



SAPIENZA
UNIVERSITÀ DI ROMA

Programmazione lineare intera

Volume 3 — I solver MIP

Che cosa fa un solver MIP (presolve, tagli, euristiche primali, strategie dell'albero, tolleranze) e come si legge il suo log, con esempi reali.

Fabio Furini

fabiofurini.github.io

Dipartimento di Ingegneria informatica, automatica e gestionale — Sapienza Università di Roma

Versione online e materiale scaricabile:

fabiofurini.github.io/programmazione-lineare-intera/
github.com/fabiofurini/programmazione-lineare-intera

Anno accademico 2026–2027

Indice

IV	I solver MIP	2
9	I solver MIP: che cosa fanno	3
9.1	Dall'algoritmo branch-and-cut a un solver MIP	4
9.2	I componenti di un solver MIP	5
9.3	Come si legge il log di Gurobi	8
9.4	Errori tipici	14
9.5	In sintesi	15



Parte IV

I solver MIP



I solver MIP: che cosa fanno

Prerequisiti

- Da questo modulo, i capitoli «Algoritmi branch-and-bound basati su LP» e «Algoritmi branch-and-cut»: le operazioni di branching, bounding e pruning, i tagli e la soluzione incumbent.
- Da questo modulo, il capitolo «Bound primali e bound duali»: i lower bound e gli upper bound e il gap di ottimalità.
- Per imparare a usare Gurobi nella pratica, vedi il modulo «Modellazione MIP» (fabiofurini.github.io/modellazione-mip), capitolo 3 «Dal modello a Python/Gurobi».

Obiettivi

Alla fine di questo capitolo sarai in grado di:

- nominare i componenti di un solver MIP e dire su quale lato dei bound agisce ciascuno di essi;
- spiegare presolve, tagli, euristiche primali, regole di branching, tolleranze e gap;
- leggere il log di Gurobi e seguire i due bound, la soluzione incumbent e l'albero;
- spiegare perché un solver può fermarsi con una soluzione che non è ottima.

9.1 Dall'algoritmo branch-and-cut a un solver MIP

Definizione 9.1: Risolutore MIP

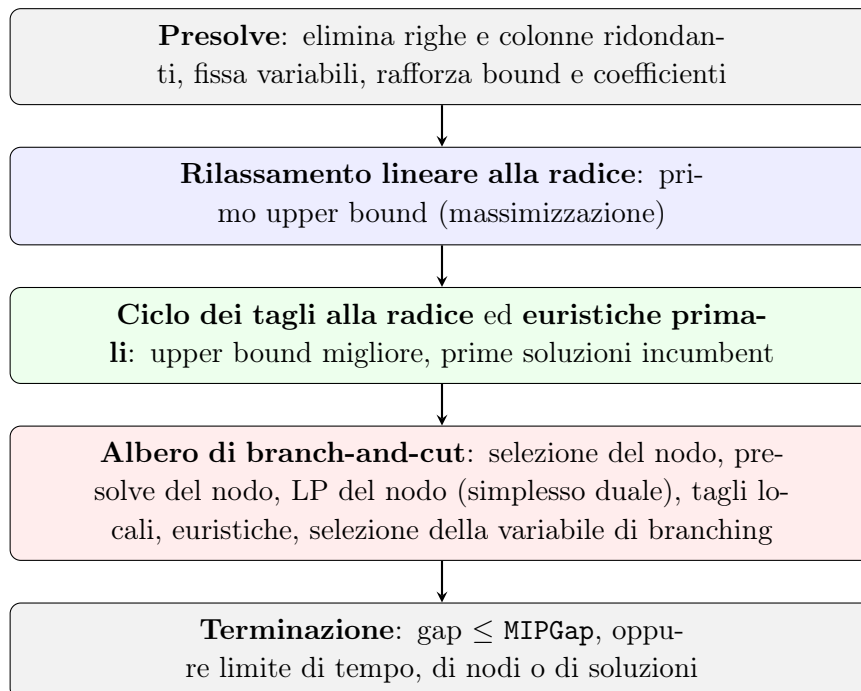
Un **solver MIP** (Gurobi, CPLEX, Xpress, HiGHS, SCIP) è un programma che calcola soluzioni ottime di problemi MIP. Il suo nucleo è l'algoritmo branch-and-cut del capitolo precedente.

- Intorno al nucleo, il solver aggiunge diversi componenti. Ciascuno di essi agisce su un lato dei bound su $z(\text{ILP})$, qui per un problema di massimizzazione:

$$\underbrace{\text{LB}}_{\text{lato primale}} \leq z(\text{ILP}) \leq \underbrace{\text{UB}}_{\text{lato duale}}$$

- I componenti sono gli stessi in tutti i solver. I dettagli (quali tagli, quali euristiche, quali regole) cambiano da solver a solver e da versione a versione, e sono documentati solo in parte.

Un solver MIP esegue il presolve, poi il nodo radice (rilassamento lineare, tagli, euristiche primali), poi l'albero di branch-and-cut, e si ferma quando il gap tra i bound scende sotto una tolleranza o quando si raggiunge un limite di tempo, di nodi o di soluzioni.



9.1.1 Dizionario tra gli appunti e il solver

Negli appunti (B&B, B&C)	Nel log di Gurobi
lower bound LB (max), soluzione incumbent $\tilde{x}$	Incumbent
upper bound UB (max)	BestBd (best bound)
gap di ottimalità	Gap
lista L dei problemi LP da elaborare	Unexpl (nodi non esplorati)
problemi LP già elaborati	Expl (nodi esplorati)
$z(LP_k)$ del nodo di branching corrente	Obj
livello del nodo nell'albero	Depth
variabili intere frazionarie in $\tilde{x}(LP_k)$	IntInf

- In un problema di minimizzazione i ruoli si scambiano: **Incumbent** è un upper bound e **BestBd** è un lower bound.

9.2 I componenti di un solver MIP

9.2.1 Presolve

Definizione 9.2: presolve

Il **presolve** trasforma un problema MIP in uno equivalente: stesso valore ottimo, e una soluzione ottima del problema originale si ricava da una soluzione ottima del nuovo problema.

- Il nuovo problema è più piccolo e, spesso, ha un rilassamento lineare più forte. Le operazioni tipiche sono tre:
 1. eliminare i **vincoli ridondanti** e **fissare le variabili** il cui valore è implicato dagli altri vincoli;
 2. **rafforzamento dei bound** (bound tightening): ricavare upper bound più piccoli e lower bound più grandi sulle variabili;
 3. **rafforzamento dei coefficienti** (coefficient tightening): modificare i coefficienti di un vincolo in modo che le soluzioni intere restino le stesse, mentre il rilassamento lineare diventa più piccolo.

Esempio 9.1: rafforzamento dei coefficienti

Consideriamo il problema con tre variabili binarie

$$\max x_1 + x_2 + x_3 \quad \text{soggetto a} \quad 6x_1 + 6x_2 + 3x_3 \leq 10, \quad x_1, x_2, x_3 \in \{0, 1\}.$$

- Il vincolo è soddisfatto dagli stessi sei punti binari del vincolo

$$7x_1 + 7x_2 + 3x_3 \leq 10.$$

Infatti $x_1 = x_2 = 1$ viola entrambi ($12 > 10$ e $14 > 10$), mentre ogni altro punto li soddisfa entrambi (i primi membri più grandi sono 9 e 10).

- Il rilassamento lineare del primo vincolo ha valore ottimo $z(\text{LP}) = \frac{13}{6}$, raggiunto per esempio in $\tilde{x} = (1, \frac{1}{6}, 1)$: è frazionaria, e $x_1 + x_2 = \frac{7}{6}$.
- Il rilassamento lineare del secondo vincolo ha valore ottimo $2 = z(\text{ILP})$: l'upper bound scende da $\frac{13}{6}$ a 2 senza cambiare le soluzioni intere né la dimensione del modello.

- Lo stesso ragionamento si può applicare in ogni nodo di branching (**presolve del nodo**, o **propagazione dei domini**): un vincolo di branching $x_j \leq 0$ può implicare che altre variabili si possano fissare.

9.2.2 Piani di taglio

- Al nodo radice il solver esegue diversi **round di tagli**. Alla fine, il log riporta quanti tagli di ciascuna famiglia sono stati aggiunti.
- Le famiglie che compaiono nei nostri log sono **Gomory** (tagli di Gomory), **MIR** (tagli di mixed-integer rounding, che generalizzano i tagli di Gomory ai problemi MIP), **StrongCG** (tagli di Chvátal–Gomory) e **Lift-and-project**.
- I tagli si generano anche nell'albero, meno spesso che alla radice (branch-and-cut).

9.2.3 Euristiche primali

- Le **euristiche primali** cercano buone soluzioni intere ammissibili: forniscono la soluzione incumbent e il lower bound LB (massimizzazione). Esempi sono l'arrotondamento della soluzione del rilassamento lineare, il **diving** e la ricerca locale intorno alla soluzione incumbent (capitolo «Euristiche MIP»).
- Nel log, una riga che inizia con H riporta una nuova soluzione incumbent trovata da un'euristica. Una riga che inizia con * ne riporta una trovata come soluzione ottima intera del problema LP di un nodo (potatura per integralità).
- La riga **Found heuristic solution** riporta una soluzione trovata prima del problema LP della radice.

9.2.4 Selezione della variabile e del nodo di branching

- Le strategie di selezione della variabile di branching del capitolo sul B&B (massima inammissibilità, strong branching, pseudocosti) sono tutte disponibili in un solver. Lo strong branching dà alberi piccoli ma è costoso; i pseudocosti costano poco ma all'inizio sono poco affidabili.
- I solver le combinano nel **reliability branching**: strong branching su una variabile finché i suoi pseudocosti non sono stati osservati un numero sufficiente di volte.
- In Gurobi il parametro **VarBranch** sceglie la strategia: -1 automatica (predefinita), 0 pseudo costo ridotto, 1 pseudo prezzo ombra, 2 massima inammissibilità, 3 strong branching.
- La selezione del nodo combina **best bound** (si sceglie il nodo con il valore $z(LP_k)$ più grande, il che fa scendere UB) e **ricerca in profondità** (che trova presto soluzioni intere).

9.2.5 Tolleranze e gap

- Un solver MIP lavora in **aritmetica a virgola mobile**: un valore è intero, un vincolo è soddisfatto, un costo ridotto è non positivo, a meno di una **tolleranza**. I valori predefiniti in Gurobi 13 sono:

Parametro	Predefinito	Significato
IntFeasTol	10^{-5}	una variabile è intera se dista meno di 10^{-5} da un intero
FeasibilityTol	10^{-6}	violazione massima di un vincolo
OptimalityTol	10^{-6}	tolleranza sui costi ridotti (ottimalità LP)
MIPGap	10^{-4}	gap relativo a cui il solver si ferma
MIPGapAbs	10^{-10}	gap assoluto a cui il solver si ferma

“Optimal solution found (tolerance 1.00e-04)” significa che la soluzione incumbent è ottima **a meno di un gap relativo di 10^{-4}** . Con un MIPGap più grande il solver si ferma con una soluzione che può **non** essere ottima, e scrive la stessa frase.

9.3 Come si legge il log di Gurobi

- Tutti i log sono prodotti da Gurobi 13.0.3 (gurobipy) con `Threads = 1` e `Seed = 0`, così che l'esecuzione sia deterministica. Con altre versioni o altri computer il numero di nodi e di iterazioni può cambiare.
- I numeri a sinistra sono numeri di riga aggiunti per i commenti; le righe `Set parameter ...` sono omesse.

9.3.1 Un problema del catalogo risolto alla radice

- Partiamo dal problema *P41* del catalogo dei problemi MIP: tre variabili intere non negative e quattro vincoli,

$$\max 2x_1 + x_2 + 3x_3 \quad \text{soggetto a} \quad x_1 + x_2 + x_3 \leq 4, \quad x_2 \leq 2, \quad x_3 \leq 3, \quad 3x_1 + x_3 \leq 6.$$

Log di Gurobi: problema *P41*, parametri predefiniti

```

01| Gurobi Optimizer version 13.0.3 build v13.0.3rc0 (linux64 - "Ubuntu 26.04 LTS")
02|
03| CPU model: Intel(R) Core(TM) i5-14500T, instruction set [SSE2|AVX|AVX2]
04| Thread count: 20 physical cores, 20 logical processors, using up to 1 threads
05|
06| Non-default parameters:
07| LogToConsole 0
08| Threads 1
09|
10| Optimize a model with 4 rows, 3 columns and 7 nonzeros (Max)
11| Model fingerprint: 0xedc792ce
12| Model has 3 linear objective coefficients
13| Variable types: 0 continuous, 3 integer (0 binary)
14| Coefficient statistics:
15|   Matrix range    [1e+00, 3e+00]
16|   Objective range [1e+00, 3e+00]
17|   Bounds range    [0e+00, 0e+00]
18|   RHS range       [2e+00, 6e+00]
19|
20| Found heuristic solution: objective 6.0000000
21| Presolve removed 2 rows and 0 columns
22| Presolve time: 0.00s
23| Presolved: 2 rows, 3 columns, 5 nonzeros
24| Variable types: 0 continuous, 3 integer (0 binary)
25|
26| Root relaxation: objective 1.100000e+01, 1 iterations, 0.00 seconds (0.00 work units)
27|
28|      Nodes      |      Current Node      |      Objective Bounds      |      Work
29|  Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
30|
31| *    0    0           0    11.0000000    11.00000    0.00%   -    0s
32|
33| Explored 1 nodes (1 simplex iterations) in 0.00 seconds (0.00 work units)
34| Thread count was 1 (of 20 available processors)
35|
36| Solution count 2: 11 6
37|
38| Optimal solution found (tolerance 1.00e-04)
39| Best objective 1.100000000000e+01, best bound 1.100000000000e+01, gap 0.0000%
```

01–08 Versione di Gurobi, computer, thread usati (1, poiché `Threads = 1`); parametri con valori non predefiniti (`Seed = 0` non compare, perché 0 è il suo valore predefinito).

- 10–13 Dimensione del modello: 4 righe, 3 colonne, 7 elementi non nulli, (Max); i tipi delle variabili (3 intere, nessuna binaria).
- 14–18 **Statistiche dei coefficienti:** il più piccolo e il più grande valore assoluto dei coefficienti. Intervalli che coprono molti ordini di grandezza segnalano difficoltà numeriche; qui tutti i valori sono compresi tra 1 e 6.
- 20 Un'euristica trova una soluzione di valore 6 prima del presolve, quindi $LB = 6$.
- 21–23 Il presolve elimina 2 righe: al modello restano 2 righe e 3 colonne.
- 26 **Rilassamento alla radice:** il valore è 11 e basta 1 iterazione del simplesso.
- 31 La soluzione del problema LP della radice è intera, quindi il nodo è risolto subito (una riga che inizia con *, vedi la Sezione 9.2.3); $LB = UB = 11$, e la soluzione è ottima.

9.3.2 Un MIP generico

- D'ora in poi usiamo un **MIP generico con 30 variabili intere**, che chiamiamo G_1 . È un problema PLI puro nella forma

$$\max \mathbf{c}'\mathbf{x} \quad \text{soggetto a} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b},$$

con $x_j \in \{0, 1, 2, 3\}$ per $j = 1, \dots, 30$ e 10 vincoli. Tutti i dati sono interi e pseudo-casuali (Python `random.Random(1)`): a_{ij} uniforme in $\{1, \dots, 20\}$, c_j uniforme in $\{5, \dots, 25\}$, e $b_i = \sum_j a_{ij}$.

- Lo script `script/genera_log.py` di questo capitolo costruisce l'istanza e produce tutti i log.

Log di Gurobi: MIP generico G_1 , parametri predefiniti

```
01| Gurobi Optimizer version 13.0.3 build v13.0.3rc0 (linux64 - "Ubuntu 26.04 LTS")
02|
03| CPU model: Intel(R) Core(TM) i5-14500T, instruction set [SSE2|AVX|AVX2]
04| Thread count: 20 physical cores, 20 logical processors, using up to 1 threads
05|
06| Non-default parameters:
07| LogToConsole 0
08| Threads 1
09|
10| Optimize a model with 10 rows, 30 columns and 300 nonzeros (Max)
11| Model fingerprint: 0xb023d504
12| Model has 30 linear objective coefficients
13| Variable types: 0 continuous, 30 integer (0 binary)
14| Coefficient statistics:
15|   Matrix range    [1e+00, 2e+01]
16|   Objective range [5e+00, 2e+01]
17|   Bounds range    [3e+00, 3e+00]
18|   RHS range       [3e+02, 4e+02]
19|
20| Found heuristic solution: objective 429.0000000
21| Presolve time: 0.00s
22| Presolved: 10 rows, 30 columns, 300 nonzeros
23| Variable types: 0 continuous, 30 integer (0 binary)
24|
25| Root relaxation: objective 5.991749e+02, 13 iterations, 0.00 seconds (0.00 work units)
26|
```

```

27|      Nodes |      Current Node |      Objective Bounds |      Work
28| Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
29|
30|      0      0 599.17487 0 6 429.00000 599.17487 39.7% - 0s
31| H 0 0 563.0000000 599.17487 6.43% - 0s
32| H 0 0 576.0000000 599.17487 4.02% - 0s
33| H 0 0 584.0000000 599.17487 2.60% - 0s
34| H 0 0 587.0000000 599.17487 2.07% - 0s
35| H 0 0 588.0000000 599.17487 1.90% - 0s
36|      0      0 597.88443 0 5 588.00000 597.88443 1.68% - 0s
37| H 0 0 589.0000000 597.88443 1.51% - 0s
38| H 0 0 592.0000000 597.88443 0.99% - 0s
39|      0      0 597.70993 0 5 592.00000 597.70993 0.96% - 0s
40|      0      0 597.67133 0 5 592.00000 597.67133 0.96% - 0s
41|      0      0 597.65097 0 4 592.00000 597.65097 0.95% - 0s
42|      0      0 597.64318 0 5 592.00000 597.64318 0.95% - 0s
43|      0      0 597.64271 0 5 592.00000 597.64271 0.95% - 0s
44|      0      0 597.56576 0 5 592.00000 597.56576 0.94% - 0s
45| H 0 0 593.0000000 597.56576 0.77% - 0s
46|      0      0 597.55314 0 6 593.00000 597.55314 0.77% - 0s
47|      0      0 597.52752 0 6 593.00000 597.52752 0.76% - 0s
48|      0      0 597.51828 0 6 593.00000 597.51828 0.76% - 0s
49|      0      0 597.47194 0 7 593.00000 597.47194 0.75% - 0s
50|      0      0 597.21692 0 6 593.00000 597.21692 0.71% - 0s
51|      0      0 597.20185 0 6 593.00000 597.20185 0.71% - 0s
52|      0      0 597.19580 0 6 593.00000 597.19580 0.71% - 0s
53|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
54|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
55|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
56|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
57|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
58|      0      0 597.07620 0 7 593.00000 597.07620 0.69% - 0s
59|      0      0 597.07620 0 7 593.00000 597.07620 0.69% - 0s
60|      0      0 597.07620 0 6 593.00000 597.07620 0.69% - 0s
61|      0      0 597.07620 0 7 593.00000 597.07620 0.69% - 0s
62|      0      0 597.07620 0 8 593.00000 597.07620 0.69% - 0s
63|      0      0 597.07074 0 7 593.00000 597.07074 0.69% - 0s
64|      0      0 597.02014 0 9 593.00000 597.02014 0.68% - 0s
65|      0      0 597.01888 0 9 593.00000 597.01888 0.68% - 0s
66|      0      0 597.01587 0 9 593.00000 597.01587 0.68% - 0s
67|      0      0 596.92506 0 8 593.00000 596.92506 0.66% - 0s
68|      0      0 596.85717 0 8 593.00000 596.85717 0.65% - 0s
69|      0      0 596.85610 0 8 593.00000 596.85610 0.65% - 0s
70|      0      0 596.85474 0 8 593.00000 596.85474 0.65% - 0s
71|      0      0 596.85333 0 8 593.00000 596.85333 0.65% - 0s
72|      0      0 596.76534 0 8 593.00000 596.76534 0.63% - 0s
73|      0      0 596.76242 0 8 593.00000 596.76242 0.63% - 0s
74|      0      0 596.75427 0 9 593.00000 596.75427 0.63% - 0s
75|      0      0 596.74106 0 8 593.00000 596.74106 0.63% - 0s
76|      0      0 596.73942 0 9 593.00000 596.73942 0.63% - 0s
77|      0      0 596.73942 0 9 593.00000 596.73942 0.63% - 0s
78|      0      2 596.64707 0 9 593.00000 596.64707 0.62% - 0s
79|
80| Cutting planes:
81| Gomory: 1
82| Lift-and-project: 1
83| MIR: 5
84| StrongCG: 3
85|
86| Explored 47 nodes (259 simplex iterations) in 0.06 seconds (0.04 work units)
87| Thread count was 1 (of 20 available processors)
88|
89| Solution count 9: 593 592 589 ... 429
90|
91| Optimal solution found (tolerance 1.00e-04)

```

92| Best objective 5.930000000000e+02, best bound 5.930000000000e+02, gap 0.0000%

10–13 10 righe, 30 colonne, 300 elementi non nulli; (**Max**): massimizzazione; le variabili sono intere ma non binarie.

20 Un'euristica trova una soluzione di valore 429: $LB = 429$.

21–23 Il presolve non cambia la dimensione del modello.

25 **Rilassamento alla radice**: il valore è 599.17487 (upper bound), dopo 13 iterazioni del simplesso.

27–28 Intestazione del **log dei nodi**: **Expl** e **Unexpl** sono i nodi esplorati e non esplorati; **Obj**, **Depth**, **IntInf** descrivono il nodo corrente; **Incumbent**, **BestBd**, **Gap** sono i due bound e il gap.

30 Nodo radice: 6 variabili intere sono frazionarie, e il gap è 39.7%.

31–35 Righe che iniziano con **H**: le euristiche primali trovano soluzioni incumbent migliori, da 563 a 588, mentre il best bound resta 599.17487.

36–78 Il **ciclo dei tagli**: ogni riga con 0 nodi esplorati è un nuovo round di tagli, e il best bound scende da 599.17487 a 596.73942. Si trovano altre soluzioni incumbent, fino a 593 (riga 45).

79 Sulla radice si esegue il branching: 2 nodi attendono nella lista, il gap è 0.62%.

81–84 Tagli aggiunti: 1 di Gomory, 1 lift-and-project, 5 MIR, 3 StrongCG.

86–92 47 nodi esplorati, 259 iterazioni del simplesso; 9 soluzioni trovate, la migliore è 593, ottima.

Qui la maggior parte del lavoro si fa alla radice: le euristiche alzano il lower bound da 429 a 593 e i tagli abbassano l'upper bound da 599.17 a 596.65. L'albero serve poi a **dimostrare** che 593 è ottimo.

9.3.3 La stessa istanza senza presolve, tagli ed euristiche

- Con **Presolve** = 0, **Cuts** = 0 e **Heuristics** = 0 il solver è vicino all'algoritmo B&B basato su LP degli appunti, e vediamo l'albero.

Log di Gurobi: G_1 , Presolve=0, Cuts=0, Heuristics=0

```
01| Gurobi Optimizer version 13.0.3 build v13.0.3rc0 (linux64 - "Ubuntu 26.04 LTS")
02|
03| CPU model: Intel(R) Core(TM) i5-14500T, instruction set [SSE2|AVX|AVX2]
04| Thread count: 20 physical cores, 20 logical processors, using up to 1 threads
05|
06| Non-default parameters:
07| Heuristics 0
08| Cuts 0
09| LogToConsole 0
10| Presolve 0
11| Threads 1
12|
```

```

13| Optimize a model with 10 rows, 30 columns and 300 nonzeros (Max)
14| Model fingerprint: 0xb023d504
15| Model has 30 linear objective coefficients
16| Variable types: 0 continuous, 30 integer (0 binary)
17| Coefficient statistics:
18|   Matrix range      [1e+00, 2e+01]
19|   Objective range   [5e+00, 2e+01]
20|   Bounds range      [3e+00, 3e+00]
21|   RHS range         [3e+02, 4e+02]
22|
23| Variable types: 0 continuous, 30 integer (0 binary)
24|
25| Root relaxation: objective 5.991749e+02, 13 iterations, 0.00 seconds (0.00 work units)
26|
27|      Nodes      |      Current Node      |      Objective Bounds      |      Work
28|  Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
29|
30|      0       0  599.17487    0   6        -  599.17487    -   -   0s
31|      0       0  598.42295    0   6        -  598.42295    -   -   0s
32|      0       2  598.42295    0   6        -  598.42295    -   -   0s
33| *    33      22                31   584.0000000  597.70712  2.35%  1.6   0s
34| *    73      18                12   593.0000000  597.40703  0.74%  1.7   0s
35|
36| Explored 119 nodes (228 simplex iterations) in 0.00 seconds (0.00 work units)
37| Thread count was 1 (of 20 available processors)
38|
39| Solution count 2: 593 584
40|
41| Optimal solution found (tolerance 1.00e-04)
42| Best objective 5.930000000000e+02, best bound 5.930000000000e+02, gap 0.0000%

```

22–23 Nessuna riga di euristica e nessuna riga di presolve: non c'è ancora una soluzione incumbent.

30–32 Il bound della radice è 599.17487 e scende a 598.42295 prima del branching sulla radice; la colonna **Incumbent** contiene un trattino (-) perché non si conosce alcuna soluzione.

33–34 Due soluzioni incumbent sono trovate come soluzioni ottime intere dei problemi LP dei nodi (*), ai nodi 33 e 73: valori 584 e 593.

36 Per dimostrare che 593 è ottimo il solver esplora 119 nodi (228 iterazioni), contro i 47 nodi con i parametri predefiniti.

9.3.4 Il gap come criterio di arresto

- Ripetiamo l'esecuzione precedente con $MIPGap = 0.01$:

Log di Gurobi: G_1 , Presolve=0, Cuts=0, Heuristics=0, MIPGap=0.01

```

01| Gurobi Optimizer version 13.0.3 build v13.0.3rc0 (linux64 - "Ubuntu 26.04 LTS")
02|
03| CPU model: Intel(R) Core(TM) i5-14500T, instruction set [SSE2|AVX|AVX2]
04| Thread count: 20 physical cores, 20 logical processors, using up to 1 threads
05|
06| Non-default parameters:
07| MIPGap 0.01
08| Heuristics 0
09| Cuts 0
10| LogToConsole 0
11| Presolve 0
12| Threads 1
13|
14| Optimize a model with 10 rows, 30 columns and 300 nonzeros (Max)
15| Model fingerprint: 0xb023d504
16| Model has 30 linear objective coefficients
17| Variable types: 0 continuous, 30 integer (0 binary)
18| Coefficient statistics:
19|   Matrix range    [1e+00, 2e+01]
20|   Objective range [5e+00, 2e+01]
21|   Bounds range   [3e+00, 3e+00]
22|   RHS range      [3e+02, 4e+02]
23|
24| Variable types: 0 continuous, 30 integer (0 binary)
25|
26| Root relaxation: objective 5.991749e+02, 13 iterations, 0.00 seconds (0.00 work units)
27|
28|      Nodes |      Current Node |      Objective Bounds |      Work
29|  Expl Unexpl |  Obj  Depth IntInf | Incumbent  BestBd  Gap | It/Node Time
30|
31|      0      0 599.17487   0   6        - 599.17487    -    -   0s
32|      0      0 598.42295   0   6        - 598.42295    -    -   0s
33|      0      2 598.42295   0   6        - 598.42295    -    -   0s
34| *    33     22          31  584.0000000 597.70712  2.35%  1.6   0s
35| *    73     18          12  593.0000000 597.40703  0.74%  1.7   0s
36|
37| Explored 74 nodes (138 simplex iterations) in 0.00 seconds (0.00 work units)
38| Thread count was 1 (of 20 available processors)
39|
40| Solution count 2: 593 584
41|
42| Optimal solution found (tolerance 1.00e-02)
43| Best objective 5.930000000000e+02, best bound 5.970000000000e+02, gap 0.6745%

```

34–35 Le stesse due soluzioni incumbent di prima, 584 e 593.

37 Il solver si ferma dopo 74 nodi invece di 119: con $LB = 593$ il best bound è 597 (il valore 597.40703 dell'ultima riga dei nodi è arrotondato per difetto dal solver, poiché tutti i dati sono interi) e il gap è $\frac{597-593}{593} \approx 0.67\% < 1\%$.

42–43 Il solver scrive “Optimal solution found (tolerance 1.00e-02)” per il valore 593.

- Il bound 597 certifica solo che 593 dista al più lo 0.67% dall'ottimo. Qui la soluzione è in effetti ottima, ma il solver non può saperlo senza esplorare i 45 nodi rimanenti.

9.3.5 Selezione della variabile di branching

- Con presolve, tagli ed euristiche disattivati, cambiamo solo **VarBranch**, su G_1 e su un secondo MIP generico G_2 (stessa costruzione, con 30 variabili intere in $\{0, \dots, 5\}$ e 8 vincoli, $b_i = \lfloor 5 \sum_j a_{ij}/3 \rfloor$). La tabella riporta i nodi esplorati (tra parentesi le iterazioni del simpleso).

VarBranch	G_1	G_2
-1 automatica	119 (228)	566 (1144)
0 pseudo costo ridotto	119 (228)	566 (1144)
1 pseudo prezzo ombra	176 (313)	645 (1270)
2 massima inammissibilità	223 (392)	527 (1097)
3 strong branching	99 (184)	497 (934)

- Su queste due istanze le differenze sono moderate: lo strong branching dà gli alberi più piccoli, e la regola della massima inammissibilità dà il più grande su G_1 e il secondo più piccolo su G_2 .
- L'ordine delle regole cambia con l'istanza, quindi il comportamento di una regola è difficile da prevedere; la scelta automatica non è mai lontana dalla migliore.
- Lo strong branching richiede meno nodi, ma ogni nodo costa di più: il solo numero di nodi non misura il tempo di calcolo.

Leggere un log significa seguire i due bound: la colonna **Incumbent** è il lato primale (euristiche e nodi interi), la colonna **BestBd** il lato duale (rilassamento, presolve, tagli, branching).

9.4 Errori tipici

Attenzione

- **Leggere “Optimal solution found” come una prova di ottimalità.** La frase riporta la tolleranza in uso: con $MIPGap = 0.01$ la soluzione può distare l'1% dall'ottimo.
- **Scambiare i bound in un problema di minimizzazione.** In un problema di minimizzazione **Incumbent** è l'upper bound e **BestBd** il lower bound.
- **Credere che l'albero spieghi tutto.** Con i parametri predefiniti, la maggior parte del lavoro del solver avviene fuori dall'albero: nei nostri log il solo nodo radice risolve $P41$ e, su G_1 , le euristiche e i tagli della radice danno la soluzione ottima e la maggior parte del bound.
- **Confrontare il numero di nodi tra esecuzioni diverse.** I numeri dipendono dalla versione del solver, dal computer e dal seme; con altri parametri la stessa istanza può richiedere 47 o 223 nodi.

- **Aspettarsi che il solver segua le regole degli appunti.** Le scelte del solver (tagli, euristiche, selezione dei nodi) non sono documentate nel dettaglio, e cambiano con la versione.

9.5 In sintesi

In sintesi

- Un solver MIP è un algoritmo branch-and-cut circondato da presolve, tagli, euristiche primali, regole di branching e di selezione dei nodi.
- Presolve e tagli alzano il lower bound (minimizzazione), le euristiche trovano la soluzione incumbent, il branching chiude il gap.
- Il log mostra i bound: **Incumbent** è il lato primale, **BestBd** il lato duale, **Gap** la loro distanza relativa.
- Il solver si ferma quando il gap scende sotto **MIPGap**: la soluzione è ottima solo a meno di quella tolleranza.
- Disattivare i componenti mostra quanto lavoro fa risparmiare ciascuno di essi su una data istanza.

Cruscotto del MIP generico G_1 . Lower bound e upper bound su $z(\text{ILP}) = 593$ nel log con i parametri predefiniti di questo capitolo:

Fase	Lower bound	Upper bound
Soluzione euristica prima della radice	429	—
Rilassamento lineare alla radice	429	599
Nodo radice, euristiche e tagli	593	596
Fine dell'albero	593	593