MIP Practice

Linear Programming Solvers Implementation

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 due punti di vista:
  * efficienza
  * stabilità numerica

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!)

Simplesso Revised: forma standard

$$\begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \end{cases} \xrightarrow{\text{base } B} \begin{cases} \min c_B^T B^{-1} b + (C_F^T - c_B^T B^{-1} N) x_N \\ x_B + B^{-1} N x_N = B^{-1} b \\ x \geq 0 \end{cases}$$

Quantità fondamentali: $\begin{cases} \beta & = B^{-1} b \\ \pi^T & = c_B^T B^{-1} \\ z & = c_B^T B^{-1} b = c_B^T \beta = \pi^T b \end{cases}$

Sketch Algoritmo (1)

0) trova base iniziale B

1) calcola B^{-1}, β, π, z

2) [PRICING] calcola i costi ridotti:

$$d_j = c_j - \pi^T a_j \qquad\qquad j \notin \mathcal{B}$$

Se sono tutti positivi sono all'ottimo.
Altrimenti scelgo una variabile x_q con costo ridotto $d_q < 0$

Calcola $\alpha_q = B^{-1} a_q$

Sketch algoritmo (2)

3) [PIVOT] sia $I = \{i : \alpha_{iq} > 0\}$

Se I è vuoto allora il problema è unbounded.
Altrimenti effettua il ratio-test:

$$\theta = \min_{i \in I} \left\{ \frac{\beta_i}{\alpha_{iq}} \right\}$$

e determina variabile che esce dalla base x_p

4) [UPDATE]

$$\begin{cases} \beta \leftarrow \beta - \theta \alpha_q \\ z \leftarrow z + \theta d_q \\ B^{-1} \leftarrow E B^{-1} \end{cases}$$

Elementary Transformation Matrices

* Descrivono le operazioni pivot
* Possono essere salvate in modo molto compatto

$$E = \begin{bmatrix} 1 & & & \eta_1 & \\ & \ddots & & \vdots & \\ & & & \eta_p & \\ & & & \vdots & \ddots \\ & & & \eta_m & & 1 \end{bmatrix}$$

$$\eta = \left[-\frac{v_1}{v_p}, \ldots, -\frac{v_{p-1}}{v_p}, \frac{1}{v_p}, -\frac{v_{p+1}}{v_p}, \ldots, -\frac{v_m}{v+p} \right]$$

Simplesso: Operazioni Base

* (FTRAN) $\alpha = B^{-1}a \longrightarrow B\alpha = a$
* (BTRAN) $\pi^T = h^T B^{-1} \longrightarrow B^T\pi = h$

Possono essere ottenute in 2 modi:

* product form inverse (PFI)
* fattorizzazione LU

PFI: definizione e operazioni

Update step: $B^{-1} \leftarrow EB^{-1}$

Partendo da I, dopo m pivot: $B^{-1} = E_m E_{m-1} \dots E_1$

Operazioni

FTRAN	$\alpha = B^{-1}a = E_k \dots E_1 a$
BTRAN	$\pi^T = h^T B^{-1} = h^T E_k \dots E_1$
UPDATE	$B^{-1} \leftarrow E_{k+1} B^{-1} = E_{k+1} E_k \dots E_1$

Ogni tanto viene effettuata una reinversione da zero, per accorciare la catena di matrici E

Fattorizzazione LU

$$B = LU$$

upper triangular

lower triangular

FTRAN

$$LU\alpha = a \qquad \begin{matrix} L\gamma = a \\ U\alpha = \gamma \end{matrix}$$

BTRAN

$$U^T L^T \pi = h \qquad \begin{matrix} U^T\gamma = h \\ L^T\pi = \gamma \end{matrix}$$

tutto facilmente calcolabile con back/forward substitutions

Fattorizzazione LU(2)

* In pratica si ottiene la forma $L^{-1}B = U$
* I passi FTRAN e BTRAN sono comunque ottenibili usando solo tali matrici (o le relative trasposte)
* Dopo un'operazione pivot perdo la fattorizzazione => algoritmi di recover!
* Fattorizzazione LU è la più stabile per problemi di grandi dimensioni

Simplesso: forma generale

$$\begin{cases} \min c^T x \\ a_i^T x \sim b_i \\ l \leq x \leq u \end{cases}$$

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

alcuni bound +/- infinito

$$\begin{cases} \min c^T x \\ Ax + Is = b \\ l \leq x \leq u \end{cases}$$

ID	Tipo	Bounds
FX	fixed	$x_j = 0$
P	non-neg	$x_j \geq 0$
N	non-pos	$x_j \leq 0$
BD	bounded	$0 \leq x_j \leq u_j$
FR	free	x_j

Perchè?

* Generalizzazione forma standard
* Più libertà nel tipo di variabili
* Bound sulle variabili gestiti implicitamente, per avere una base più piccola
* Noi considereremo il caso in cui non ci sono variabili non positive
* Cosa cambia nel simplesso revised?

Basic Solutions

* Cambia il concetto di soluzione feasible: x è feasible se ogni componente è dentro i rispettivi bound
* variabili fuori base non necessariamente nulle (anche all'upper bound), ma determinano ancora univocamente quelle in base

var. in base $\mathcal{B}$

var. fuori base all'upper $\mathcal{U}$

$$
\begin{aligned}
x_B &= B^{-1}(b - Nx_F) \\
&= B^{-1}(b - \sum_{k \in \mathcal{U}} a_k u_k) \\
&= B^{-1} b_u
\end{aligned}
$$

Optimality Conditions

La formula dei costi ridotti non cambia, ma le variabili fuori base possono comportarsi diversamente:

ID	Valore	Cond. costo ridotto
FX	0	qualsiasi
P	0	$d_j \geq 0$
BD	0	$d_j \geq 0$
BD	u_j	$d_j \leq 0$
FR	0	$d_j = 0$

non entra mai in base!

novità

Cambio base

Ci sono 2 casi: $d_q < 0$ e $d_q > 0$. Vediamo il primo

$$Bx_B(t) + ta_q = b_u$$

t indica di quanto si muove la var. che entra

$$x_B(t) = B^{-1}b_u - tB^{-1}a_q$$

β α_q

$$x_{\mathcal{B}(i)}(t) = \beta_i - t\alpha_{iq}$$

Devo fare in modo che tutte le variabili in base rimangano all'interno dei rispettivi bound (sempre vero per le var. free)

una var. free in base non esce mai

Cambio base (2)

Definiamo:

$$I^+ = \{i : \alpha_{iq} > 0,\ \text{type}(x_{\mathcal{B}(i)}) \in \{\text{FX, P, BD}\}\}$$

$$I^- = \{i : \alpha_{iq} < 0,\ \text{type}(x_{\mathcal{B}(i)}) \in \{\text{FX, BD}\}\}$$

allora:

$$\theta^+ = \min_{i \in I^+} \left\{ \frac{\beta_i}{\alpha_{iq}} \right\} \qquad \theta^- = \min_{i \in I^-} \left\{ \frac{\beta_i - u_{\mathcal{B}(i)}}{\alpha_{iq}} \right\}$$

$$\theta = \min\{\theta^+, \theta^-, u_q\}$$

bound flip: molto vantaggioso!

Cambio base (3)

Caso $d_q > 0$: Definiamo:

$$I^+ = \{i : \alpha_{iq} > 0,\ \text{type}(x_{\mathcal{B}(i)}) \in \{\text{FX, BD}\}\}$$

$$I^- = \{i : \alpha_{iq} < 0,\ \text{type}(x_{\mathcal{B}(i)}) \in \{\text{FX, P, BD}\}\}$$

allora:

$$\theta^- = \max_{i \in I^-} \left\{ \frac{\beta_i}{\alpha_{iq}} \right\} \qquad \theta^+ = \max_{i \in I^+} \left\{ \frac{\beta_i - u_{\mathcal{B}(i)}}{\alpha_{iq}} \right\}$$

$$\theta = \max\{\theta^+, \theta^-, -u_q\}$$

bound flip: molto vantaggioso!

Qualche link...

* COIN CLP: http://www.coin-or.org/projects/Clp.xml
* ZIB SoPLEX: http://soplex.zib.de
* GNU GLPK: http://www.gnu.org/software/glpk
* Maros "Computational Techniques of the Simplex Method", 2002