Logica e variabili binarie

Indice

Proposizioni e variabili binarie

Letterali, clausole, forma normale congiuntiva

Da CNF a vincoli lineari

Implicazioni logiche

Scindere un'implicazione: quando si può

Contare: al più uno, almeno uno, esattamente uno

Operatori logici e operatori insiemistici

Verificare la traduzione, non fidarsene

Cinque esercizi

I vincoli logici dentro un modello di ottimizzazione

Quello che questo capitolo ha messo in mano

Obiettivi (1/2)

Alla fine di questo capitolo sarai in grado di:

- portare una condizione logica fra decisioni binarie in forma normale congiuntiva;
- tradurre ogni clausola in una disuguaglianza lineare e dimostrare che la traduzione è esatta;
- scrivere implicazioni, la loro contronominale e il *se e solo se*, e dire quando un'implicazione si può spezzare;

Obiettivi (2/2)

Alla fine di questo capitolo sarai in grado di:

- scrivere vincoli di cardinalità: al più uno, almeno uno, esattamente uno;
- aggiungere vincoli logici a un modello di ottimizzazione e risolverlo in `gurobipy`.

Prerequisiti

- Dal modulo *Analisi matematica 1*, il capitolo «Basi di logica e tecniche di dimostrazione»: proposizioni, connettivi logici e loro tavole di verità
- Da questo modulo, Capitolo 1 «Che cos'è un modello MIP»: le variabili binarie e i pattern di set covering, packing e partitioning

Di cosa parliamo

- Una variabile binaria è una risposta «**sì**/**no**»
- In questo capitolo impariamo a tradurre in **vincoli lineari** le condizioni logiche fra quelle risposte
- E — soprattutto — a **dimostrare** che la traduzione è esatta

Proposizioni e variabili binarie

Introduzione: Proposizioni e variabili binarie

- Partiamo dal legame fra una **proposizione** (vera o falsa) e una **variabile binaria**
- Introduciamo le **espressioni booleane**, costruite con `AND`, `OR`, `NOT`
- Vediamo il problema della **soddisfacibilità**: esiste un'assegnazione che rende vera l'espressione?
- È il linguaggio con cui descriveremo le condizioni logiche nei modelli MIP

Una proposizione, una variabile

- Una **funzione booleana** restituisce uno dei due valori `TRUE` e `FALSE`
- La si rappresenta con una variabile binaria

$$x \in \{0, 1\}, \qquad \begin{aligned} x = 1 &\iff \text{la proposizione è } \texttt{TRUE} \\ x = 0 &\iff \text{la proposizione è } \texttt{FALSE} \end{aligned}$$

- Un'**espressione booleana** si costruisce con:
  - variabili binarie
  - i tre operatori `AND` (congiunzione, $\wedge$), `OR` (disgiunzione, $\vee$) e `NOT` (negazione, $\neg$)
  - le parentesi

Esempio: due espressioni booleane

$$\texttt{NOT}\ x_a \quad \texttt{OR} \quad \big((x_b \quad \texttt{OR} \quad x_c) \quad \texttt{AND} \quad (x_d \quad \texttt{OR} \quad x_e)\big)$$

$$(x_a \quad \texttt{OR} \quad x_b) \quad \texttt{AND} \quad (x_c \quad \texttt{OR} \quad x_d) \quad \texttt{AND} \quad \texttt{NOT}\ x_e$$

Soddisfacibilità

- Il **problema di soddisfacibilità booleana** chiede se esiste un'assegnazione di valori logici alle variabili che renda vera l'espressione
- Se esiste, l'espressione è **soddisfacibile**
- Altrimenti è **insoddisfacibile**

Esempio: soddisfacibili e no (1/2)

- La formula

$$\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ ((x_b\ \ \texttt{OR}\ \ x_c)\ \ \texttt{AND}\ \ (x_d\ \ \texttt{OR}\ \ x_e))$$

  è soddisfatta, per esempio, da $x_a = 0$ (le altre qualsiasi)

- La formula

$$(x_a\ \ \texttt{OR}\ \ x_b)\ \ \texttt{AND}\ \ (x_c\ \ \texttt{OR}\ \ x_d)\ \ \texttt{AND}\ \ \texttt{NOT}\ x_e$$

  è soddisfatta da $x_a = 1$, $x_c = 1$, $x_b = x_d = x_e = 0$

Esempio: soddisfacibili e no (2/2)

- La formula

$$(x_a \ \texttt{OR} \ x_b) \ \texttt{AND} \ (\texttt{NOT} \ x_a \ \texttt{OR} \ x_b)$$
$$\texttt{AND} \ (x_a \ \texttt{OR} \ \texttt{NOT} \ x_b) \ \texttt{AND} \ (\texttt{NOT} \ x_a \ \texttt{OR} \ \texttt{NOT} \ x_b)$$

  è **insoddisfacibile**

- Le quattro assegnazioni possibili di (x_a, x_b) falsificano, ciascuna, una delle quattro clausole

Riepilogo

Proposizioni e variabili binarie:

- Una proposizione vera/falsa $\leftrightarrow$ una variabile $x \in \{0, 1\}$
- Le espressioni booleane usano `AND` , `OR` , `NOT` e parentesi

Soddisfacibilità:

- Un'espressione è soddisfacibile se qualche assegnazione la rende vera
- Altrimenti è insoddisfacibile

Letterali, clausole, forma normale congiuntiva

Introduzione: Letterali, clausole, CNF

- Definiamo tre parole chiave: **letterale**, **clausola**, **forma normale congiuntiva** (CNF)
- La CNF è la forma da cui partiremo per scrivere i vincoli lineari
- Vediamo le **equivalenze** che permettono di portare in CNF qualsiasi espressione booleana

Le tre parole

- Un **letterale** è una variabile (x_a, letterale positivo) o la sua negazione (`NOT` x_a, letterale negativo)
- Una **clausola** è una disgiunzione (`OR`) di letterali, o un singolo letterale
- Un'espressione è in **forma normale congiuntiva** (CNF) se è una congiunzione (`AND`) di clausole, o una singola clausola

- Delle due espressioni dell'esempio precedente, la seconda è in CNF e la prima no
- Ogni espressione booleana **può** essere riscritta in CNF
- Per farlo si usano le equivalenze che seguono

Le equivalenze che servono (1/2)

Per ogni terna di variabili binarie $x_a, x_b, x_c \in \{0, 1\}$:

Distributive (D1), (D2):

$$x_a \texttt{ AND } (x_b \texttt{ OR } x_c) \iff (x_a \texttt{ AND } x_b) \texttt{ OR } (x_a \texttt{ AND } x_c)$$

$$x_a \texttt{ OR } (x_b \texttt{ AND } x_c) \iff (x_a \texttt{ OR } x_b) \texttt{ AND } (x_a \texttt{ OR } x_c)$$

Leggi di De Morgan (M1), (M2):

$$\texttt{NOT}\,(x_a \texttt{ OR } x_b) \iff \texttt{NOT}\, x_a \texttt{ AND } \texttt{NOT}\, x_b$$

$$\texttt{NOT}\,(x_a \texttt{ AND } x_b) \iff \texttt{NOT}\, x_a \texttt{ OR } \texttt{NOT}\, x_b$$

Le equivalenze che servono (2/2)

Assorbimento (A1), (A2):

$$x_a \ \texttt{OR} \ (x_a \ \texttt{AND} \ x_b) \iff x_a$$

$$x_a \ \texttt{AND} \ (x_a \ \texttt{OR} \ x_b) \iff x_a$$

Doppia negazione (N):

$$\texttt{NOT}\,(\texttt{NOT}\ x_a) \iff x_a$$

- Valgono anche con più di due termini fra parentesi e con espressioni al posto delle variabili

Come si dimostrano

- Ogni equivalenza si dimostra **per casi**
- Con due variabili ci sono 4 assegnazioni, con tre ce ne sono 8
- Si verifica che i due membri hanno lo stesso valore su ciascuna assegnazione
- Lo script `cap02_logica.py` esegue questo controllo su tutte e sette le equivalenze

Raccogliere: la (D1) letta al contrario

- Letta da destra a sinistra, la distributiva (D1) permette di **raccogliere** due congiunzioni che condividono un letterale:

$$\begin{aligned} & (x_a \ \texttt{AND} \ x_b) \ \texttt{OR} \ (x_a \ \texttt{AND} \ x_c) \\ \Longleftrightarrow \ & x_a \ \texttt{AND} \ (x_b \ \texttt{OR} \ x_c) \end{aligned}$$

- Serve quando si porta in CNF una disgiunzione di più congiunzioni con letterali in comune
- Riduce, passo dopo passo, il numero di termini da distribuire

Esempio: due trasformazioni in CNF (1/3)

- Per la formula

$$\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ ((x_b\ \ \texttt{OR}\ \ x_c)\ \ \texttt{AND}\ \ (x_d\ \ \texttt{OR}\ \ x_e))$$

basta la (D2), e diventa

$$(\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ x_b\ \ \texttt{OR}\ \ x_c)\ \ \texttt{AND}\ \ (\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ x_d\ \ \texttt{OR}\ \ x_e)$$

Esempio: due trasformazioni in CNF (2/3)

- La formula

$$(x_a \ \texttt{AND} \ x_b) \ \texttt{OR} \ (x_a \ \texttt{AND} \ x_c) \ \texttt{OR} \ (x_b \ \texttt{AND} \ x_c)$$

  dice «almeno due fra a, b, c»
- Si raccoglie x_a nelle prime due:

$$\big[x_a \ \texttt{AND} \ (x_b \ \texttt{OR} \ x_c)\big] \ \texttt{OR} \ (x_b \ \texttt{AND} \ x_c)$$

- Si applica la (D2) e si ottengono i due fattori

$$x_a \ \texttt{OR} \ (x_b \ \texttt{AND} \ x_c)$$

$$\text{e} \quad (x_b \ \texttt{OR} \ x_c) \ \texttt{OR} \ (x_b \ \texttt{AND} \ x_c)$$

Esempio: due trasformazioni in CNF (3/3)

- Il secondo fattore si riduce a $(x_b \ \texttt{OR} \ x_c)$ per assorbimento (A1)
- Il primo si distribuisce di nuovo con la (D2)
- In tutto:

$$(x_a \ \texttt{OR} \ x_b) \ \texttt{AND} \ (x_a \ \texttt{OR} \ x_c) \ \texttt{AND} \ (x_b \ \texttt{OR} \ x_c)$$

- Sarà la chiave dell'Esercizio 4

Riepilogo

Le tre parole:

- Letterale: x oppure `NOT` x
- Clausola: `OR` di letterali
- CNF: `AND` di clausole

Portare in CNF:

- Distributive, De Morgan, assorbimento, doppia negazione
- Ogni equivalenza si dimostra per casi, enumerando le assegnazioni
- Raccogliere con la (D1) al contrario riduce i termini da distribuire

Da CNF a vincoli lineari

Introduzione: Da CNF a vincoli lineari

- Questo è il cuore del capitolo: da un'espressione in CNF a un **sistema di vincoli lineari**
- Bastano **tre regole**
- Vediamo perché la traduzione è **esatta** e in che forma conviene scrivere i vincoli

La traduzione, in tre regole

La traduzione, in tre regole

Un'espressione booleana in CNF è equivalente a un sistema di vincoli lineari sulle stesse variabili binarie:

1. ogni **clausola** diventa un vincolo di disuguaglianza ≥ 1;
2. ogni `OR` dentro la clausola diventa un $+$;
3. ogni letterale negativo `NOT` x diventa $1 - x$.

Perché è esatta (1/2)

- Una clausola con letterali positivi P e negativi N diventa

$$\sum_{i \in P} x_i + \sum_{i \in N} (1 - x_i) \geq 1$$

- cioè, portando le costanti a destra,

$$\sum_{i \in P} x_i - \sum_{i \in N} x_i \geq 1 - |N|$$

Perché è esatta (2/2)

- $x_i = 1$ contribuisce 1 al membro sinistro se il letterale è positivo e soddisfatto
- $1 - x_i = 1$ contribuisce 1 se il letterale è negativo e soddisfatto
- La somma conta **quanti letterali della clausola sono veri**: chiedere che sia ≥ 1 è esattamente chiedere che la clausola sia vera
- La congiunzione di clausole diventa il sistema di tutti i vincoli, perché tutte devono valere insieme

La forma in cui si scrive il vincolo

Osservazione 1: La forma in cui si scrive il vincolo

Con due o più letterali negativi si preferisce, per leggibilità, la forma equivalente ottenuta moltiplicando per -1*: per esempio*

$$1 - x_1 + 1 - x_6 + x_7 \geq 1 \quad \textit{si scrive} \quad x_1 + x_6 - x_7 \leq 1.$$

È lo stesso vincolo. Quello che non cambia mai è il numero di vincoli: ***uno per clausola****, e le clausole si contano dopo la semplificazione (via le tautologie* x `OR` `NOT` x *e le clausole assorbite da una loro sottoclausola).*

Esempio: tre traduzioni (1/2)

$$(\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ x_b\ \ \texttt{OR}\ \ x_c)\ \ \texttt{AND}\ \ (\texttt{NOT}\ x_a\ \ \texttt{OR}\ \ x_d\ \ \texttt{OR}\ \ x_e)$$

$$\iff \begin{cases} -x_a + x_b + x_c \geq 0 \\ -x_a + x_d + x_e \geq 0 \end{cases}$$

$$(x_a\ \ \texttt{OR}\ \ x_b)\ \ \texttt{AND}\ \ (x_c\ \ \texttt{OR}\ \ x_d)\ \ \texttt{AND}\ \ \texttt{NOT}\ x_e$$

$$\iff \begin{cases} x_a + x_b \geq 1 \\ x_c + x_d \geq 1 \\ x_e \leq 0 \end{cases}$$

Esempio: tre traduzioni (2/2)

$$(x_a \ \texttt{OR} \ x_b) \ \texttt{AND} \ (\texttt{NOT} \ x_a \ \texttt{OR} \ x_b)$$
$$\texttt{AND} \ (x_a \ \texttt{OR} \ \texttt{NOT} \ x_b) \ \texttt{AND} \ (\texttt{NOT} \ x_a \ \texttt{OR} \ \texttt{NOT} \ x_b)$$

$$\iff \begin{cases} x_a + x_b \geq 1 \\ -x_a + x_b \geq 0 \\ x_a - x_b \geq 0 \\ x_a + x_b \leq 1 \end{cases}$$

- L'ultimo sistema è privo di soluzioni binarie, come l'espressione era insoddisfacibile
- I primi tre vincoli danno $x_a = x_b = 1$, che viola il quarto

Riepilogo

Da CNF a vincoli lineari:

- Clausola $\rightarrow$ vincolo ≥ 1; $\texttt{OR} \rightarrow +$; $\texttt{NOT}\ x \rightarrow 1 - x$
- La somma conta i letterali veri della clausola: la traduzione è esatta

In pratica:

- Con molti letterali negativi si scrive la forma $\leq$, equivalente
- Sempre **un vincolo per clausola**, contando le clausole dopo la semplificazione

Implicazioni logiche

Introduzione: Implicazioni logiche

- Nei modelli le condizioni logiche sono quasi sempre **implicazioni**: «se scelgo *a* allora devo scegliere *b*»
- Vediamo come si traduce l'implicazione fra due variabili
- E poi la ricetta generale, con antecedente e conseguente composti, e quanto costa ciascun caso in numero di vincoli

L'implicazione fra due variabili

L'implicazione fra due variabili

L'implicazione logica $x_a \Rightarrow x_b$ («se a allora b») equivale all'espressione booleana

$$\texttt{NOT}\ x_a \quad \texttt{OR} \quad x_b,$$

già in CNF, e quindi al vincolo lineare

$$1 - x_a + x_b \geq 1 \iff x_b - x_a \geq 0.$$

La contronominale è la stessa cosa

- Per contrapposizione, $x_a \Rightarrow x_b$ implica anche la contronominale

$$\texttt{NOT}\ x_b \Rightarrow \texttt{NOT}\ x_a$$

- Non è un secondo vincolo: la sua espressione booleana è

$$\texttt{NOT}\,(\texttt{NOT}\ x_b) \quad \texttt{OR} \quad \texttt{NOT}\ x_a$$

  che per doppia negazione (N) è la stessa $\texttt{NOT}\ x_a \quad \texttt{OR} \quad x_b$
- Un solo vincolo impone entrambe le letture: nel modello conviene scrivere quella che si legge meglio nel contesto applicativo

Implicazione non è equivalenza

Osservazione 2: Implicazione non è equivalenza

$x_b - x_a \geq 0$ dice «se a allora b», non «a se e solo se b»: ammette $x_a = 0$, $x_b = 1$.

*Per il «se e solo se» servono **due** implicazioni, cioè $x_a = x_b$, oppure una implicazione più un argomento di ottimalità, che è il modo in cui si scrive quando una delle due implicazioni segue dal segno del coefficiente nell'obiettivo.*

La ricetta generale

- La stessa ricetta vale quando antecedente e conseguente sono espressioni booleane qualsiasi:

1. si scrive `NOT` (antecedente) `OR` (conseguente)
2. si porta in CNF
3. si traduce con le tre regole

Esempio: le quattro combinazioni di base (1/2)

Le espressioni in CNF:

Implicazione	Espressione in CNF
x_a AND $x_b \Rightarrow x_c$	NOT x_a OR NOT x_b OR x_c
x_a OR $x_b \Rightarrow x_c$	(NOT x_a OR x_c) AND (NOT x_b OR x_c)
$x_a \Rightarrow x_b$ AND x_c	(NOT x_a OR x_b) AND (NOT x_a OR x_c)
$x_a \Rightarrow x_b$ OR x_c	NOT x_a OR x_b OR x_c

Esempio: le quattro combinazioni di base (2/2)

I vincoli lineari e il loro numero:

Implicazione	Vincoli	#
x_a AND $x_b \Rightarrow x_c$	$x_a + x_b - x_c \leq 1$	1
x_a OR $x_b \Rightarrow x_c$	$x_c - x_a \geq 0, \quad x_c - x_b \geq 0$	2
$x_a \Rightarrow x_b$ AND x_c	$x_b - x_a \geq 0, \quad x_c - x_a \geq 0$	2
$x_a \Rightarrow x_b$ OR x_c	$x_b + x_c - x_a \geq 0$	1

Come leggere la tabella

- Una **disgiunzione nell'antecedente** e una **congiunzione nel conseguente** costano **due** vincoli
- Una congiunzione nell'antecedente e una disgiunzione nel conseguente ne costano **uno**
- È il motivo per cui il vincolo aggregato $\sum_j x_{jm} \leq k\, y_m$ e quello disaggregato $x_{jm} \leq y_m$ hanno costi diversi in numero di righe, e forze diverse nel rilassamento

Esempio: un antecedente con tre letterali

$$x_a \quad \texttt{AND} \quad \texttt{NOT}\ x_b \quad \texttt{AND} \quad x_c \Rightarrow \texttt{NOT}\ x_d$$

$$\iff \quad \texttt{NOT}\ x_a \quad \texttt{OR} \quad x_b \quad \texttt{OR} \quad \texttt{NOT}\ x_c \quad \texttt{OR} \quad \texttt{NOT}\ x_d$$

$$\iff \quad x_a - x_b + x_c + x_d \leq 2$$

- Tre letterali negativi nella clausola: si usa la forma $\leq$

Riepilogo

Implicazioni:

- $x_a \Rightarrow x_b$ equivale a `NOT` x_a `OR` x_b, cioè $x_b - x_a \geq 0$
- La contronominale è lo stesso vincolo
- Implicazione non è equivalenza: per il «se e solo se» servono due implicazioni

Caso generale:

- `NOT` (antecedente) `OR` (conseguente), CNF, traduzione
- `OR` nell'antecedente e `AND` nel conseguente costano due vincoli

Scindere un'implicazione: quando si può

Introduzione: Scindere un'implicazione

- Spesso conviene **spezzare** un'implicazione composta in implicazioni più semplici
- Ci sono due scissioni valide e una che **non** lo è
- Sbagliare la scissione produce un modello che può escludere l'ottimo: vediamo perché

Due scissioni valide e una no

Valide:

$$(x_a \quad \texttt{OR} \quad x_b) \Rightarrow x_c \iff (x_a \Rightarrow x_c) \quad \texttt{AND} \quad (x_b \Rightarrow x_c) \quad \text{(S1)}$$

$$x_a \Rightarrow (x_b \quad \texttt{AND} \quad x_c) \iff (x_a \Rightarrow x_b) \quad \texttt{AND} \quad (x_a \Rightarrow x_c) \quad \text{(S2)}$$

Non valida: la scissione analoga con antecedente **congiunzione**

$$(x_a \quad \texttt{AND} \quad x_b) \Rightarrow x_c \quad \textbf{non} \iff \quad (x_a \Rightarrow x_c) \quad \texttt{AND} \quad (x_b \Rightarrow x_c)$$

Il controesempio, e in che verso sbaglia

Osservazione 3: Il controesempio, e in che verso sbaglia

*Con $x_a = 1$, $x_b = 0$, $x_c = 0$ l'implicazione $(x_a \ \ \texttt{AND} \ \ x_b) \Rightarrow x_c$ è **vera** — il suo antecedente è `FALSE` — ma $x_a \Rightarrow x_c$ è **falsa**.*

*La congiunzione delle due implicazioni scisse è quindi **strettamente più forte** dell'originale: taglia soluzioni che il problema ammette. Sbagliare qui non produce un modello «un po' diverso», produce un modello che può escludere l'ottimo.*

La strada più corta

- Le due scissioni valide sono spesso la strada più corta
- Nella tabella delle quattro combinazioni, le righe con due vincoli si ottengono direttamente da (S1) o (S2)
- Senza passare da De Morgan e dalla distributiva

Riepilogo

Scissioni:

- Antecedente `OR` : si scinde in due implicazioni (S1)
- Conseguente `AND` : si scinde in due implicazioni (S2)
- Antecedente `AND` : **non** si scinde

Perché conta:

- La scissione sbagliata è strettamente più forte: taglia soluzioni ammissibili e può escludere l'ottimo

Contare: al più uno, almeno uno, esattamente uno

Introduzione: Contare

- Molte condizioni sono di **cardinalità**: «almeno uno», «al più uno», «esattamente uno», «almeno p»...
- Su un insieme I di variabili binarie sono **già lineari** e non richiedono la CNF
- Vedremo che un vincolo di cardinalità può valere più di molte clausole

Le condizioni di cardinalità (1/2)

Condizione	Vincolo	Nota
almeno uno	$\sum_{i \in I} x_i \geq 1$	è la clausola $\bigvee_{i \in I} x_i$: *set covering*
al più uno	$\sum_{i \in I} x_i \leq 1$	*set packing*; equivale alle $\binom{\lvert I \rvert}{2}$ clausole `NOT` x_i `OR` `NOT` x_j, ma in un solo vincolo e con un rilassamento più stretto
esattamente uno	$\sum_{i \in I} x_i = 1$	*set partitioning*: le due precedenti insieme

Le condizioni di cardinalità (2/2)

Condizione	Vincolo	Nota
almeno p	$\sum_{i \in I} x_i \geq p$	in CNF servirebbero $\binom{\lvert I \rvert}{\lvert I \rvert - p + 1}$ clausole
al più p	$\sum_{i \in I} x_i \leq p$	in CNF servirebbero $\binom{\lvert I \rvert}{p+1}$ clausole
nessuno	$\sum_{i \in I} x_i \leq 0$	cioè $x_i = 0$ per ogni i

Un vincolo di cardinalità val più di molte clausole (1/2)

- «Al più uno fra tre» si può scrivere come tre clausole:

$$\begin{aligned}
\texttt{NOT}\ x_1 \quad \texttt{OR} \quad \texttt{NOT}\ x_2 &\iff x_1 + x_2 \leq 1 \\
\texttt{NOT}\ x_1 \quad \texttt{OR} \quad \texttt{NOT}\ x_3 &\iff x_1 + x_3 \leq 1 \\
\texttt{NOT}\ x_2 \quad \texttt{OR} \quad \texttt{NOT}\ x_3 &\iff x_2 + x_3 \leq 1
\end{aligned}$$

- oppure come il solo

$$x_1 + x_2 + x_3 \leq 1$$

Un vincolo di cardinalità val più di molte clausole (2/2)

- I due sistemi hanno le **stesse** soluzioni binarie, ma non lo stesso rilassamento
- $x_1 = x_2 = x_3 = 1/2$ soddisfa le tre clausole e viola il vincolo aggregato
- La versione compatta è più corta **e** più forte: un caso in cui non c'è niente da scambiare
- Quando si può contare, si conta; la CNF serve quando la condizione non è una cardinalità

Riepilogo

Cardinalità:

- Almeno uno (≥ 1, set covering), al più uno (≤ 1, set packing), esattamente uno ($= 1$, set partitioning)
- Sono già lineari: non serve la CNF

Contare conviene:

- Un vincolo al posto di molte clausole
- «Al più uno» contato ha un rilassamento più stretto delle clausole a coppie

Operatori logici e operatori insiemistici

Introduzione: Operatori logici e insiemistici

- Gli operatori logici hanno un corrispondente negli **insiemi**
- Le equivalenze della logica diventano identità insiemistiche
- Così si ha una seconda dimostrazione delle stesse regole, con i diagrammi di Venn

Dalla logica agli insiemi

- Associamo a ogni variabile binaria un sottoinsieme di un universo U
- $x_a = 1$ significa «l'elemento scelto sta in A»

Logica	Insiemi	
x_a `AND` x_b	$A \cap B$	intersezione
x_a `OR` x_b	$A \cup B$	unione
`NOT` x_a	$\overline{A}$	complemento

Le equivalenze diventano identità insiemistiche

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad \text{(D1)}$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad \text{(D2)}$$
$$\overline{A \cup B} = \overline{A} \cap \overline{B} \quad \text{(M1)}$$
$$\overline{A \cap B} = \overline{A} \cup \overline{B} \quad \text{(M2)}$$
$$A \cup (A \cap B) = A \quad \text{(A1)}$$
$$A \cap (A \cup B) = A \quad \text{(A2)}$$

- Chi preferisce i diagrammi di Venn ha una seconda dimostrazione; chi preferisce l'enumerazione ha la prima
- Le due si controllano a vicenda

Riepilogo

Logica e insiemi:

- `AND` $\leftrightarrow \cap$, `OR` $\leftrightarrow \cup$, `NOT` $\leftrightarrow$ complemento
- Distributive, De Morgan e assorbimento valgono anche per gli insiemi
- Venn ed enumerazione: due dimostrazioni che si controllano a vicenda

Verificare la traduzione, non fidarsene

Introduzione: Verificare la traduzione

- Una traduzione in vincoli lineari va **verificata**, non presa sulla fiducia
- Con poche variabili la verifica è un'enumerazione: è la dimostrazione per casi
- Il modulo `booleane.py` fa questa verifica e tutto il resto del lavoro del capitolo

Il criterio operativo

- Una traduzione in vincoli lineari è corretta quando, per **ogni** assegnazione binaria delle variabili coinvolte:

  l'espressione è vera $\iff$ tutti i vincoli sono soddisfatti

- Con poche variabili è un'enumerazione di 2^n casi, che si fa in un istante
- È la dimostrazione, per casi, del risultato
- Il modulo `python/booleane.py` la esegue: tutte le traduzioni degli esercizi sono verificate così

Il modulo `booleane.py`

Il modulo fa anche il resto del lavoro del capitolo:

1. elimina le implicazioni ($A \Rightarrow B$ diventa `NOT` A `OR` B)
2. spinge le negazioni sui letterali con De Morgan
3. distribuisce l' `OR` sull' `AND`
4. scarta le clausole tautologiche e quelle assorbite
5. applica le tre regole di traduzione

- Non è un solutore SAT: è la trascrizione, riga per riga, delle regole di questo capitolo

Riepilogo

Verificare:

- Corretta $\iff$ per ogni assegnazione: espressione vera $\iff$ vincoli soddisfatti
- Con n variabili: 2^n casi, un istante

`booleane.py`:

- Implicazioni, De Morgan, distributiva, semplificazione, traduzione
- La trascrizione delle regole del capitolo

Cinque esercizi

Introduzione: Cinque esercizi

- Mettiamo in pratica la ricetta su cinque gruppi di condizioni logiche fra **progetti**
- Implicazioni dirette, negate, composte, «almeno due fra», scissioni
- Per ogni condizione: espressione in CNF, vincoli lineari, numero di vincoli

Le variabili

- In tutti e cinque gli esercizi i progetti sono numerati
- Per ciascun progetto p la variabile binaria è

$$x_p = \begin{cases} 1 & \text{se il progetto } p \text{ è scelto} \\ 0 & \text{altrimenti} \end{cases}$$

- Per ogni condizione si chiede: l'espressione in CNF, i vincoli lineari e il loro numero
- L'equivalenza si verifica per enumerazione delle assegnazioni

Esercizio 1: implicazioni dirette

Dieci progetti. Si modellino come vincoli lineari:

1. se si sceglie il progetto 2 allora si sceglie il 3;
2. se si sceglie il 2 allora non si sceglie il 4;
3. se si scelgono l'1 e il 6 allora si sceglie il 7;
4. se si sceglie l'1 oppure il 6 allora si sceglie l'8;
5. se si scelgono il 2 e il 3 allora non si sceglie il 9;
6. se si sceglie il 2 oppure il 3 allora non si sceglie il 10.

Esercizio 2: antecedenti e conseguenti negati

Dieci progetti. Si modellino:

1. se non si sceglie il 3 allora si sceglie il 2;
2. se non si sceglie il 4 allora non si sceglie il 2;
3. se si sceglie il 7 allora si scelgono l'1 e il 6;
4. se si sceglie l'8 allora si sceglie l'1 oppure il 6;
5. se non si sceglie il 9 allora si scelgono il 2 e il 3;
6. se non si sceglie il 10 allora si sceglie il 2 oppure il 3.

Esercizio 3: antecedenti e conseguenti composti (1/2)

Otto progetti. Si modellino:

1. x_7 `OR` $x_3 \Rightarrow x_1$ `AND` x_2;
2. x_1 `AND` x_6 `AND` $x_7 \Rightarrow x_8$;
3. x_5 `AND` x_2 `AND` `NOT` $x_4 \Rightarrow$ `NOT` x_3;

Esercizio 3: antecedenti e conseguenti composti (2/2)

4. $(x_1 \texttt{ OR } x_4) \texttt{ AND } x_6 \Rightarrow x_2 \texttt{ AND } (x_5 \texttt{ OR } x_7)$;
5. $(x_2 \texttt{ OR } x_5) \texttt{ AND NOT } x_8 \Rightarrow x_3 \texttt{ OR NOT } x_6$;
6. $(x_1 \texttt{ OR } x_4) \texttt{ AND } (x_2 \texttt{ OR } x_5) \texttt{ AND NOT } x_8 \Rightarrow x_3 \texttt{ AND } (\texttt{NOT } x_6 \texttt{ OR } x_7)$.

Esercizio 4: «almeno due fra»

Nove progetti. Si modellino:

1. se si sceglie il 4 allora almeno due fra 1, 2, 3;
2. se almeno due fra 6, 7, 8 allora si sceglie il 5;
3. se non si sceglie il 4 allora almeno due fra 1, 2, 3, 9;
4. se si sceglie l'8 allora (1 e 6) oppure (1 e 7) oppure (2 e 6);
5. se almeno due fra 1, 3, 5 allora non si sceglie il 9;
6. se (1 e 2) oppure (3 e 4) allora si sceglie il 5.

«Almeno due» si scrive anche contando

Osservazione 4: «Almeno due» si scrive anche contando

La condizione «almeno due fra 1, 2, 3» è anche $x_1+x_2+x_3 \geq 2$. *Come conseguente dell'implicazione governata da* x_4 *diventa*

$$x_1 + x_2 + x_3 \geq 2x_4 :$$

un vincolo solo, contro le tre clausole del punto 1. I due sistemi hanno le stesse 16 *soluzioni binarie — si verifica per enumerazione — ma non lo stesso rilassamento.*

La forma contata è più forte (1/2)

- Questa volta è la forma contata a essere **più forte**
- Sommando le tre clausole $x_i + x_j \geq x_4$ si ottiene solo

$$x_1 + x_2 + x_3 \geq \frac{3}{2} x_4,$$

  più debole di $\geq 2x_4$
- Viceversa, con $0 \leq x_i \leq 1$, la forma contata implica ciascuna clausola

La forma contata è più forte (2/2)

- Sul problema

$$\begin{aligned} \max \quad & x_1 + x_2 + x_3 + 3x_4 \\ \text{con} \quad & x_1 + x_2 + x_3 + 2x_4 \leq 3 \end{aligned}$$

  il cui ottimo intero è $z(MILP) = 3$
- le tre clausole danno $z(LP^+) = 27/7 \approx 3{,}86$
- il vincolo contato dà $z(LP^+) = 15/4 = 3{,}75$: più vicino all'ottimo intero

La morale

- La morale non è «contare è sempre meglio»
- «Al più uno» contato era più forte delle clausole a coppie, qui «almeno due» contato è di nuovo più forte
- Ma non sempre va così: una somma aggregata può benissimo essere più **debole** delle disuguaglianze che riassume
- Due formulazioni con lo stesso insieme intero vanno confrontate sul rilassamento, **risolvendolo**, e non a occhio

Esercizio 5: scissioni

Dieci progetti. Si modellino:

1. x_1 OR $x_2 \Rightarrow x_3$;
2. $x_4 \Rightarrow x_5$ AND x_6;
3. x_1 OR $x_2 \Rightarrow x_3$ AND x_4;
4. x_1 AND $x_2 \Rightarrow x_3$;
5. x_5 OR $x_6 \Rightarrow$ NOT x_7;
6. NOT x_8 OR NOT $x_9 \Rightarrow x_{10}$.

Riepilogo

Gli esercizi:

- Una binaria per progetto; ogni condizione: CNF, vincoli, conteggio
- Implicazioni dirette, negate, composte, «almeno due fra», scissioni

«Almeno due»:

- Tre clausole oppure un vincolo contato $x_1 + x_2 + x_3 \geq 2x_4$
- Stesse soluzioni intere, rilassamento diverso: si confronta risolvendo

I vincoli logici dentro un modello di ottimizzazione

Introduzione: I vincoli logici dentro un modello

- I vincoli logici raramente vivono da soli: si aggiungono a un modello che **ottimizza** qualcosa
- Usiamo i 10 progetti dell'Esercizio 1, con ricavi, costi e un budget
- Vediamo il modello, l'istanza, la soluzione e il codice `gurobipy`

I dati

- $n = 10$ progetti dell'Esercizio 1, di ricavo r_p e costo c_p:

progetto p	1	2	3	4	5	6	7	8	9	10
ricavo r_p	9	7	4	8	3	6	2	5	7	6
costo c_p	4	3	2	4	2	3	1	3	4	3

- Budget $b = 14$

Selezione di progetti con vincoli logici

Selezione di progetti con vincoli logici

$$\max \quad \sum_{p=1}^{n} r_p \, x_p$$

$$\text{soggetto a} \quad \sum_{p=1}^{n} c_p \, x_p \leq b,$$

le clausole delle sei implicazioni,

$$x_p \in \{0,1\}, \quad \forall p \in \{1,2,\ldots,n\}.$$

Descrizione

- L'**obiettivo** somma i ricavi dei progetti scelti
- Il primo vincolo, uno solo, è il **budget**
- Le sei implicazioni, portate in CNF, danno **otto clausole**, e ogni clausola è una disuguaglianza: non aggiungono variabili, solo righe
- I **domini** sono le n condizioni di binarietà, una per progetto

Il modello dell'istanza

Selezione di progetti, dieci binarie

$$
\begin{aligned}
\max\ & 9x_1+7x_2+4x_3+8x_4+3x_5+6x_6+2x_7+5x_8+7x_9+6x_{10} \\
\text{soggetto a}\ & 4x_1+3x_2+2x_3+4x_4+2x_5+3x_6+x_7+3x_8+4x_9+3x_{10} \le 14 \\
& -x_2+x_3 \ge 0 \\
& x_2+x_4 \le 1 \\
& x_1+x_6-x_7 \le 1 \\
& -x_1+x_8 \ge 0 \\
& -x_6+x_8 \ge 0 \\
& x_2+x_3+x_9 \le 2 \\
& x_2+x_{10} \le 1 \\
& x_3+x_{10} \le 1 \\
& x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \in \{0,1\}
\end{aligned}
$$

La soluzione

- Senza i vincoli logici l'ottimo è 30
- Con i vincoli logici l'ottimo scende a $z(MILP) = 28$
- Si scelgono i progetti $1, 2, 3, 5, 8$, di costo complessivo 14: il budget è saturo
- Il rilassamento vale $z(LP^+) = 29$

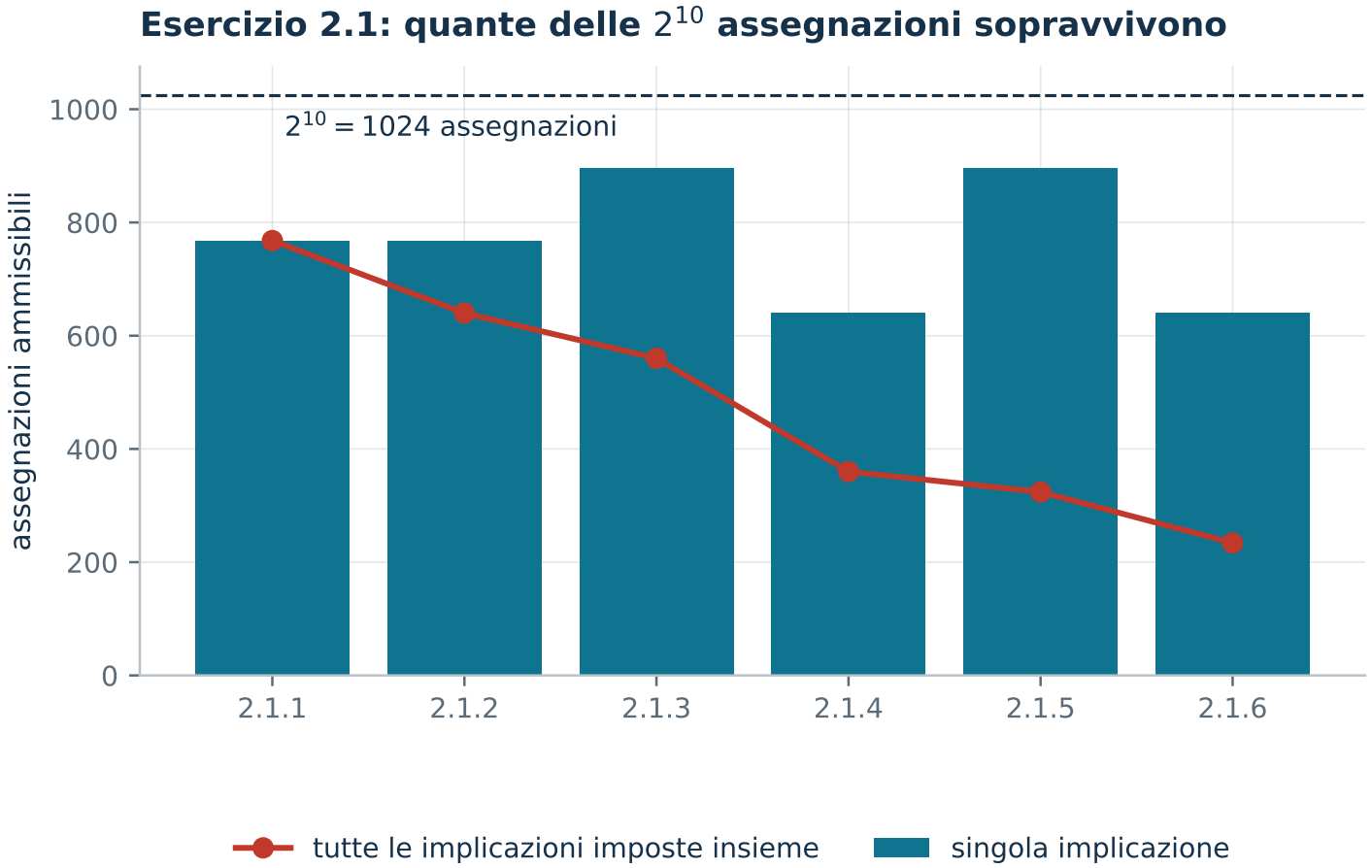
Le sei implicazioni sono soddisfatte:

- si sceglie il 2 e infatti anche il 3; non si sceglie il 4; non si scelgono insieme 1 e 6
- si sceglie l'1 e infatti l'8; si scelgono 2 e 3 e infatti non il 9; si sceglie il 2 e infatti non il 10

Quanto tagliano sei implicazioni

- Le $2^{10} = 1024$ assegnazioni delle dieci binarie si riducono a 234 quando si impongono tutte e sei le implicazioni: meno di un quarto
- Nessuna implicazione, da sola, taglia più della metà dello spazio; insieme lo riducono di tre quarti
- È il motivo per cui un modello si scrive un vincolo alla volta, ma si valuta tutto intero

Le assegnazioni che sopravvivono



- Barre: ciascuna implicazione presa da sola; linea: tutte quelle imposte fino a quel punto

Il modello in gurobipy

- Accanto a `gurobipy` usiamo il piccolo modulo didattico `booleane.py`, distribuito con il materiale del corso
- `V` crea una variabile; `AND`, `OR`, `NOT` e `IMP` la compongono: si scrive la formula come a mano
- `cnf` la converte in forma normale congiuntiva
- `vincolo` traduce una clausola nei coefficienti, nel verso e nel termine noto della disuguaglianza

Il codice (1/3): le implicazioni

```
from booleane import cnf, vincolo
from booleane import IMP, AND, OR, NOT, V

x = {p: V(f"x{p}") for p in range(1, 11)}
implicazioni = [
    IMP(x[2], x[3]),
    IMP(x[2], NOT(x[4])),
    IMP(AND(x[1], x[6]), x[7]),
    IMP(OR(x[1], x[6]), x[8]),
    IMP(AND(x[2], x[3]), NOT(x[9])),
    IMP(OR(x[2], x[3]), NOT(x[10]))]
```

Il codice (2/3): variabili, obiettivo, budget

```
import gurobipy as gp
from gurobipy import GRB

P = range(1, 11)
m = gp.Model("selezione_progetti")
xv = m.addVars(P, vtype=GRB.BINARY, name="x")
m.setObjective(gp.quicksum(r[p]*xv[p] for p in P),
               GRB.MAXIMIZE)
m.addConstr(gp.quicksum(c[p]*xv[p] for p in P) <= b)
```

Il codice (3/3): una clausola, un vincolo

```
# una clausola, un vincolo
for formula in implicazioni:
    for clausola in cnf(formula):
        coef, verso, rhs = vincolo(clausola)
        lhs = gp.quicksum(k*xv[int(n[1:])]
                          for n, k in coef.items())
        m.addConstr(lhs <= rhs if verso == "<="
                        else lhs >= rhs)
m.optimize()
```

- Tutto quello che fa il codice è il procedimento di questo capitolo, scritto una volta sola
- I due bound — quello primale, dall'euristica, e quello duale, dal rilassamento — riproducono in forma didattica i due lati del **sandwich** che il solver cerca progressivamente di chiudere

Riepilogo

Il modello:

- Selezione di progetti: obiettivo, budget, otto clausole, binarietà
- I vincoli logici aggiungono righe, non variabili

I risultati:

- Ottimo da 30 a $z(MILP) = 28$ (progetti $1, 2, 3, 5, 8$), $z(LP^+) = 29$
- Da 1024 a 234 assegnazioni: insieme, le implicazioni tagliano tre quarti

Quello che questo capitolo ha messo in mano

Introduzione: Quello che il capitolo ha messo in mano

- Raccogliamo la ricetta in poche righe
- E diamo un nome alle tre forme che ricorrono più spesso nei modelli

La ricetta, sempre la stessa

- Una binaria a ogni fatto elementare
- Si porta la condizione in CNF — una congiunzione di OR
- Ogni clausola diventa una disuguaglianza lineare

- L' `AND` non costa nulla: i vincoli sono già in AND fra loro
- L' `OR` diventa una somma ≥ 1; il `NOT` diventa $1 - x$
- L'implicazione $x \Rightarrow y$ diventa $x \leq y$
- Il «se e solo se» chiede tutte e due le disuguaglianze

Tre forme con un nome

- **Set covering**: $\sum_{j \in S} x_j \geq 1$ (almeno uno)
- **Set packing**: $\sum_{j \in S} x_j \leq 1$ (al più uno)
- **Set partitioning**: $\sum_{j \in S} x_j = 1$ (esattamente uno)

Riconoscerle in un enunciato è metà del lavoro di modellazione.

Riepilogo (1/2)

Linguaggio:

- Proposizioni $\leftrightarrow$ binarie; letterali, clausole, CNF
- Distributive, De Morgan, assorbimento per portare in CNF

Traduzione:

- Clausola $\rightarrow$ vincolo ≥ 1, `OR` $\rightarrow +$, `NOT` $x \rightarrow 1 - x$
- Implicazione: `NOT` (antecedente) `OR` (conseguente)
- Scissioni valide solo per antecedente `OR` e conseguente `AND`

Riepilogo (2/2)

Contare:

- Set covering, packing, partitioning: già lineari
- Formulazioni con lo stesso insieme intero si confrontano sul rilassamento

Verificare:

- Enumerazione delle 2^n assegnazioni: la dimostrazione per casi
- Nei modelli i vincoli logici aggiungono righe, e insieme tagliano molto