capacitato il bound LP è debole, e il gap è chiuso dal branch-and-bound del solver, che dimostra che 160 è ottimo.

Soluzioni con un certificato

Un'euristica basata sulla PLI non si limita a trovare soluzioni:
dentro un solver MIP lavora con un bound duale,
quindi il gap della sua soluzione è sempre noto,
e un gap uguale a 0 certifica l'ottimalità.

Capitoli precedenti:

- ▶ solo bound primali: gap da 4.3% a 66.0% sulle istanze del capitolo

Euristiche basate sulla PLI:

- ▶ il modello dà il bound LP, il solver il bound del branch-and-bound
- ▶ su ogni istanza del capitolo il gap è 0%: le soluzioni sono dimostrate ottime

Errori tipici

Introduzione: Errori tipici

- ▶ Raccogliamo gli errori che si commettono più spesso con le euristiche basate sulla PLI
- ▶ Ciascuno di essi si è visto in uno degli esempi del capitolo
- ▶ Controllateli prima di fidarvi del risultato di un metodo

Errori tipici (1/4)

Osservazione 2: Arrotondare un problema con vincoli di uguaglianza

Gli arrotondamenti indipendenti di un vincolo di assegnamento non sommano quasi mai a 1: sul BPP e su $P \parallel C_{\max}$ l'arrotondamento fallisce.

Osservazione 3: Credere che un diving possa annullare una decisione

Un diving è un solo cammino dell'albero, senza backtracking: il primo fissaggio sbagliato (sul KP, $x_2 := 0$) non viene mai rivisto.

Osservazione 4: Usare la feasibility pump per ottenere una buona soluzione

Dopo il primo LP la pump ignora l'obiettivo: restituisce una soluzione ammissibile, eventualmente con contenitori vuoti lasciati aperti.

Osservazione 5: Fidarsi del fissaggio rigido

RINS fissa le variabili su cui il LP e l'incumbent concordano, quindi non può trovare una soluzione migliore che ne cambi una.

Osservazione 6: Dimenticare il vincolo di cutoff

Senza $f(\mathbf{x}) \leq f(\bar{\mathbf{x}}) - \theta$ il sotto-MIP restituisce di nuovo la soluzione incumbent; con esso, un sotto-MIP inammissibile dimostra un ottimo locale.

Osservazione 7: Dichiarare l'ottimalità senza le condizioni

La condizione di ottimalità della proximity search (capitolo «[Euristiche MIP](#)» del modulo Programmazione lineare intera) richiede $\theta = 1$ e valori obiettivo interi su tutte le soluzioni ammissibili.

Osservazione 8: Ignorare la miopia di relax-and-fix

Il futuro rilassato è ottimistico: una decisione che sembra economica può imporre un setup più avanti.

Osservazione 9: Generalizzare da una sola istanza o da una sola esecuzione del solver

Cammini e tempi dipendono dalla versione del solver e dalle regole di pareggio, quindi le lezioni sono qualitative.

Costruire una soluzione:

- ▶ l'arrotondamento fallisce sui vincoli di uguaglianza; un diving non rivede mai un fissaggio sbagliato
- ▶ la pump cerca l'ammissibilità, non la qualità

Migliorare e certificare:

- ▶ il fissaggio rigido può escludere le soluzioni migliori; mai dimenticare il cutoff
- ▶ l'ottimalità dalla proximity search richiede $\theta = 1$ e valori obiettivo interi
- ▶ relax-and-fix è miope; una sola istanza o una sola esecuzione non dimostra nulla

Implementazione in Python



Introduzione: Implementazione in Python

- ▶ I metodi del capitolo sono eseguiti con Gurobi, da Python
- ▶ Guardiamo brevi estratti: come si costruiscono i modelli con gurobipy, e come si esegue l'esperimento con i parametri di Gurobi
- ▶ I codici completi sono nella cartella MIP_HEUR_CODE

I codici del capitolo

- ▶ I codici dei metodi (arrotondamento, diving, feasibility pump, RINS, local branching, proximity search, relax-and-fix e fix-and-optimize) sono quelli del capitolo «**Euristiche MIP**» del modulo *Programmazione lineare intera*.
- ▶ Qui diamo i modelli dei problemi classici (`models.py`) e l'esperimento con i parametri di Gurobi (`gurobi_heuristics.py`).
- ▶ La cartella `MIP_HEUR_CODE` contiene tutte le funzioni e `instances.py`, con le istanze del capitolo e degli esercizi.
- ▶ Il comando `python3 run_tests.py` esegue ogni esempio del capitolo e ogni istanza degli esercizi, confrontando ogni valore con il valore atteso, scritto nel file.

Il modello ILP_{KP} in gurobipy

```
def build_kp(p, w, c):  
    m = gp.Model("KP", env=ENV)  
    n = len(p)  
    x = m.addVars(n, vtype=GRB.BINARY, name="x")  
    m.setObjective(gp.quicksum(p[j] * x[j]  
                               for j in range(n)), GRB.MAXIMIZE)  
    m.addConstr(gp.quicksum(w[j] * x[j]  
                             for j in range(n)) <= c,  
                name="capacity")  
  
    m.update()  
    return m
```

Il modello $ILP_{P||C_{max}}$ in gurobipy

```
def build_pcm(t, m):  
    model = gp.Model("PCmax", env=ENV)  
    n = len(t)  
    x = model.addVars(n, m, vtype=GRB.BINARY,  
                      name="x")  
    Cmax = model.addVar(name="Cmax")  
    model.setObjective(Cmax, GRB.MINIMIZE)  
    model.addConstrs((x.sum(j, "*") == 1  
                      for j in range(n)), name="assign")  
    model.addConstrs((-gp.quicksum(t[j] * x[j, h]  
                                   for j in range(n)) + Cmax >= 0  
                      for h in range(m)), name="load")  
    model.update()  
    return model
```

I parametri di Gurobi

```
s = m.copy()
s.Params.WorkLimit = work_limit
for name, value in params.items():
    s.setParam(name, value)
found = []
# called by Gurobi at every new incumbent
def callback(model, where):
    if where == GRB.Callback.MIPSOL:
        found.append(
            (model.cbGet(GRB.Callback.WORK),
             model.cbGet(GRB.Callback.MIPSOL_OBJ)))
s.optimize(callback)
```

I codici:

- ▶ i metodi sono quelli del capitolo «Euristiche MIP» del modulo *Programmazione lineare intera*
- ▶ `models.py` costruisce i modelli dei problemi classici con `gurobipy`
- ▶ `gurobi_heuristics.py` esegue l'esperimento con i parametri di Gurobi, registrando ogni nuova incumbent con una callback

Test:

- ▶ `python3 run_tests.py` confronta ogni esempio e ogni esercizio con i valori attesi

In sintesi

Introduzione: In sintesi

- ▶ Raccogliamo le idee chiave del capitolo sulle euristiche MIP per i problemi classici
- ▶ Il quadro riassuntivo raccoglie i risultati migliori sulle istanze di riferimento

Riepilogo (1/2)

- ▶ Le euristiche basate sulla PLI del capitolo «**Euristiche MIP**» del modulo *Programmazione lineare intera* si applicano ai problemi classici attraverso i loro modelli MIP.
- ▶ L'arrotondamento e il diving costruiscono una soluzione a partire dal LP; la feasibility pump cerca solo l'ammissibilità; RINS, local branching e proximity search migliorano una soluzione incumbent con un sotto-MIP.
- ▶ Relax-and-fix costruisce una soluzione lungo il tempo, fix-and-optimize la migliora finestra per finestra.
- ▶ Dentro un solver MIP il bound duale è disponibile: il gap della soluzione euristica è noto, e un gap uguale a 0 certifica l'ottimalità.

Riepilogo (2/2)

Quadro riassuntivo delle istanze di riferimento. Migliore soluzione trovata dai metodi di questo capitolo contro il bound duale (max per il KP, min per gli altri):

istanza	soluzione	bound duale	gap
KP	17	17	0%
BPP	3	3	0%
$P \parallel C_{\max}$	9	9	0%
lot sizing, capacitato	160	160	0%
BPP medio	23	23	0%