Legami fra variabili

Indice (1/2)

Come si legge il capitolo

Attivazione: forma aggregata e disaggregata

Costo fisso, capacità e flusso continuo

Lotto minimo e variabile semicontinua

Conteggi interi e arrotondamento all'intero superiore

L'ausiliaria di massimo

Min-max, max-min e differenza

Il valore assoluto

Indice (2/2)

Big-M: vincoli condizionati e disgiunzioni

Precedenze e sequenziamento

«Se e solo se»

Contare i tipi diversi

Alldiff ed espansione binaria

Vincoli violabili, deviazioni e penalità

Funzioni lineari a tratti

La mappa delle tecniche

Obiettivi (1/2)

Alla fine di questo capitolo sarai in grado di:

- legare variabili binarie, intere e continue con le quattordici tecniche del capitolo;
- dimostrare quale verso di un legame è imposto dai vincoli e quale segue dall'ottimalità;
- scegliere dai dati il big-M valido più piccolo;

Obiettivi (2/2)

Alla fine di questo capitolo sarai in grado di:

- confrontare forma aggregata e disaggregata con la forza del rilassamento;
- usare la mappa delle tecniche per modellare un legame nuovo.

Prerequisiti

- Da questo modulo, Capitolo 5 «Logica e variabili binarie»: i vincoli lineari delle implicazioni fra variabili binarie e le loro dimostrazioni
- Da questo modulo, Capitolo 2 «Rilassamenti, dualità e bound»: il confronto fra formulazioni attraverso i loro rilassamenti LP
- Dal modulo *Programmazione lineare intera*, Capitolo 3 «Formulazioni ideali»: le formulazioni più forti e l'involucro convesso dei punti interi

Come si legge il capitolo

Introduzione: legami fra variabili

- Il capitolo precedente collega variabili *tutte binarie*
- Qui si collegano famiglie *diverse*: binarie con continue, binarie con intere, continue fra loro
- Sono **quattordici tecniche**: il vero contenuto della modellazione MIP
- Quasi ogni modello di questo corso è una combinazione di queste

Come si legge ogni sezione

Ogni tecnica è presentata in cinque passi:

(a) il **legame in parole**: che cosa si vuole dire

(b) i **vincoli lineari** che lo dicono, con il conteggio

(c) la **dimostrazione**: nei due versi, con un argomento di scambio, o con un *controesempio*

(d) la **forza del rilassamento**, su un'istanza minima

(e) la **riga** `gurobipy` che la scrive nel solver

Il passo (c) distingue un modello corretto da uno plausibile.
Il passo (d) distingue un modello che si risolve da uno che non si risolve.

Due proprietà che non vanno confuse

Osservazione 1: Due proprietà che non vanno confuse

- *Proprietà **imposta dai vincoli**: vale per ogni soluzione ammissibile, ottima o no. Si dimostra guardando solo i vincoli*
- *Proprietà **di ottimalità**: vale solo nelle soluzioni ottime. Si dimostra con un argomento di scambio*

L'argomento di scambio

- Si prende una soluzione ammissibile che viola la proprietà
- Si costruisce una soluzione modificata
- Si verifica che resta ammissibile e si confronta il valore

- Se il valore migliora *strettamente*: «in ogni ottimo»
- Se non peggiora: «esiste un ottimo», che è più debole

Scrivere «in ogni ottimo» quando il coefficiente è solo non negativo è l'errore più comune di questo capitolo.

Riepilogo

- Ogni tecnica: parole, vincoli, dimostrazione, rilassamento, `gurobipy`
- **Imposto dai vincoli**: vale in ogni soluzione ammissibile
- **Di ottimalità**: si dimostra con l'argomento di scambio; miglioramento stretto $\Rightarrow$ «in ogni ottimo», altrimenti solo «esiste un ottimo»

Attivazione: forma aggregata e disaggregata

Introduzione: attivazione

- Una risorsa (una macchina, una sede, un impianto) va *attivata* prima di essere usata
- Le binarie x_{ij} dicono «l'oggetto i usa la risorsa j»
- Le binarie y_j dicono «la risorsa j è attivata»
- Si vuole: **se qualcuno usa j, allora j è attivata**

I vincoli

Due modi, con n oggetti e m risorse:

Disaggregato ($n\,m$ vincoli):

$$x_{ij} \leq y_j, \qquad \forall i \in \{1, 2, \ldots, n\},\ \forall j \in \{1, 2, \ldots, m\}$$

Aggregato (m vincoli):

$$\sum_{i=1}^{n} x_{ij} \leq n\, y_j, \qquad \forall j \in \{1, 2, \ldots, m\}$$

- Con una capacità k_j (al più k_j oggetti): $\sum_i x_{ij} \leq k_j y_j$
- Allora impone *anche* la capacità: due condizioni in un vincolo solo

La dimostrazione

Su punti binari le due forme sono **equivalenti**.

- *Disaggregato* $\Rightarrow$ *aggregato*: sommando $x_{ij} \leq y_j$ su i si ottiene $\sum_i x_{ij} \leq n y_j$
- *Aggregato* $\Rightarrow$ *disaggregato, su punti binari*:
  - se $y_j = 0$: $\sum_i x_{ij} \leq 0$, quindi $x_{ij} = 0 \leq y_j$ per ogni i
  - se $y_j = 1$: $x_{ij} \leq 1 = y_j$ perché le x sono binarie

Il secondo verso usa la binarietà: *fuori* dai punti binari non vale, ed è questo che rende le due forme diverse nel rilassamento.

Il verso opposto non è imposto

- «Se j è attivata allora qualcuno la usa»: $y_j = 1 \Rightarrow \sum_i x_{ij} \geq 1$
- **Non** segue dai vincoli: $y_j = 1$ con tutte le $x_{ij} = 0$ è ammissibile in entrambe le forme
- Segue dall'ottimalità se il costo di attivazione f_j è *strettamente* positivo:
  - ottimo con $y_j = 1$ e nessun $x_{ij} = 1$
  - porre $y_j = 0$ lascia i vincoli soddisfatti e riduce il costo di $f_j > 0$: contraddizione
  - quindi «in ogni ottimo»
- Se $f_j = 0$ è ammesso: solo «esiste un ottimo in cui $y_j = 0$»

La forza del rilassamento

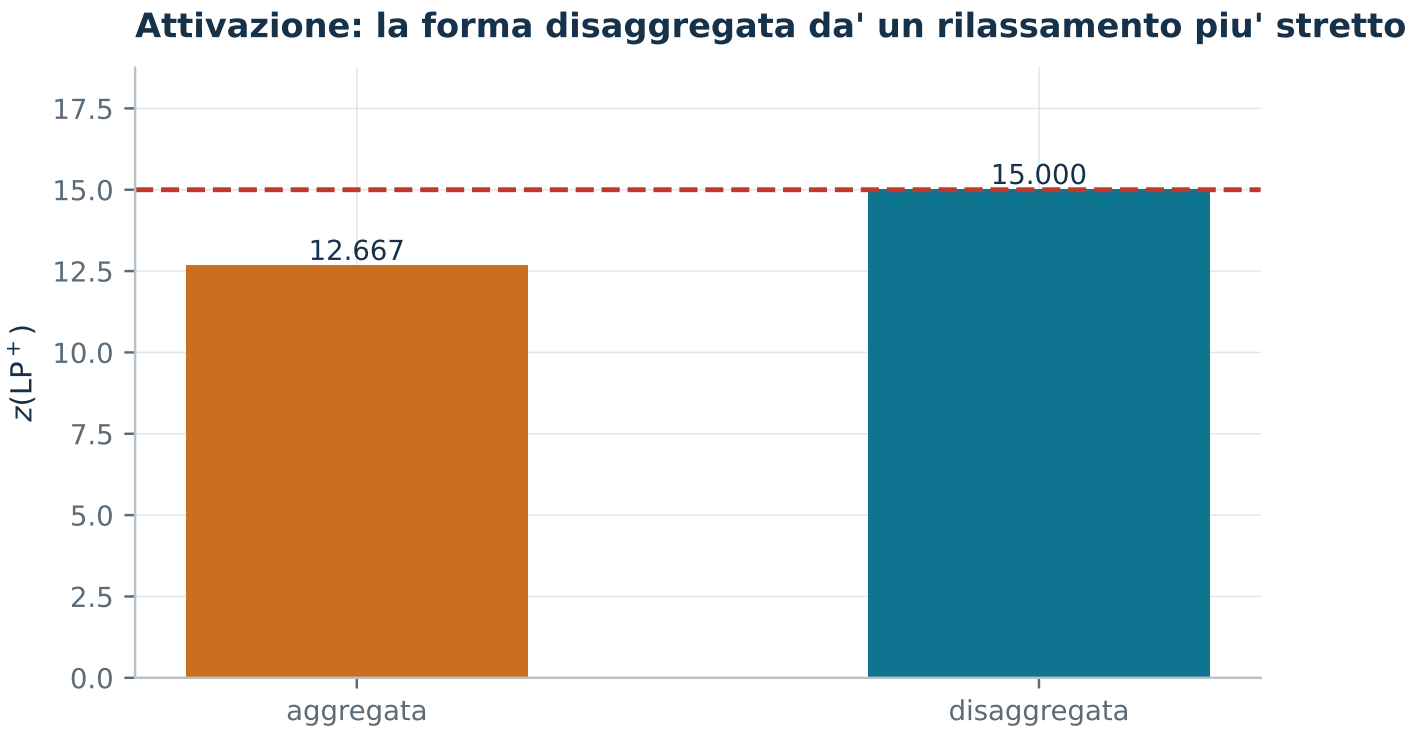
Istanza: $n = 3$ clienti, $m = 2$ sedi, costi di attivazione $f = (8, 6)$, costi di servizio

$$c = \begin{pmatrix} 2 & 5 \\ 4 & 1 \\ 3 & 3 \end{pmatrix}$$

Formulazione	vincoli di link	$z(LP^+)$
aggregata	$m = 2$	$38/3 \approx 12{,}67$
disaggregata	$n\,m = 6$	15

L'ottimo intero è $z(MILP) = 15$.

Il rilassamento delle due forme



La linea rossa è l'ottimo intero.

Più righe, rilassamento più stretto

- La forma disaggregata raggiunge l'ottimo intero già nel rilassamento
- Quella aggregata si ferma a 38/3
- È il compromesso tipico di questa tecnica
- Su istanze grandi la scelta non è ovvia ($n\,m$ righe possono essere molte), ma su istanze difficili la forma disaggregata quasi sempre paga

In gurobipy

```
# disaggregato
m.addConstrs((x[i, j] <= y[j]
              for i in range(n) for j in range(mm)),
             name="link")
# aggregato
m.addConstrs((x.sum("*", j) <= n * y[j]
              for j in range(mm)), name="link")
```

Riepilogo

- Attivazione: disaggregata $x_{ij} \leq y_j$ ($n\,m$ vincoli) oppure aggregata $\sum_i x_{ij} \leq n y_j$ (m vincoli)
- Equivalenti sui punti binari, diverse nel rilassamento: la disaggregata è più stretta
- «Attivata $\Rightarrow$ usata» non è imposto: segue dall'ottimalità solo se $f_j > 0$

Costo fisso, capacità e flusso continuo

Introduzione: costo fisso

- Come l'attivazione, ma la quantità usata è *continua*
- $q_j \geq 0$ è quanto si produce nell'impianto j
- y_j dice se l'impianto è aperto
- Un impianto chiuso non produce; uno aperto produce al più la sua capacità c_j

I vincoli e la dimostrazione

$$q_j \leq c_j\, y_j, \qquad \forall j \in \{1, 2, \ldots, m\} \qquad (m \text{ vincoli})$$

Un solo vincolo per impianto, con il coefficiente *giusto*: la capacità, non un numero grande a caso.

- $y_j = 0$: $q_j \leq 0$ e, con $q_j \geq 0$, $q_j = 0$. «Chiuso $\Rightarrow$ non produce» è imposto, e con esso «produce $\Rightarrow$ aperto»
- $y_j = 1$: $q_j \leq c_j$, la capacità
- «Aperto $\Rightarrow$ produce» non è imposto: segue dall'ottimalità solo se $f_j > 0$

La forza del rilassamento

Istanza: due impianti, costi fissi $f = (10, 14)$, costi unitari $c = (3, 2)$, capacità $C = (6, 7)$, domanda $D = 9$.
L'ottimo è $z(MILP) = 44$ (entrambi aperti, $q = (2, 7)$).

Coefficiente nel vincolo	$z(LP^+)$
la capacità, c_j	$112/3 \approx 37{,}33$
big-M $= 100$ (più $q_j \leq c_j$ a parte)	$1059/50 = 21{,}18$

Stesso insieme intero, stesso ottimo, rilassamenti lontanissimi.

La regola del coefficiente

Il coefficiente di una binaria di attivazione è il più piccolo valore che la variabile continua può assumere quando l'attivazione vale 1, e va ricavato dai dati.

In gurobipy

```
m.addConstrs((q[j] <= c[j] * y[j]
              for j in range(mm)), name="link")
```

Riepilogo

- Costo fisso: $q_j \leq c_j y_j$, un vincolo per impianto
- «Chiuso $\Rightarrow$ non produce» è imposto; «aperto $\Rightarrow$ produce» solo per ottimalità con $f_j > 0$
- Il coefficiente è la capacità, ricavata dai dati: un big-M grande dà lo stesso ottimo ma un rilassamento molto più debole

Lotto minimo e variabile semicontinua

Introduzione: lotto minimo

- «Se si produce, si produce almeno ℓ»
- La quantità q_j vale zero oppure sta fra una soglia ℓ e la capacità c_j
- Non è un intervallo: è l'unione di un punto e di un intervallo
- Una variabile con questo dominio si chiama **semicontinua**; se è anche intera (pezzi, unità, persone) si chiama **semi-intera**, e i vincoli sono gli stessi

I vincoli e la dimostrazione

Con la stessa binaria y_j della sezione precedente:

$$\ell\, y_j \leq q_j, \qquad \forall j \in \{1, 2, \ldots, m\} \qquad (m \text{ vincoli})$$

$$q_j \leq c_j\, y_j, \quad \forall j \in \{1, 2, \ldots, m\} \qquad (m \text{ vincoli})$$

Entrambi i versi sono imposti dai vincoli:

- $y_j = 0$: $0 \leq q_j \leq 0$, cioè $q_j = 0$
- $y_j = 1$: $\ell \leq q_j \leq c_j$

$$q_j \in \{0\} \cup [\ell,\ c_j]$$

Serve $\ell \leq c_j$

- Se $\ell > c_j$, $y_j = 1$ è inammissibile e la variabile è costretta a zero
- È un errore di dati
- Il solver lo segnala come inammissibilità solo se y_j è forzata a 1 da altri vincoli

La forza del rilassamento

- Stessa istanza della sezione precedente, con $\ell = 5$
- L'ottimo passa da 44 a $z(MILP) = 49$, con $q = (5, 5)$ invece di $(2, 7)$
- La soglia costringe a produrre 5 nel secondo impianto anche se il primo è più caro
- Ma $z(LP^+)$ resta $112/3$, identico al caso senza soglia

Perché il lotto minimo non si vede nel rilassamento

Osservazione 2: Perché il lotto minimo non si vede nel rilassamento

Nel rilassamento y_j è libera in $[0, 1]$, e il vincolo $q_j \geq \ell y_j$ si soddisfa abbassando y_j: basta $y_j \leq q_j/\ell$. Il vincolo di soglia non morde mai sul continuo. Tutto il suo effetto si scarica sull'interezza: per questo i modelli con lotto minimo sono tipicamente più difficili di quelli senza, a parità di dimensione.

In gurobipy

```
m.addConstrs((q[j] >= ell * y[j]
              for j in range(mm)), name="lotto")
m.addConstrs((q[j] <= c[j] * y[j]
              for j in range(mm)), name="capacita")
```

- Gurobi ha anche il tipo `GRB.SEMICONT`, che dichiara direttamente il dominio
- In questo corso si scrive la formulazione a mano, perché è quella che si deve saper dimostrare

Riepilogo

- Lotto minimo: $\ell y_j \leq q_j \leq c_j y_j$, cioè $q_j \in \{0\} \cup [\ell, c_j]$
- Entrambi i versi imposti dai vincoli; serve $\ell \leq c_j$
- La soglia non morde nel rilassamento: tutto il suo effetto è sull'interezza

Conteggi interi, capacità multipla, arrotondamento all'intero superiore

Introduzione: conteggi interi

- «Quanti contenitori servono?»
- La variabile non è binaria ma *intera*: $w \in \mathbb{Z}_{\geq 0}$
- Conta oggetti indivisibili (scatole, camion, turni, operai)
- Ciascuno porta una capacità c

I vincoli e la dimostrazione

$$\sum_i a_i x_i \leq c\, w \qquad \text{(1 vincolo)}$$

$$w \in \mathbb{Z}_{\geq 0} \qquad \text{(1 variabile intera)}$$

- Posto $q = \sum_i a_i x_i$: il vincolo impone $w \geq q/c$, l'interezza impone $w \geq \lceil q/c \rceil$
- Con un obiettivo che minimizza (o paga) w, in ogni ottimo w vale esattamente quel tetto: se valesse di più, abbassarlo di 1 resterebbe ammissibile e ridurrebbe il costo
- Con costo nullo per w: solo «esiste un ottimo»

Il tetto non si scrive con $\lceil \cdot \rceil$

Osservazione 3: Il tetto non si scrive con $\lceil \cdot \rceil$

$\lceil t \rceil$ non è una funzione lineare: non si può scrivere dentro un vincolo. La coppia «disuguaglianza $\leq$ cw + dichiarazione w intera» la realizza implicitamente, e va letta così quando si spiega il modello.

La forza del rilassamento

- 17 pezzi unitari, capienza $c = 5$
- Il rilassamento dà $w \geq 17/5 = 3{,}4$
- L'ottimo intero è $z(MILP) = 4$
- Il gap $4 - 17/5 = 3/5$ viene tutto dall'interezza: nessun taglio lineare sulle sole x lo chiude, serve una disuguaglianza che usi w intera

In gurobipy

```
w = m.addVar(vtype=GRB.INTEGER, name="w")
m.addConstr(gp.quicksum(a[i] * x[i]
                        for i in range(n)) <= c * w,
            name="capienza")
```

Riepilogo

- Conteggio intero: $\sum_i a_i x_i \leq c\, w$ con w intera
- L'interezza realizza implicitamente il tetto $\lceil q/c \rceil$; l'uguaglianza vale all'ottimo se w costa
- Il gap viene tutto dall'interezza di w

L'ausiliaria di massimo

Introduzione: l'ausiliaria di massimo

- Una variabile z deve valere il *massimo* di certe quantità
- Esempi: il tempo dell'attività più lunga, il costo di connessione più alto, il carico della macchina più carica
- Vedremo che metà del legame è imposta dai vincoli e metà viene dall'ottimalità

I vincoli

Se le quantità sono $t_j x_j$ con x_j binaria:

$$z \geq t_j\, x_j, \quad \forall j \in \{1, 2, \ldots, n\} \qquad (n \text{ vincoli})$$

$$z \geq 0$$

La dimostrazione, in tre passi

1. *Imposto dal vincolo*: per ogni j con $x_j = 1$ si ha $z \geq t_j$, quindi $z \geq \max_{j:x_j=1} t_j$, in ogni soluzione ammissibile
2. *Dall'ottimalità*: se z è nell'obiettivo con coefficiente strettamente positivo in un minimo (o strettamente negativo in un massimo) e in nessun altro vincolo, abbassare z fino al massimo lascia i vincoli soddisfatti e migliora strettamente l'obiettivo
3. *Sintesi*: in ogni ottimo $z = \max_{j:x_j=1} t_j$ (il massimo sull'insieme vuoto è 0, garantito da $z \geq 0$)

Il passo 2 cade se z è usata altrove

Osservazione 4: Il passo 2 cade se z è usata altrove

Se z compare in un altro vincolo che la vuole grande (per esempio $z \geq w$, con w un'altra variabile) l'argomento di scambio non funziona più: abbassare z può violare quel vincolo. In quel caso $z \geq \max$ resta vero, ma $z = \max$ no. Se serve il massimo di due famiglie diverse, servono due ausiliarie.

La forza del rilassamento

- $\min z$ con $z \geq t_j x_j$, $t = (4, 7, 3)$ e $\sum_j x_j \geq 2$
- Ottimo intero $z(MILP) = 4$: si scelgono i lavori 1 e 3
- Il rilassamento vale $168/61 \approx 2{,}75$: la soluzione frazionaria $x_j = z/t_j$ spalma la scelta su tutti e tre i lavori e abbassa il massimo
- Il legame di massimo dà rilassamenti *deboli*: una delle ragioni per cui i problemi di makespan sono difficili

In gurobipy

```
z = m.addVar(name="z")
m.addConstrs((z >= t[j] * x[j] for j in range(n)),
             name="massimo")
```

Riepilogo

- Ausiliaria di massimo: $z \geq t_j x_j$, $z \geq 0$
- $z \geq \max$ è imposto; $z = \max$ vale solo all'ottimo, se z ha il segno giusto nell'obiettivo e non compare altrove
- Rilassamenti deboli: i problemi di makespan sono difficili

Min-max, max-min e differenza fra il massimo e il minimo

Introduzione: obiettivi di equità

Tre obiettivi di equità, spesso confusi:

- **min-max**: minimizzare il carico più alto
- **max-min**: massimizzare il carico più basso
- **differenza**: minimizzare lo scarto fra il più alto e il più basso

I vincoli

Con ℓ_k il carico della risorsa $k \in \{1, 2, \ldots, m\}$:

min-max (m vincoli):

$$\min z \quad \text{con} \quad z \geq \ell_k, \ \forall k \in \{1, 2, \ldots, m\}$$

max-min (m vincoli):

$$\max u \quad \text{con} \quad u \leq \ell_k, \ \forall k \in \{1, 2, \ldots, m\}$$

differenza ($2m$ vincoli):

$$\min (z - u) \quad \text{con entrambi}$$

La dimostrazione

- Ciascuna è l'ausiliaria di massimo (o di minimo) della sezione precedente
- L'argomento di scambio va nel verso giusto:
  - in un $\min z$ la z scende al massimo dei carichi
  - in un $\max u$ la u sale al minimo dei carichi
- Nella differenza entrambe le spinte sono presenti contemporaneamente e le due conclusioni valgono insieme

I tre obiettivi non si confrontano

Osservazione 5: I tre obiettivi non si confrontano

Cinque pesi $p = (3, 5, 2, 4, 7)$ *su due operai (totale* 21*): le tre versioni scelgono la stessa ripartizione* $(11, 10)$*, la migliore possibile perché il totale è dispari, ma i valori ottimi sono* 11*,* 10 *e* 1*. Confrontare «*$z = 11$*» di un min-max con «*$z = 1$*» di una differenza non significa niente.*

- Prima di confrontare due modelli bisogna che ottimizzino la stessa cosa
- Con più di due risorse, min-max e max-min in generale scelgono ripartizioni diverse

La forza del rilassamento e gurobipy

- Il min-max sull'istanza dei cinque pesi dà $z(LP^+) = 21/2 = 10{,}5$ contro $z(MILP) = 11$
- Il rilassamento distribuisce i pesi a metà esatta, cosa che l'interezza non permette

```
z = m.addVar(name="z")
m.addConstrs((z >= carico[k]
              for k in range(len(carico))),
             name="max")
m.setObjective(z, GRB.MINIMIZE)
```

Riepilogo

- Min-max: $\min z$, $z \geq \ell_k$; max-min: $\max u$, $u \leq \ell_k$; differenza: $\min(z - u)$ con entrambi
- Sono l'ausiliaria di massimo/minimo con lo scambio nel verso giusto
- I valori ottimi dei tre obiettivi non si confrontano fra loro

Il valore assoluto

Introduzione: il valore assoluto

- Si vuole $|u - v|$: uno scarto, un errore, uno sbilanciamento
- Il caso in obiettivo e il caso in vincolo si comportano in modo radicalmente diverso
- Vedremo tre casi: in obiettivo, come vincolo $\leq$, come vincolo $\geq$

I vincoli (1/2): in obiettivo e come vincolo $\leq$

In obiettivo, minimizzando: una variabile $d \geq 0$ e due vincoli

$$d \geq u - v \qquad \text{(1 vincolo)}$$

$$d \geq v - u \qquad \text{(1 vincolo)}$$

con d nell'obiettivo da minimizzare. Nessuna binaria.

Come vincolo $\leq$: $|u - v| \leq k$ è semplicemente

$$u - v \leq k \quad \text{e} \quad v - u \leq k \qquad \text{(2 vincoli, nessuna binaria)}$$

I vincoli (2/2): come vincolo $\geq$

- $|u - v| \geq k$ **non** si scrive senza binarie
- È la disgiunzione «$u - v \geq k$ *oppure* $v - u \geq k$»
- Richiede una binaria b e un big-M:

$$u - v \geq k - M(1 - b) \qquad \text{(1 vincolo)}$$

$$v - u \geq k - M\,b \qquad \text{(1 vincolo, 1 binaria)}$$

La dimostrazione

- **In obiettivo**: i due vincoli impongono $d \geq |u - v|$ (uno dei due membri destri è $|u - v|$)
- L'obiettivo minimizza d, che non compare altrove: l'argomento di scambio dell'ausiliaria di massimo la porta all'uguaglianza
- **Come vincolo** $\geq$: $b = 1$ disattiva il secondo vincolo (se $M \geq k + \max(v - u)$) e lascia il primo, e viceversa
- È una disgiunzione, non una congiunzione: senza binaria si imporrebbero entrambi, cioè $0 \geq 2k$, inammissibile per $k > 0$

Il caso $\geq$ non è simmetrico al caso $\leq$

Osservazione 6: Il caso $\geq$ non è simmetrico al caso $\leq$

$|u - v| \leq k$ è l'intersezione di due semipiani: un insieme convesso, che si scrive con due vincoli lineari. $|u - v| \geq k$ è il complemento di una striscia: non è convesso, e nessun sistema di vincoli lineari senza variabili intere può descriverlo. La binaria non è un trucco: è necessaria.

La forza del rilassamento e gurobipy

- Istanza dei cinque pesi: min $|\ell_1 - \ell_2|$ ha $z(MILP) = 1$ e $z(LP^+) = 0$
- Il continuo divide 21 in due metà uguali e azzera lo scarto: il rilassamento di un obiettivo di valore assoluto è tipicamente 0, cioè inutile

```
d = m.addVar(name="d")
m.addConstr(d >= u - v, name="abs_piu")
m.addConstr(d >= v - u, name="abs_meno")
```

La stessa idea torna con i vincoli violabili, in una forma equivalente con due deviazioni.

Riepilogo

- In obiettivo (minimizzando): $d \geq u - v$, $d \geq v - u$, nessuna binaria
- $|u - v| \leq k$: due vincoli lineari; $|u - v| \geq k$: disgiunzione, serve una binaria
- Il rilassamento di un obiettivo di valore assoluto è tipicamente 0

Big-M: vincoli condizionati e disgiunzioni

Introduzione: big-M

- «Se $y = 1$ allora vale il vincolo $a'x \leq b$; se $y = 0$ il vincolo non c'è»
- Un vincolo che si accende e si spegne
- Il punto delicato è la scelta di M: da essa dipendono la correttezza del modello e la forza del rilassamento

Il vincolo e la dimostrazione

$$a'x \leq b + M(1 - y) \qquad (1 \text{ vincolo})$$

- Con $y = 1$: $a'x \leq b$, acceso
- Con $y = 0$: $a'x \leq b + M$, spento *purché* M non tagli nulla:

  $M \geq \max\{a'x : x \text{ ammissibile per gli altri vincoli}\} - b$

Come si sceglie M

Si distinguono tre cose:

- un M **valido**: soddisfa la condizione, quindi il modello ha l'insieme intero giusto
- un M **migliorabile**: valido ma più grande del necessario; stesso insieme intero, rilassamento più debole
- il **minimo** M **dimostrato**: il più piccolo valore per cui si è *dimostrata* la validità, tipicamente calcolando il massimo sui soli bound delle variabili. Non è necessariamente il minimo assoluto, ed è onesto dirlo

Un M non valido non è «un po' diverso»

Osservazione 7: Un M non valido non è «un po' diverso»

Con $a = (3, 4, 5)$, $b = 6$ e x binarie, il massimo di $a'x$ è 12, quindi $M = 12-6 = 6$ è valido. Su $\max\ x_1+x_2+x_3+y$ *l'ottimo è $z(MILP) = 3$ (con $y = 0$ e tutte le x a 1). Con $M = 5$ il vincolo con $y = 0$ resta $3x_1 + 4x_2 + 5x_3 \leq 11$, che esclude $x = (1, 1, 1)$: l'ottimo scende a 2. Il modello non risponde più alla domanda posta.*

La forza del rilassamento

Sulla stessa istanza:

M	$z(LP^+)$
6 (il minimo dimostrato)	3, coincide con $z(MILP)$
20	$37/10 = 3{,}7$
1000	$1997/500 \approx 3{,}994$, quasi 4

- Il degrado è monotono e rapido
- Regola operativa: **calcolare M dai dati, sempre, e scriverlo nel testo subito dopo il modello**

In gurobipy

```
# calcolato dai dati, non scelto a occhio
M = sum(max(a[j], 0) for j in range(n)) - b
m.addConstr(gp.quicksum(a[j] * x[j]
                        for j in range(n))
            <= b + M * (1 - y), name="cond")
```

La stessa idea torna nelle precedenze.

Riepilogo

- Vincolo condizionato: $a'x \leq b + M(1 - y)$
- M valido $\geq \max a'x - b$; troppo piccolo cambia il problema, troppo grande indebolisce il rilassamento
- M si calcola dai dati e si dichiara subito dopo il modello

Precedenze e sequenziamento

Introduzione: precedenze

- Una macchina esegue un lavoro alla volta
- Per ogni coppia di lavori uno dei due precede l'altro
- È una **disgiunzione per coppia**, non una precedenza fissata dai dati
- Si scrive con il big-M della sezione precedente

I vincoli

$\kappa_j \geq 0$ istante di completamento del lavoro j, t_j la sua durata, $s_{ij} \in \{0, 1\}$ vale 1 se j precede i:

$$
\begin{aligned}
s_{ij} + s_{ji} &= 1, && \forall i < j && (n(n-1)/2 \text{ vincoli}) \\
\kappa_i &\geq \kappa_j + t_i - M(1 - s_{ij}), && \forall i \neq j && (n(n-1) \text{ vincoli}) \\
\kappa_j &\geq t_j, && \forall j && (n \text{ vincoli})
\end{aligned}
$$

con $i, j \in \{1, 2, \ldots, n\}$.

La dimostrazione

- $s_{ij} + s_{ji} = 1$: esattamente uno dei due ordini è scelto
- Se $s_{ij} = 1$: $\kappa_i \geq \kappa_j + t_i$, il lavoro i finisce almeno t_i dopo la fine di j, cioè non si sovrappongono
- Se $s_{ij} = 0$: $\kappa_i \geq \kappa_j + t_i - M$, che deve essere sempre soddisfatto

Il minimo M

Poiché $\kappa_i \geq t_i$ e $\kappa_j \leq \sum_h t_h$ in ogni soluzione sensata, basta

$$M = \sum_{h=1}^{n} t_h$$

perché allora

$$\kappa_j + t_i - M \leq \sum_h t_h + t_i - \sum_h t_h = t_i \leq \kappa_i$$

- L'orizzonte temporale $\sum_h t_h$ va *dichiarato*
- Senza un limite superiore sui κ_j, nessun M finito è valido

La forza del rilassamento

- Tre lavori di durata $(3, 2, 4)$ su una macchina, obiettivo makespan
- $z(MILP) = 9 = \sum_h t_h$ (ovvio: una sola macchina), con completamenti $(3, 5, 9)$
- Il rilassamento vale 4: con $s_{ij} = 1/2$ tutti i vincoli di precedenza si spengono a metà e i lavori possono sovrapporsi
- È il rilassamento più debole di tutto il capitolo: spiega perché i modelli di sequenziamento big-M scalano male

In gurobipy

```
M = sum(t)                    # l'orizzonte, dichiarato
for i in range(n):
    for j in range(i):
        m.addConstr(s[i, j] + s[j, i] == 1,
                    name=f"disg{i}{j}")
        m.addConstr(kappa[i] >= kappa[j] + t[i]
                    - M * (1 - s[i, j]),
                    name=f"prec{i}{j}")
        m.addConstr(kappa[j] >= kappa[i] + t[j]
                    - M * (1 - s[j, i]),
                    name=f"prec{j}{i}")
```

Riepilogo

- Sequenziamento: $s_{ij} + s_{ji} = 1$ e $\kappa_i \geq \kappa_j + t_i - M(1 - s_{ij})$
- $M = \sum_h t_h$, l'orizzonte, che va dichiarato
- Rilassamento debolissimo: con $s_{ij} = 1/2$ i lavori si sovrappongono

«Se e solo se»

Introduzione: «se e solo se»

- Una binaria y deve valere 1 *esattamente quando* una condizione è vera
- Per esempio: «tutti i lavori della classe sono eseguiti»
- Non basta un'implicazione: ne servono due

I vincoli e la dimostrazione

Con $x_1, \ldots, x_p$ binarie e la condizione «tutte valgono 1»:

$$y \leq x_j, \qquad \forall j \in \{1, 2, \ldots, p\} \qquad (p \text{ vincoli})$$

$$y \geq \sum_{j=1}^{p} x_j - (p-1) \qquad (1 \text{ vincolo})$$

- Il primo dà $y = 1 \Rightarrow x_j = 1$ per ogni j: se anche un solo x_j è zero, $y = 0$
- Il secondo dà il verso opposto: se tutti gli $x_j = 1$ il membro destro vale 1, quindi $y = 1$; altrimenti è ≤ 0 e non dice nulla
- Insieme: $y = 1 \iff \sum_j x_j = p$ in *ogni* soluzione ammissibile

Quando basta un verso solo

- Se y compare nell'obiettivo di un massimo con un premio $v > 0$ e in nessun altro vincolo, il secondo vincolo si può omettere
- In ogni ottimo: se tutti gli $x_j = 1$ e $y = 0$, alzare y a 1 resta ammissibile per $y \leq x_j$ e aumenta l'obiettivo di $v > 0$
- È un argomento di ottimalità, e vale solo all'ottimo

Con $v = 0$ l'argomento cade (1/2)

Osservazione 8: Con $v = 0$ l'argomento cade, e y smette di significare

Istanza a tre lavori, ricavi $(2, 2, 2)$, capacità 3. Con premio $v = 0$ e solo $y \leq x_j$, l'ottimo può avere $y = 0$ con $x = (1, 1, 1)$: y non è più un indicatore fedele.

Con $v = 0$ l'argomento cade (2/2)

v	vincoli	$z(MILP)$	ottimo	fedele?
9	solo $y \leq x_j$	15	$y = 1, x = (1, 1, 1)$	sì
9	entrambi	15	$y = 1, x = (1, 1, 1)$	sì
0	solo $y \leq x_j$	6	$y = 0, x = (1, 1, 1)$	**no**
0	entrambi	6	$y = 1, x = (1, 1, 1)$	sì

- Con $v = 0$ il valore ottimo è lo stesso nei due modelli
- Se y serve solo a pagare un premio, il verso mancante è inutile; se y compare in *altri* vincoli (per esempio sul numero di classi complete), va scritto

La forza del rilassamento e gurobipy

- $z(LP^+) = 15 = z(MILP)$ su questa istanza: con entrambi i versi il rilassamento è esatto
- Il secondo vincolo è debole nel rilassamento (con $x_j = 1/2$ il membro destro è negativo), ma qui non serve

```
m.addConstrs((y <= x[j] for j in range(p)),
             name="iff_su")
m.addConstr(y >= gp.quicksum(x[j] for j in range(p))
            - (p - 1), name="iff_giu")
```

Riepilogo

- «Se e solo se»: $y \leq x_j$ e $y \geq \sum_j x_j - (p-1)$
- Insieme impongono $y = 1 \iff$ tutte le $x_j = 1$ in ogni soluzione ammissibile
- Un verso si può omettere solo se segue dall'ottimalità ($v > 0$) e y non compare altrove

Contare i tipi diversi

Introduzione: contare i tipi

- «Si devono produrre almeno due tipi diversi»
- «Al più tre alimenti in dieta»
- «Non più di quattro configurazioni»
- Si conta *quanti* elementi sono attivi, non quanto se ne produce

I vincoli

Con $q_j \geq 0$ la quantità e y_j l'indicatore:

$$q_j \leq c_j\, y_j, \quad \forall j \in \{1, 2, \ldots, n\} \qquad (n \text{ vincoli})$$

$$q_j \geq \ell\, y_j, \quad \forall j \in \{1, 2, \ldots, n\} \qquad (n \text{ vincoli})$$

$$\sum_{j=1}^{n} y_j \geq p \qquad (1 \text{ vincolo})$$

La dimostrazione, e perché servono entrambe le soglie

- $q_j \leq c_j y_j$ dà «$y_j = 0 \Rightarrow q_j = 0$»
- $q_j \geq \ell y_j$ dà il verso mancante «$y_j = 1 \Rightarrow q_j \geq \ell$»
- Senza la soglia, $y_j = 1$ con $q_j = 0$ è ammissibile: il conteggio si soddisfa accendendo indicatori vuoti, e «almeno p tipi diversi» non dice più niente

Il conteggio dei tipi funziona solo se ogni tipo acceso produce davvero qualcosa, e la soglia ℓ è ciò che lo garantisce.

La forza del rilassamento e gurobipy

- Tre tipi, ricavi unitari $(4, 3, 5)$, risorsa 12, capacità 10 ciascuno, soglia $\ell = 3$, almeno due tipi
- $z(MILP) = 57$, con $q = (3, 0, 9)$; anche il rilassamento vale 57: nessun gap

```
m.addConstrs((q[j] <= c[j] * y[j] for j in range(n)),
             name="attiva")
m.addConstrs((q[j] >= ell * y[j] for j in range(n)),
             name="lotto")
m.addConstr(y.sum() >= p, name="almeno_p_tipi")
```

Riepilogo

- Contare i tipi: $\ell y_j \leq q_j \leq c_j y_j$ e $\sum_j y_j \geq p$
- Senza la soglia ℓ il conteggio è vuoto: si accendono indicatori senza produrre
- Sull'istanza d'esempio il rilassamento è esatto

Alldiff ed espansione binaria

Introduzione: alldiff ed espansione binaria

Due esigenze diverse ma imparentate:

- dare a n oggetti n valori *tutti distinti* (**alldiff**)
- rappresentare una variabile intera limitata con variabili binarie (**espansione binaria**)

I vincoli

Alldiff, con $p_{iv} = 1$ se l'oggetto i riceve il valore v:

$$\sum_{v=1}^{n} p_{iv} = 1, \quad \forall i \in \{1, 2, \ldots, n\} \qquad (n \text{ vincoli})$$

$$\sum_{i=1}^{n} p_{iv} = 1, \quad \forall v \in \{1, 2, \ldots, n\} \qquad (n \text{ vincoli})$$

Espansione binaria di $v \in \{0, 1, \ldots, 2^p - 1\}$:

$$v = \sum_{k=0}^{p-1} 2^k \, b_k \qquad (1 \text{ vincolo})$$

$$b_k \in \{0, 1\}, \qquad \forall k \in \{0, \ldots, p-1\} \qquad (p \text{ binarie})$$

La dimostrazione

- L'alldiff è un doppio *set partitioning*:
  - il primo gruppo dà «ogni oggetto un valore»
  - il secondo «ogni valore a un solo oggetto»
- Insieme impongono una biiezione, cioè valori tutti distinti
- L'espansione binaria è la rappresentazione in base 2, unica per ogni intero in quell'intervallo: la corrispondenza fra v e $(b_0, \ldots, b_{p-1})$ è biunivoca

L'alldiff ha rilassamento esatto, l'espansione no

- La matrice del doppio partitioning è quella del problema di assegnamento: è **totalmente unimodulare**
- Tutti i vertici del rilassamento sono interi e $z(LP^+) = z(MILP)$ (sull'istanza 3×3 entrambi valgono 7): l'interezza è gratis
- L'espansione binaria non aggiunge forza: $\sum_k 2^k b_k$ con $b_k \in [0, 1]$ copre tutto $[0, 2^p - 1]$, come $0 \leq v \leq 2^p - 1$
- Serve a *riformulare*, non a rafforzare: per esempio quando un'altra parte del modello ha bisogno di indicatori binari

Esempio e gurobipy

- $v \in \{0, \ldots, 7\}$ con $v \geq 5$, min v: il modello dà $v = 5 = 1 + 4$, cioè $(b_0, b_1, b_2) = (1, 0, 1)$

```
m.addConstrs((p.sum(i, "*") == 1 for i in range(n)),
             name="un_valore")
m.addConstrs((p.sum("*", v) == 1 for v in range(n)),
             name="alldiff")
m.addConstr(v == gp.quicksum(2 ** k * b[k]
            for k in range(p)), name="espansione")
```

Riepilogo

- Alldiff: doppio partitioning, $\sum_v p_{iv} = 1$ e $\sum_i p_{iv} = 1$
- Matrice totalmente unimodulare: rilassamento esatto
- Espansione binaria $v = \sum_k 2^k b_k$: riformula, non rafforza

Vincoli violabili, deviazioni e penalità

Introduzione: vincoli violabili

- Un vincolo che si *preferirebbe* rispettare, ma che può essere violato pagando
- Esempi: la domanda di un periodo, una preferenza di orario, un obiettivo di servizio
- Trasformare un vincolo rigido in uno morbido è spesso la differenza fra un modello inammissibile e uno utile

I vincoli

Al posto di $a'x = \beta$ si scrive

$$a'x + s^- - s^+ = \beta \qquad \text{(1 vincolo)}$$
$$s^- \geq 0 \qquad \text{(1 variabile continua)}$$
$$s^+ \geq 0 \qquad \text{(1 variabile continua)}$$

- Nell'obiettivo si aggiunge $\pi^- s^- + \pi^+ s^+$, con penalità $\pi^-, \pi^+ > 0$
- s^- misura di quanto si sta *sotto* il target, s^+ di quanto si sta *sopra*

La dimostrazione

- Il vincolo, da solo, è sempre soddisfacibile: qualunque $a'x$ si compensa con una delle due deviazioni
- **In ogni ottimo almeno una delle due deviazioni è nulla**:
  - se $s^- = \sigma > 0$ e $s^+ = \tau > 0$, sottrarre $\min(\sigma, \tau)$ da entrambe lascia il vincolo soddisfatto
  - e riduce l'obiettivo di $(\pi^- + \pi^+) \min(\sigma, \tau) > 0$
- Quindi $s^- + s^+ = |a'x - \beta|$ in ogni ottimo: il valore assoluto scritto con un'uguaglianza invece che con due disuguaglianze

Con penalità nulla la deviazione non costa niente

Osservazione 9: Con penalità nulla la deviazione non costa niente

L'argomento di scambio richiede solo $\pi^- + \pi^+ > 0$. *Con* $\pi^- = 0$ *e* $\pi^+ > 0$ *ogni ottimo ha ancora almeno una deviazione nulla, ma restare sotto* β *non costa niente: il vincolo soft non spinge più* $a'x$ *verso l'obiettivo. Solo se* $\pi^- + \pi^+ \leq 0$ *una soluzione ottima può avere entrambe le deviazioni positive (con somma negativa il modello è addirittura illimitato), e la lettura «*s^- *è la sotto-copertura» si perde. La forza della conclusione dipende dal segno dei coefficienti nell'obiettivo.*

Esempio: domanda non coperta

- Domanda 6 in ciascuno di tre periodi, disponibilità totale $15 < 18$
- Penalità $\pi^+ = 3$ e $\pi^- = 2$
- Il modello rigido sarebbe inammissibile
- Quello morbido dà $z(MILP) = 6$, con $q = (3, 6, 6)$ e una sotto-copertura di 3 concentrata sul primo periodo
- Nulla, nei dati, dice che vada concentrata: qualunque ripartizione della sotto-copertura totale 3 ha lo stesso costo, e il solver ne restituisce una

In gurobipy

```
sm = m.addVars(T, name="s_meno")
sp = m.addVars(T, name="s_piu")
m.addConstrs((q[t] + sm[t] - sp[t] == domanda[t]
              for t in range(T)), name="target")
m.setObjective(costo + gp.quicksum(
    pen_giu * sm[t] + pen_su * sp[t]
    for t in range(T)), GRB.MINIMIZE)
```

Riepilogo

- Vincolo morbido: $a'x + s^- - s^+ = \beta$ con penalità $\pi^- s^- + \pi^+ s^+$
- Con penalità strettamente positive, all'ottimo almeno una deviazione è nulla e $s^- + s^+ = |a'x - \beta|$
- Con penalità nulla la deviazione non costa niente

Funzioni lineari a tratti

Introduzione: funzioni lineari a tratti

- Il costo non è proporzionale alla quantità: cambia a scaglioni
- Sconti sopra una soglia, sovrapprezzi oltre una capacità, tariffe a fasce
- La funzione $g(q)$ è continua e lineare a tratti, con nodi $q_0 < q_1 < \cdots < q_K$ e valori $g_0, \ldots, g_K$

I vincoli

$\lambda_k \geq 0$ pesi della combinazione convessa, $w_t \in \{0, 1\}$ indicatore del tratto t (fra q_{t-1} e q_t):

$$\sum_{k=0}^{K} \lambda_k = 1, \qquad q = \sum_{k=0}^{K} q_k \lambda_k, \qquad g(q) = \sum_{k=0}^{K} g_k \lambda_k$$

$$\sum_{t=1}^{K} w_t = 1$$

$$\lambda_k \leq \sum_{t\,:\,k \in \{t-1, t\}} w_t, \qquad \forall k \in \{0, \ldots, K\} \quad (K+1 \text{ vincoli})$$

La dimostrazione

- L'ultimo vincolo è l'**adiacenza**: λ_k può essere positiva solo se il tratto scelto ha q_k come estremo
- Con $\sum_t w_t = 1$ il tratto è uno solo: al più due λ consecutive sono positive
- Allora $(q, g(q))$ è un punto del segmento fra (q_{t-1}, g_{t-1}) e (q_t, g_t): un punto *del grafico*
- Senza adiacenza si mescolano nodi non consecutivi e (q, g) finisce nell'**inviluppo convesso inferiore**, sotto la funzione

Se la funzione è convessa l'adiacenza è superflua

Osservazione 10: Se la funzione è convessa l'adiacenza è superflua, altrimenti no

Quando g è convessa e si minimizza, l'inviluppo convesso inferiore coincide con la funzione e le λ ottime risultano automaticamente adiacenti: il modello senza binarie è già corretto. Quando g non è convessa (sconti di quantità, costo marginale che scende) la differenza si vede subito.

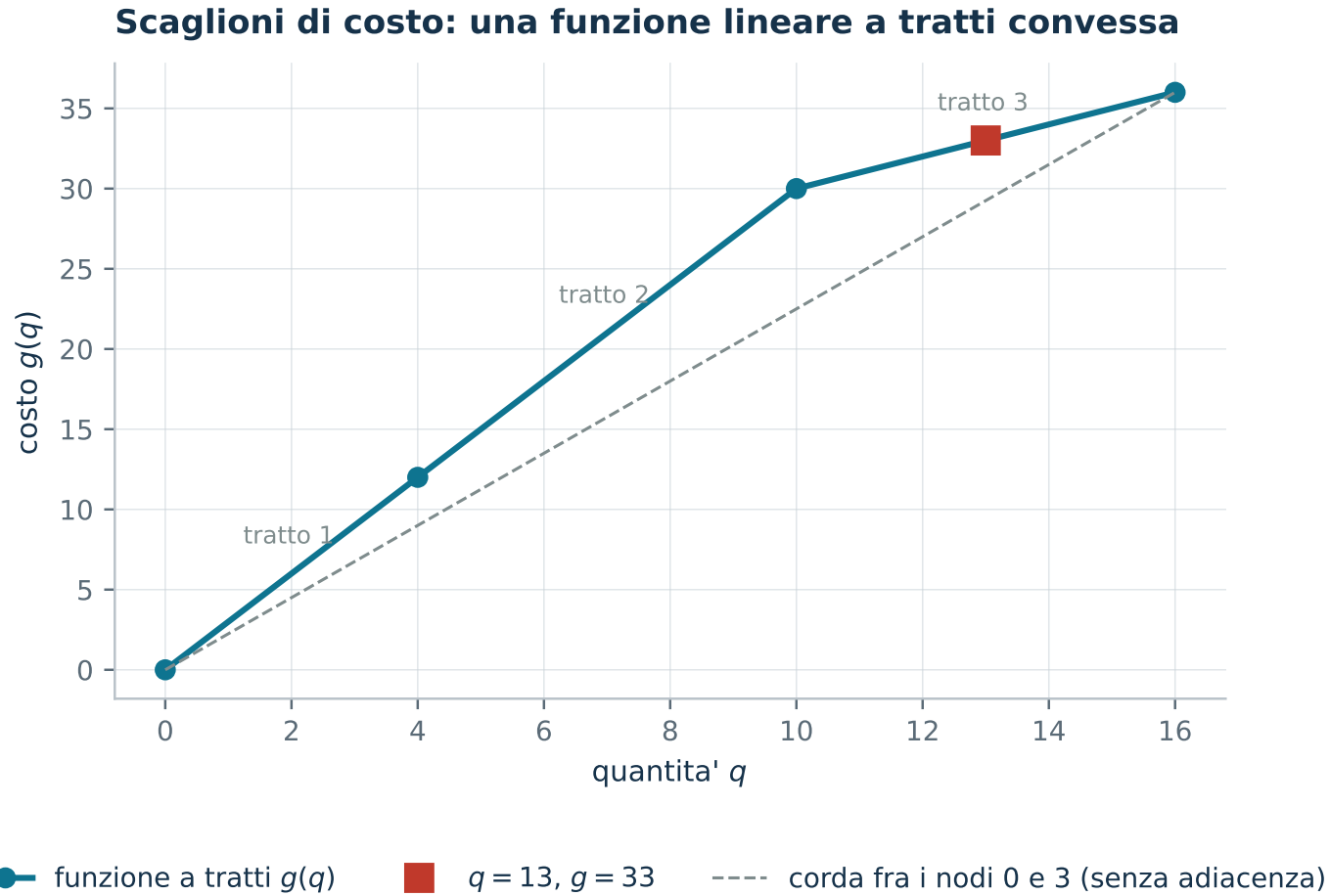
Esempio: sconto di quantità

Nodi $(0, 4, 10, 16)$, valori $(0, 12, 30, 36)$, domanda $q \geq 13$:

Formulazione	valore ottimo
senza adiacenza (combinazione libera)	$117/4 = 29{,}25$
con adiacenza	33

- La prima mescola i nodi 0 e 3: un costo che la funzione *non assume in nessun punto*, il valore dell'inviluppo
- La seconda mescola i nodi 2 e 3, adiacenti: $g(13) = 30 + 6 \cdot \frac{3}{6} = 33$

La funzione a tratti con sconto di quantità



La corda tratteggiata fra i nodi estremi è ciò che la formulazione senza adiacenza permette di raggiungere: sta *sotto* il grafico.

La forza del rilassamento

- Non confondere due cose
- L'adiacenza cambia l'*insieme intero*: i due modelli hanno ottimi diversi, 29,25 e 33
- Ma non la forza del rilassamento: con w_t frazionarie il vincolo di adiacenza non morde e i due modelli hanno lo stesso $z(LP^+) = 117/4$

In gurobipy

```
m.addConstr(lam.sum() == 1, name="convessa")
m.addConstr(q == gp.quicksum(nodi[k] * lam[k]
            for k in range(K + 1)), name="ascissa")
m.addConstr(w.sum() == 1, name="un_tratto")
# adiacenza: lambda_k solo sui tratti che toccano k
for k in range(K + 1):
    m.addConstr(lam[k] <= gp.quicksum(w[t]
                for t in (k - 1, k) if 0 <= t < K),
                name=f"adiacenza{k}")
```

Gurobi e la formulazione manuale

- Gurobi offre anche i vincoli `addGenConstrPWL` e i tipi SOS2, che fanno lo stesso lavoro internamente
- Qui la formulazione manuale resta il materiale principale, perché è quella di cui si deve saper dimostrare la correttezza

Riepilogo

- Funzione a tratti: combinazione convessa dei nodi + un tratto solo + adiacenza
- Con g convessa da minimizzare l'adiacenza è superflua; altrimenti è obbligatoria
- L'adiacenza cambia l'insieme intero, non la forza del rilassamento

La mappa delle tecniche

Introduzione: la mappa delle tecniche

- Una tabella pensata per la consultazione
- Si legge dalla prima colonna («che cosa voglio dire») alla seconda («come si scrive»)
- La terza ricorda che cosa va dichiarato ogni volta che la si usa

La mappa delle tecniche (1/4)

Tecnica	Formulazione	Da dichiarare
attivazione	$x_{ij} \leq y_j$ oppure $\sum_i x_{ij} \leq k_j y_j$	quale forma, e perché; segno di f_j
costo fisso	$q_j \leq c_j y_j$	che c_j è la capacità, non un big-M
lotto minimo	$\ell y_j \leq q_j \leq c_j y_j$	che $\ell \leq c_j$; che il rilassamento non ne risente

La mappa delle tecniche (2/4)

Tecnica	Formulazione	Da dichiarare
conteggio intero	$\sum_i a_i x_i \leq Kw$, w intera	che l'interezza realizza il tetto
ausiliaria di massimo	$z \geq t_j x_j$, $z \geq 0$	che z non compare altrove; segno nell'obiettivo
min-max / max-min	$T \geq L_k$ e min T; $U \leq L_k$ e max U	quale dei tre obiettivi, e che non si confrontano
valore assoluto	$d \geq \pm(u - v)$ in obiettivo; disgiunzione se $\geq k$	se è obiettivo o vincolo, e in quale verso

La mappa delle tecniche (3/4)

Tecnica	Formulazione	Da dichiarare
big-M	$a'x \leq b + M(1-y)$	il valore di M calcolato dai dati
precedenze	$s_{ij} + s_{ji} = 1$, $\kappa_i \geq \kappa_j + t_i - M(1-s_{ij})$	l'orizzonte e $M = \sum_h t_h$
se e solo se	$y \leq x_j$ e $y \geq \sum_j x_j - (p-1)$	se il secondo verso serve o segue dall'ottimalità

La mappa delle tecniche (4/4)

Tecnica	Formulazione	Da dichiarare
conteggio dei tipi	$\ell y_j \leq q_j \leq c_j y_j$, $\sum_j y_j \geq p$	che senza la soglia ℓ il conteggio è vuoto
alldiff / espansione	doppio partitioning; $v = \sum_k 2^k b_k$	che l'alldiff ha rilassamento esatto
vincoli violabili	$a'x + s^- - s^+ = \beta$ con penalità	i due segni delle penalità
funzione a tratti	combinazione convessa + adiacenza	se g è convessa; altrimenti l'adiacenza è obbligatoria

Le tre domande da farsi davanti a un legame nuovo

1. **Quale verso è imposto dai vincoli e quale no?** Si guarda solo i vincoli, un caso per valore della binaria
2. **Il verso mancante segue dall'ottimalità?** Argomento di scambio: dipende dal *segno* del coefficiente nell'obiettivo; strettamente diverso da zero dà «in ogni ottimo», nullo dà al più «esiste un ottimo»
3. **Quanto è forte il rilassamento?** Si confronta $z(LP^+)$ con $z(MILP)$ su un'istanza piccola e, con due formulazioni, le si confronta fra loro, dopo averle dimostrate equivalenti sui punti interi

Riepilogo

- Quattordici tecniche, ciascuna con la sua formulazione e ciò che va dichiarato
- Per ogni legame nuovo: verso imposto, verso di ottimalità, forza del rilassamento

Riepilogo finale (1/2)

Correttezza:

- Distinguere sempre ciò che è imposto dai vincoli da ciò che vale solo all'ottimo
- L'argomento di scambio dà «in ogni ottimo» solo con coefficiente strettamente diverso da zero

Coefficienti:

- Capacità e big-M si ricavano dai dati e si dichiarano
- Un M non valido cambia il problema; uno troppo grande indebolisce il rilassamento

Riepilogo finale (2/2)

Rilassamento:

- Forma disaggregata più stretta dell'aggregata
- Lotto minimo, massimo, valore assoluto e precedenze big-M danno rilassamenti deboli
- Alldiff: matrice totalmente unimodulare, rilassamento esatto

Disgiunzioni:

- Insiemi non convessi ($|u - v| \geq k$, precedenze, funzioni a tratti non convesse) richiedono binarie