Cplex API Tutorial

Domenico Salvagnin

Dalla teoria alla pratica...

* Solver LP/MIP sono molto usati in pratica: perchè?
* Successo del paradigma LP/MIP dovuto alla disponibilità di solver molto efficaci da vari punti di vista:
  * efficienza
  * stabilità numerica
  * facilità d'uso/embedding

Evoluzione recente

* Negli ultimi 15 anni (circa):
  * hardware speedup: 1000x
  * miglioramenti simplesso: 1000x
  * miglioramenti branch-and-cut: 1000x

1.000.000.000 x

(meglio della legge di Moore!)

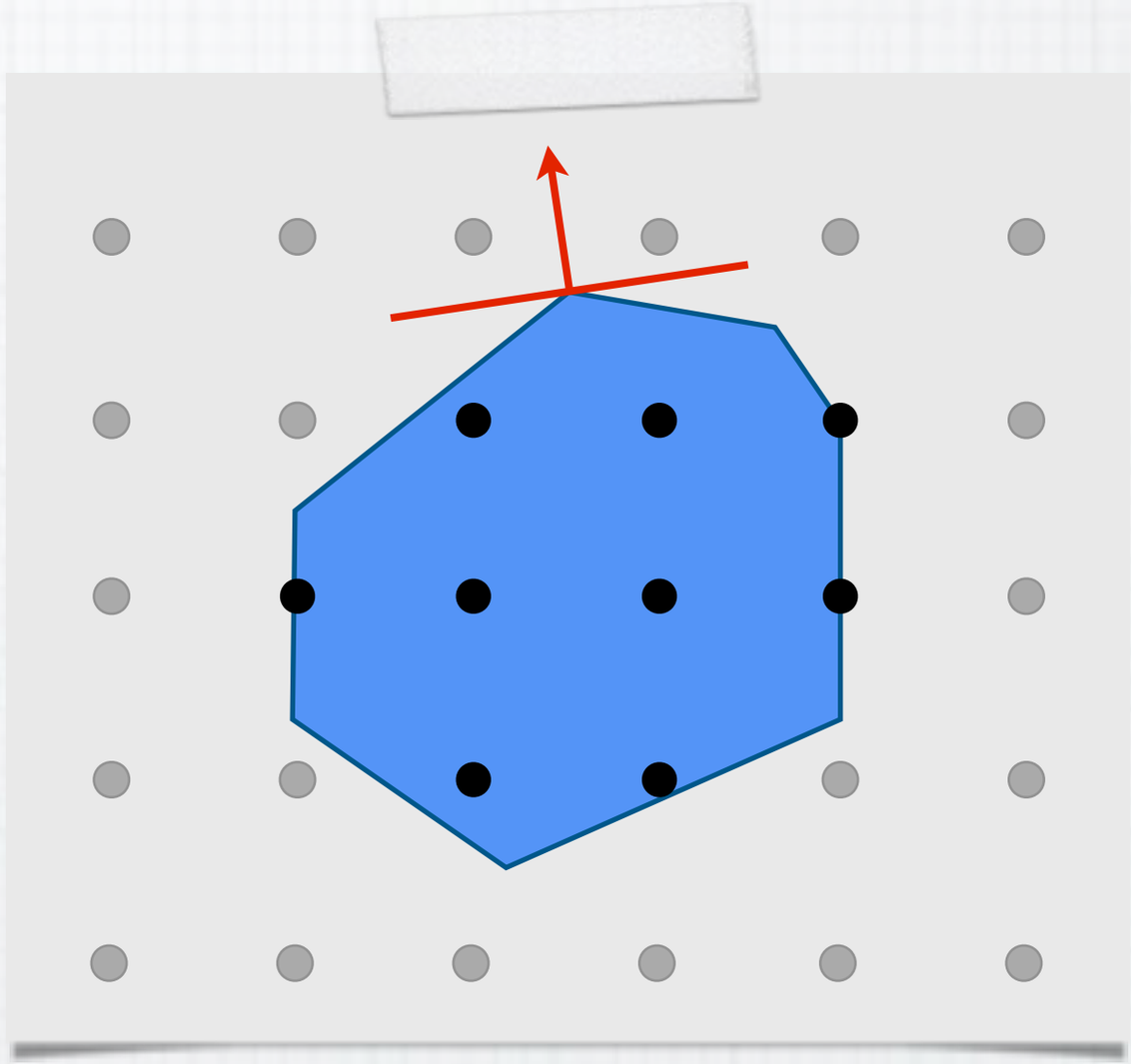
(Mixed) Integer Programming

$$\min \quad c^T x$$

$$s.t. \quad Ax \sim b$$

$$l \leq x \leq u$$

$$x_j \in \mathbb{Z}, j \in J$$

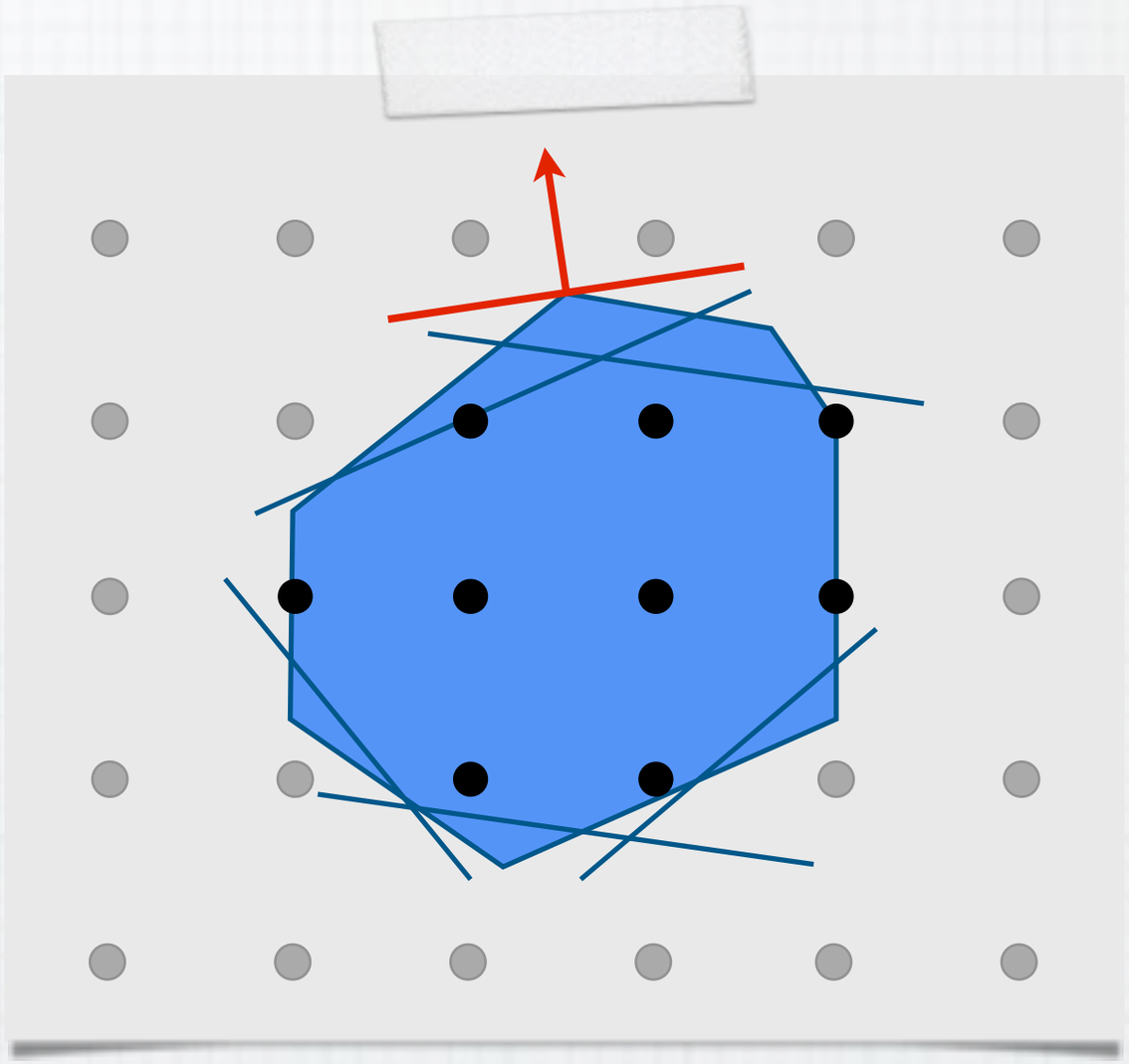


$\sim \in \{\leq, \geq, =\}$

alcuni bound +/-∞

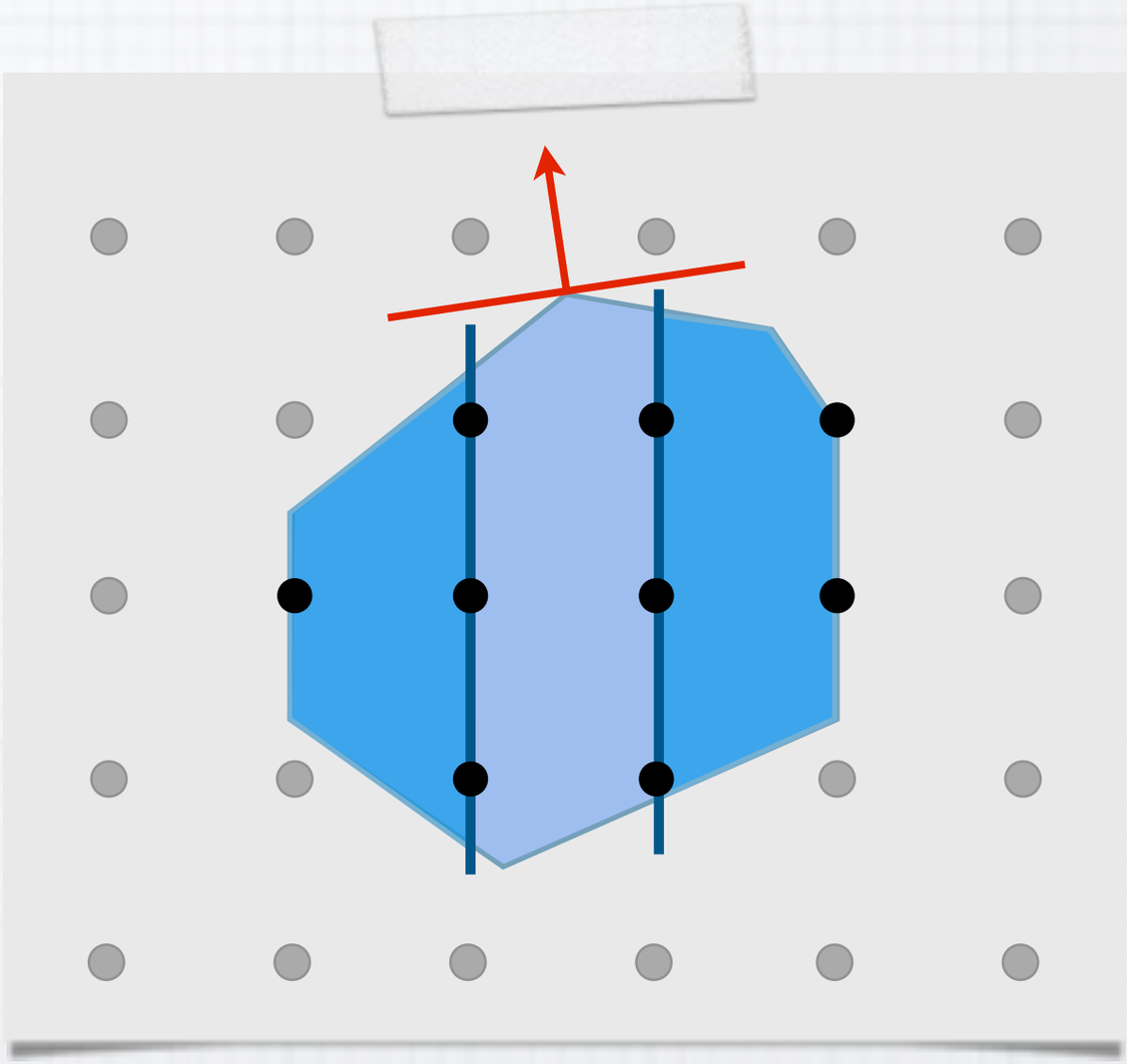
Cutting planes

* Aggiungo vincoli lineari validi per tutti i punti interi ma non per tutti quelli del rilassamento
* Tali vincoli si dicono **tagli**
* Il processo di generazione dei tagli si dice **separazione**



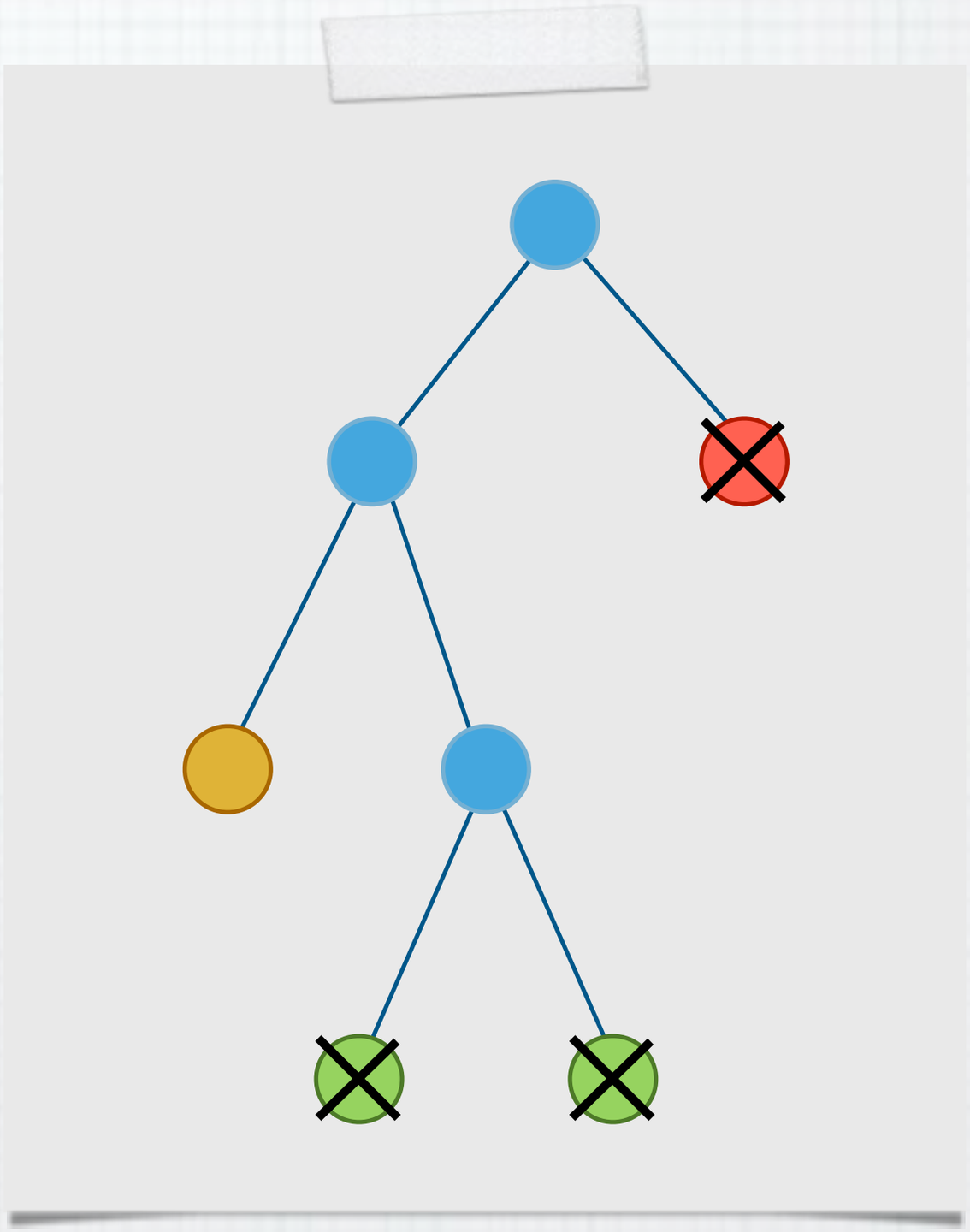
Branch-and-Bound (1)

* Approccio divide & conquer
* Se il problema è troppo difficile (soluzione frazionaria) lo divido in 2 sottoproblemi (branching)
* Li risolvo (applicando ricorsivamente il metodo)
* Prendo la soluzione migliore trovata



Branch-and-bound (2)

* Struttura ad albero
* Branching si esegue su una variabile frazionaria (ragionamento per casi)
* Molto migliore di una enumerazione esplicita. Non devo effettuare branching se:
  - sottoproblema infeasible
  - soluzione intera
  - rilassamento peggiore della migliore soluzione (bounding)



Branch-and-bound (3)

* Branch-and-bound ha sempre due bound a disposizione:
  * **bound primale**: valore della migliore soluzione feasible corrente (incumbent)
  * **bound duale**: valore del miglior rilassamento dei nodi non ancora processati
* La differenza tra i due si dice **optimality gap**
* Tali bound convergono durante l'esecuzione dell'algoritmo (alla fine gap = 0)
* L'efficienza dell'algoritmo dipende dalla velocità con cui convergono!

IBM ILOG Cplex

* uno dei primi software MIP stabili ed efficienti
* miglior solver sul mercato da quasi 20 anni (ultimamente a parimerito...)
* Numerose modalità di interazione:
  * prompt interattivo
  * librerie C (Callable Library)
  * librerie C++ (Concert Technology)
  * librerie wrapper Python/Java/.Net
  * plugin Matlab/Excel

Cplex Callable Library

* API C per l'uso degli algoritmi LP/QP/MIP/MIQP
* basata su due oggetti opachi: environment & problem
* environment: gestore licenza/parametri/callback
* problem: contiene dati del problema (variabili,vincoli,etc...)

CPX(C)ENVptr CPXopenCPLEX/CPXcloseCPLEX

CPX(C)LPptr CPXcreateprob/CPXfreeprob

Cplex Callable Library II

* i due oggetti opachi possono essere modificati/letti con delle funzioni apposite (API vera e propria)
* 99% delle funzioni della Callable Library seguono la stessa convenzione d'uso:

```
int CPXnomefunzione(environment[,problem],...);
```

codice di errore (0=ok)
CPXgeterrorstring ritorna
descrizione problema

oggetti opachi

parametri

Cplex Callable Library III

* come molte librerie C, i tipi di dato supportato sono:
  * scalari (passati per valore o puntatore): double/int
  * vettori (passati come array C): double*/int*
  * stringhe: char, char*
* per le matrici si usa una struttura dati sparsa: