Problemi e modelli

Mauro Passacantando

Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2021/22

Esempio 1 - problema

Un coltivatore ha a disposizione 12 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg di semi di lattuga, 18 t di tuberi e 160 t di concime. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata per la coltivazione di lattuga è di 3000 €/ettaro e quella delle patate è di 5000 €/ettaro. L'assorbimento delle risorse per ogni tipo di coltivazione è di 7 kg di semi e 10 t di concime per ettaro di lattuga, 3 t di tuberi e 20 t di concime per ettaro di patate. Stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga

x_P = numero di ettari da coltivare a patate

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga

x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga

x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Vincoli da rispettare:

$x_L + x_P \leq 12$ (terreno disponibile)

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga

x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Vincoli da rispettare:

$x_L + x_P \leq 12$ (terreno disponibile)

$7x_L \leq 70$ (semi di lattuga disponibili)

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga

x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Vincoli da rispettare:

$x_L + x_P \leq 12$ (terreno disponibile)

$7x_L \leq 70$ (semi di lattuga disponibili)

$3x_P \leq 18$ (tuberi disponibili)

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga
x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Vincoli da rispettare:

$x_L + x_P \leq 12$ (terreno disponibile)
$7x_L \leq 70$ (semi di lattuga disponibili)
$3x_P \leq 18$ (tuberi disponibili)
$10x_L + 20x_P \leq 160$ (concime disponibile)

Esempio 1 - modello

Variabili decisionali:

x_L = numero di ettari da coltivare a lattuga
x_P = numero di ettari da coltivare a patate

Funzione obiettivo: $3000x_L + 5000x_P$ (da massimizzare)

Vincoli da rispettare:

$x_L + x_P \leq 12$ (terreno disponibile)
$7x_L \leq 70$ (semi di lattuga disponibili)
$3x_P \leq 18$ (tuberi disponibili)
$10x_L + 20x_P \leq 160$ (concime disponibile)

Modello matematico (di programmazione lineare):

$$\begin{cases} \max\ 3000x_L + 5000x_P \\ x_L + x_P \leq 12 \\ 7x_L \leq 70 \\ 3x_P \leq 18 \\ 10x_L + 20x_P \leq 160 \\ x_L \geq 0,\ x_P \geq 0 \end{cases}$$

Esempio 2 - problema

Un *personal trainer* deve preparare un piano di allenamento settimanale di 8 ore combinando diverse attività fisiche. Nella tabella seguente sono riportate le attività possibili, le calorie consumate in un'ora di attività e il numero massimo di ore dedicabili ad ogni attività:

Attività	Camminare	Jogging	Nuoto	Ginnastica	Bicicletta
Calorie consumate	100	300	200	250	150
Max numero ore	6	3	4	3	5

Il piano di allenamento richiede almeno due ore di sport all'aperto (camminare, jogging, bicicletta), che le calorie consumate con gli sport all'aperto non superino il 50% delle calorie totali consumate e che le ore di nuoto non siano più del 10% del totale. Qual è il piano di allenamento che massimizza le calorie consumate?

Esempio 2 - modello

Variabili decisionali:
x_1 = ore dedicate a camminare,
x_2 = ore dedicate al jogging,
x_3 = ore dedicate al nuoto,
x_4 = ore dedicate alla ginnastica,
x_5 = ore dedicate alla bicicletta.

Modello matematico (di programmazione lineare):

$$\begin{cases} \max\ 100x_1 + 300x_2 + 200x_3 + 250x_4 + 150x_5 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 8 \\ x_1 + x_2 + x_5 \geq 2 \\ 100x_1 + 300x_2 + 150x_5 \leq 0.5\,(100x_1 + 300x_2 + 200x_3 + 250x_4 + 150x_5) \\ x_1 \leq 6 \\ x_2 \leq 3 \\ x_3 \leq 0.8 \\ x_4 \leq 3 \\ x_5 \leq 5 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

Esempio 3 - problema

Un'azienda produce palloni da basket e da calcio che vende rispettivamente a 20 e 15 euro ciascuno. L'azienda compra ogni settimana 400 m^2 di cuoio e ha bisogno di 16 dm^2 di cuoio per produrre un pallone da basket e 14 dm^2 per un pallone da calcio. La produzione di un pallone da basket richiede 8 minuti di lavoro di una macchina mentre quello da calcio ne richiede 12. La macchina a disposizione dell'azienda può lavorare 12 ore al giorno per 5 giorni a settimana. Dovendo produrre almeno 100 palloni da basket ed almeno 150 palloni da calcio, l'azienda vuole determinare la produzione settimanale che massimizza il profitto.

Esempio 3 - modello

Variabili decisionali:
x_B = numero di palloni da basket prodotti
x_C = numero di palloni da calcio prodotti

Modello matematico (di programmazione lineare intera):

$$\begin{cases} \max\ 20\,x_B + 15\,x_C \\ 16\,x_B + 14\,x_C \leq 40000 \\ 8\,x_B + 12\,x_C \leq 3600 \\ x_B \geq 100 \\ x_C \geq 150 \\ x_B, x_C \in \mathbb{N} \end{cases}$$

Esempio 4 - problema

Un'azienda produce 4 tipi di TV (32, 40, 50 e 55 pollici) e ha a disposizione 2 stabilimenti produttivi (A e B). L'azienda dispone di 50 operai in A e 60 in B ognuno dei quali lavora 8 ore al giorno per 5 giorni alla settimana. Le ore necessarie per produrre i TV e le richieste minime da soddisfare sono indicate nella seguente tabella:

TV	32"	40"	50"	55"
Stabilimento A	1.3	1.4	1.6	1.8
Stabilimento B	1.4	1.6	1.9	2.1
Richiesta	1000	800	500	300

Sapendo che i 4 tipi di TV vengono venduti rispettivamente a 400, 700, 900, e 1300 euro, l'azienda vuole determinare quanti TV di ogni tipo produrre nei due stabilimenti in modo da massimizzare il ricavo complessivo.

Esempio 4 - modello

Variabili decisionali:
x_{ij} = numero di TV di tipo i prodotti nello stabilimento j,
con $i = 1, 2, 3, 4$ e $j = A, B$.

Modello matematico (di programmazione lineare intera):

$$\begin{cases} \max\ 400\,(x_{1A} + x_{1B}) + 700\,(x_{2A} + x_{2B}) + 900\,(x_{3A} + x_{3B}) + 1300\,(x_{4A} + x_{4B}) \\ 1.3\,x_{1A} + 1.4\,x_{2A} + 1.6\,x_{3A} + 1.8\,x_{4A} \leq 2000 \\ 1.4\,x_{1B} + 1.6\,x_{2B} + 1.9\,x_{3B} + 2.1\,x_{4B} \leq 2400 \\ x_{1A} + x_{1B} \geq 1000 \\ x_{2A} + x_{2B} \geq 800 \\ x_{3A} + x_{3B} \geq 500 \\ x_{4A} + x_{4B} \geq 300 \\ x_{ij} \in \mathbb{N} \end{cases}$$

Esempio 5 - problema

Supponiamo di voler investire un capitale di 100 mila euro. Abbiamo a disposizione 9 investimenti possibili e per ciascuno di essi conosciamo il ricavo atteso ed il costo attuale:

Investimento	1	2	3	4	5	6	7	8	9
Ricavo atteso (k€)	50	65	35	16	18	45	45	40	25
Costo (k€)	40	50	25	10	10	40	35	30	20

Compatibilmente con il capitale disponibile, quali investimenti dobbiamo fare in modo da massimizzare il ricavo totale atteso?

Esempio 5 - modello

Variabili decisionali: $x_j = \begin{cases} 1 & \text{se l'investimento } j \text{ viene scelto,} \\ 0 & \text{altrimenti,} \end{cases}$ per $j = 1, \ldots, 9$

Modello matematico (di programmazione lineare binaria):

$$\begin{cases} \max\ 50x_1 + 65x_2 + 35x_3 + 16x_4 + 18x_5 + 45x_6 + 45x_7 + 40x_8 + 25x_9 \\ 40x_1 + 50x_2 + 25x_3 + 10x_4 + 10x_5 + 40x_6 + 35x_7 + 30x_8 + 20x_9 \leq 100 \\ x_j \in \{0, 1\}, \qquad j = 1, \ldots, 9 \end{cases}$$

Problema dello zaino (knapsack problem)

Dati n oggetti di valore $v_1, \ldots, v_n$ e peso $p_1, \ldots, p_n$, ed un contenitore di capacità C, quali oggetti inserisco nel contenitore, rispettando la sua capacità, in modo da massimizzare il valore totale?

Variabili: $$x_j = \begin{cases} 1 & \text{se oggetto } j \text{ viene inserito,} \\ 0 & \text{altrimenti.} \end{cases}$$

Modello:

$$\begin{cases} \max \displaystyle\sum_{j=1}^{n} v_j \, x_j \\ \displaystyle\sum_{j=1}^{n} p_j \, x_j \leq C \\ x_j \in \{0, 1\}, \; j = 1, \ldots, n \end{cases}$$

Problema semplice da definire, ma difficile da risolvere (NP-hard).

Problema del Bin Packing

Dati n oggetti di peso $p_1, \ldots, p_n$ e m contenitori ognuno di capacità C, trovare il minimo numero di contenitori in cui inserire tutti gli oggetti.

Variabili: $x_{ij} = \begin{cases} 1 & \text{se l'oggetto } j \text{ è inserito nel contenitore } i, \\ 0 & \text{altrimenti,} \end{cases}$ $y_i = \begin{cases} 1 & \text{se } i \text{ è usato,} \\ 0 & \text{altrimenti.} \end{cases}$

Modello:

$$\min \sum_{i=1}^{m} y_i$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall\, j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} p_j\, x_{ij} \leq C\, y_i \quad \forall\, i = 1, \ldots, m \tag{2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j$$

$$y_i \in \{0, 1\} \qquad \forall\, i$$

(1): ogni oggetto è inserito in un solo contenitore (semiassegnamento)
(2): capacità contenitori

Carico fisso - problema

Un'azienda conserviera produce tonno all'olio, tonno al vapore e tonno agli aromi. Per la produzione di ogni tipo di tonno è necessario affittare una macchina avente i seguenti costi: 200 euro a settimana per produrre tonno-olio; 150 euro a settimana per il tonno-vapore; 100 euro a settimana per il tonno-aromi. I tempi di lavorazione, le richieste di materia prima ed il ricavo di ogni prodotto sono riassunti nella seguente tabella:

prodotto	tempi di lavorazione (ore/scatola)	materia prima (kg/scatola)	ricavo (euro/scatola)
tonno-olio	3	4	6
tonno-vapore	2	3	4
tonno-aromi	6	4	7

Ogni settimana sono disponibili 150 ore di lavoro e 160 kg di tonno. Determinare il piano produttivo settimanale in modo da massimizzare il profitto complessivo.

Carico fisso - modello

Variabili:

x_1, x_2, x_3 = numero di scatole di tonno all'olio, al vapore e agli aromi prodotti.

$y_1 = \begin{cases} 1 & \text{se viene prodotto tonno all'olio,} \\ 0 & \text{altrimenti.} \end{cases}$ Analogamente si definiscono y_2, y_3.

Modello:

$$\begin{cases} \max\ 6x_1 + 4x_2 + 7x_3 - 200\, y_1 - 150\, y_2 - 100\, y_3 \\ 3x_1 + 2x_2 + 6x_3 \leq 150 \\ 4x_1 + 3x_2 + 4x_3 \leq 160 \\ x_1 \leq M\, y_1 \\ x_2 \leq M\, y_2 \\ x_3 \leq M\, y_3 \\ x_1, x_2, x_3 \in \mathbb{N} \\ y_1, y_2, y_3 \in \{0, 1\} \end{cases}$$

dove M è una costante sufficientemente grande (ad esempio, $M = 100$).

Vincoli di alternativa - problema

Un'azienda produce tre tipi di auto: utilitarie, berline, station-wagon. Le risorse necessarie, i tempi di lavorazione ed i profitti sono sintetizzati nella seguente tabella:

auto	acciaio (tonnellate)	lavoro (ore)	profitto (euro)
utilitarie	1.5	30	2000
berline	3	25	3000
station-wagon	5	40	4000

L'azienda ha a disposizione 6000 tonnellate di acciaio e 60000 ore di lavoro. Inoltre, se l'azienda produce un tipo di veicolo, allora ne deve produrre almeno 1000 unità.
Trovare il piano di produzione che massimizza il profitto totale.

Vincoli di alternativa - modello

Variabili:

$x_1, x_2, x_3 =$ numero di utilitarie, berline e station–wagon prodotte.

$$y_1 = \begin{cases} 1 & \text{se l'azienda produce utilitarie,} \\ 0 & \text{altrimenti.} \end{cases}$$ Analogamente si definiscono y_2, y_3.

Modello:

$$\begin{cases} \max\ 2000\,x_1 + 3000\,x_2 + 4000\,x_3 \\ 1.5\,x_1 + 3\,x_2 + 5\,x_3 \leq 6000 \\ 30\,x_1 + 25\,x_2 + 40\,x_3 \leq 60000 \\ x_1 \leq 2000 y_1 \\ 1000 y_1 \leq x_1 \\ x_2 \leq 2000\,y_2 \\ 1000 y_2 \leq x_2 \\ x_3 \leq 1200\,y_3 \\ 1000 y_3 \leq x_3 \\ x_1, x_2, x_3 \in \mathbb{N} \\ y_1, y_2, y_3 \in \{0, 1\} \end{cases}$$

Distribuzione di lavori - problema

Un'azienda ha a disposizione m macchine (che lavorano in parallelo) per eseguire n lavori; ogni lavoro deve essere eseguito da una sola macchina; t_{ij} è il tempo necessario alla macchina j per eseguire il lavoro i. Determinare come assegnare i lavori alle macchine in modo da effettuare tutti i lavori nel minor tempo possibile.

Esempio. $m = 4$, $n = 8$, tempi di lavorazione indicati in tabella:

	macchina 1	macchina 2	macchina 3	macchina 4
lavoro 1	6	-	7	3
lavoro 2	4	7	5	6
lavoro 3	-	5	9	4
lavoro 4	-	4	6	5
lavoro 5	5	8	2	-
lavoro 6	9	9	3	7
lavoro 7	8	2	-	6
lavoro 8	7	6	9	5

Distribuzione di lavori - modello

Variabili: t = tempo necessario per eseguire tutti i lavori

$$x_{ij} = \begin{cases} 1 & \text{se il lavoro } i \text{ è eseguito dalla macchina } j, \\ 0 & \text{altrimenti.} \end{cases}$$

Modello:

$$\begin{cases} \min \ t \\ \sum_{i=1}^{n} t_{ij}\, x_{ij} \leq t & \text{per ogni macchina } j = 1, \dots m \\ \sum_{j=1}^{m} x_{ij} = 1 & \text{per ogni lavoro } i = 1, \dots n \\ x_{ij} \in \{0, 1\} \end{cases}$$

Selezione di sottoinsiemi - problemi

Dati un insieme $I = \{1, \ldots, m\}$ ed una famiglia $S_1, \ldots, S_n$ di sottoinsiemi di I, ognuno con un costo (o valore) c_j, con $j = 1, \ldots, n$

Problema di copertura: determinare una sottofamiglia $\mathcal{F}$ di sottoinsiemi di costo minimo tale che ogni elemento di I appartenga ad almeno un sottoinsieme di $\mathcal{F}$.

Problema di partizione: determinare una sottofamiglia $\mathcal{F}$ di sottoinsiemi di costo minimo tale che ogni elemento di I appartenga esattamente ad un sottoinsieme di $\mathcal{F}$.

Problema di riempimento: determinare una sottofamiglia $\mathcal{F}$ di sottoinsiemi di valore massimo tale che ogni elemento di I appartenga ad al più un sottoinsieme di $\mathcal{F}$.

Esempio: $I = \{1, 2, 3, 4, 5, 6\}$,
$S_1 = \{1, 3\}$, $S_2 = \{2, 4\}$, $S_3 = \{2, 5, 6\}$, $S_4 = \{5, 6\}$, $S_5 = \{3, 4\}$.

$\mathcal{F}_1 = \{S_1, S_2, S_3\}$ è una copertura (ma non è una partizione)
$\mathcal{F}_2 = \{S_1, S_2, S_4\}$ è una partizione
$\mathcal{F}_3 = \{S_3, S_5\}$ è un riempimento (ma non è una partizione)

Selezione di sottoinsiemi - modelli

Definiamo la matrice: $a_{ij} = \begin{cases} 1 & \text{se } i \in S_j, \\ 0 & \text{altrimenti.} \end{cases}$

Ad ogni sottofamiglia $\mathcal{F}$ associamo una variabile x, dove $x_j = \begin{cases} 1 & \text{se } S_j \in \mathcal{F}, \\ 0 & \text{altrimenti.} \end{cases}$

Copertura

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall\, i \\ x_j \in \{0,1\} \quad \forall\, j \end{cases}$$

Partizione

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall\, i \\ x_j \in \{0,1\} \quad \forall\, j \end{cases}$$

Riempimento

$$\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall\, i \\ x_j \in \{0,1\} \quad \forall\, j \end{cases}$$

Problemi di copertura - esempio

Dislocazione mezzi soccorso

Insieme U di siti in cui sono presenti utenti.
Insieme M di siti dove è possibile dislocare ambulanze.
t_{ij} = tempo necessario utente in $i \in U$ sia raggiunto da ambulanza posta in $j \in M$
c_j = costo di attivazione del sito $j \in M$
T = massima attesa per ogni utente.

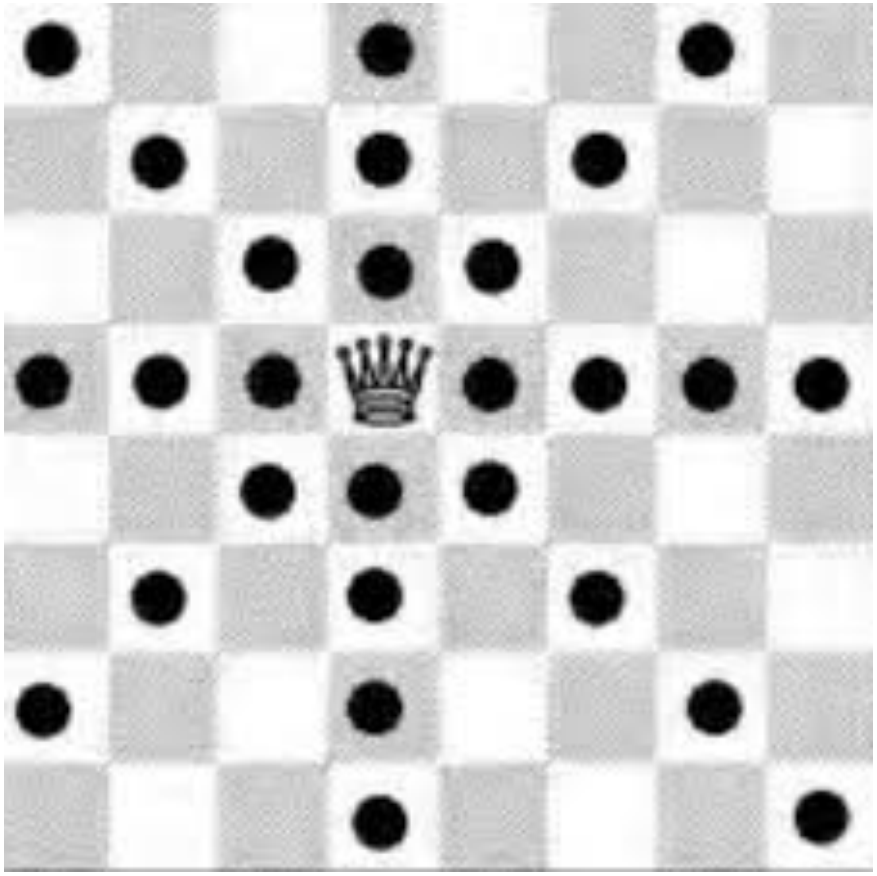
Obiettivo: decidere in quali siti dislocare ambulanze in modo da minimizzare il costo totale con la garanzia che ogni utente possa essere raggiunto in al più T minuti.

Il problema può essere formulato come problema di copertura, dove
$I = U$
sottoinsiemi $S_j = \{$utenti raggiunti entro T minuti da un'ambulanza posta in $j\}$.

Problemi di copertura - Esercizio 1.1

Si consideri il problema di disporre in una scacchiera il minimo numero di regine in modo che tutte le caselle non occupate risultino "sotto attacco" da almeno una regina.

Formularlo come problema di copertura. Come definire l'insieme I? Come definire la famiglia di sottoinsiemi di I?



Problemi di partizione - esempio

Pianificazione di equipaggi

Compagnia aerea ha m rotte (punto partenza, punto arrivo, durata volo).
Esistono n possibili insiemi di rotte (*round trip*) che può fare un equipaggio.
c_j = costo del *round trip* j.

Obiettivo: determinare un insieme di *round trip* con il minimo costo totale in modo che ogni rotta appartenga ad esattamente un *round trip*.

Il problema può essere formulato come problema di partizione, dove:
$I = \{\text{rotte}\}$, sottoinsiemi S_j = *round trip*.

Problemi di partizione - Esercizio 1.2

Considerare il problema del Sudoku e formularlo come problema di partizione. Come definire l'insieme I? Come definire la famiglia di sottoinsiemi di I?

			1			7	4	
	5			9			3	2
		6	7			9		
4			8					
	2						1	
					9			5
		4			7	3		
7	3			2			6	
	6	5			4			

Problemi di riempimento - esempio

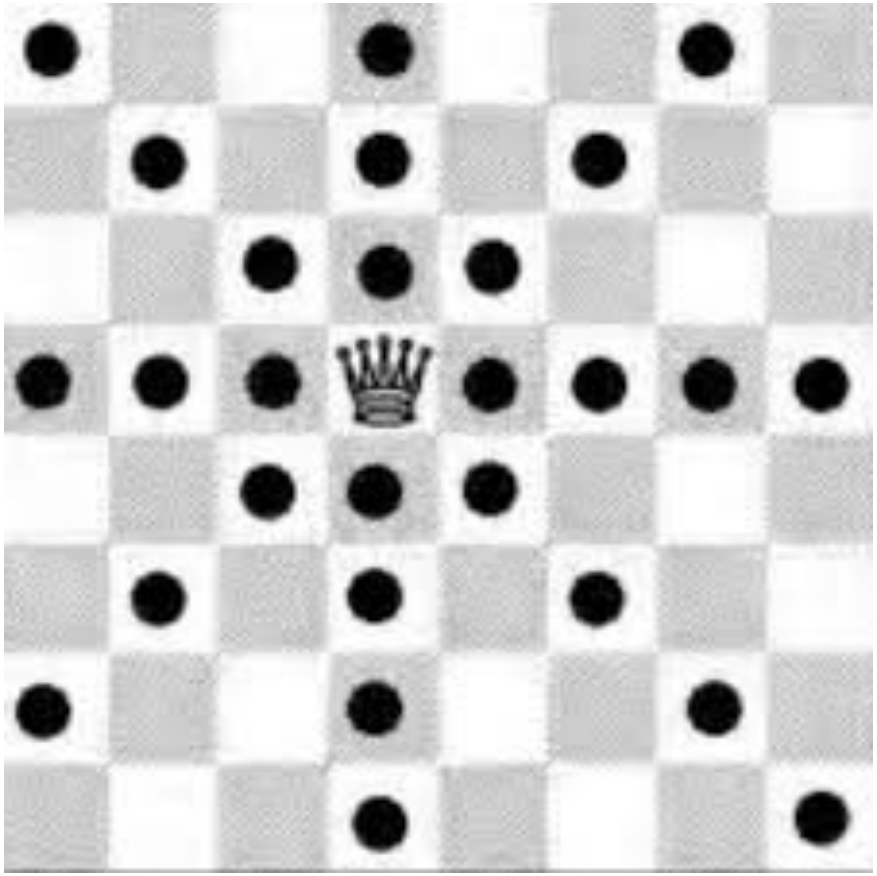
Lezioni al tempo del Covid-19

L'Università di Pisa sta valutando se svolgere le lezioni del secondo semestre in presenza, rispettando le norme sul distanziamento sociale dovute all'emergenza Covid-19. L'ateneo deve decidere, per ogni aula, qual è il massimo numero di studenti che possono essere presenti, in modo che ogni studente sia lontano almeno 1 m dagli altri studenti.

Il problema può essere formulato come problema di riempimento, dove
$J = \{\text{posti disponibili in aula}\} = \{1, \ldots, n\}$,
$I = \{(a, b) \in J \times J : \ a \text{ e } b \text{ sono lontani meno di 1 m, con } a < b\}$,
per ogni $j \in J$, il sottoinsieme $S_j = \{(a, b) \in I : \ a = j \text{ oppure } b = j\}$ di valore 1.

Problemi di riempimento - Esercizio 1.3

Si consideri il problema di disporre in una scacchiera il massimo numero di regine che non si "attaccano" reciprocamente.
Formularlo come problema di riempimento. Come definire l'insieme I? Come definire la famiglia di sottoinsiemi di I?



Esercizi di riepilogo - studente pigro

Liso si è iscritto nell'a.a. 2020/21 al corso di laurea in Informatica. Essendo molto pigro, capisce dopo le prime settimane di lezione che non riuscirà a superare tutti gli esami del primo anno. D'altra parte, secondo il regolamento del corso di laurea,

> *uno studente può sostenere esami di insegnamenti del secondo e del terzo anno solo dopo avere conseguito almeno 25 CFU di insegnamenti del primo anno, di cui almeno 12 CFU di insegnamenti del settore INF e almeno 9 CFU di insegnamenti MAT.*

Poiché non vuole essere costretto a sostenere solo esami del primo anno anche l'anno successivo, Liso vuole trovare quali sono gli esami del primo anno che deve superare in modo da poter sostenere l'anno successivo anche esami del secondo anno, cercando però di minimizzare il numero di CFU complessivi.

Insegnamenti del primo anno	CFU	Settore
Programmazione e Algoritmi	15	INF
Laboratorio I	12	INF
Fondamenti dell'informatica	9	INF
Analisi matematica	12	MAT
Algebra lineare	6	MAT
Lingua inglese	3	

Esercizi di riepilogo - studente pigro

Variabili:

Modello:

Esercizi di riepilogo - facility location

La direzione di un'azienda decide di aprire al più n magazzini per servire m punti vendita. La direzione deve stabilire anche la capacità di ogni magazzino aperto, che può essere scelta nell'insieme $\{u_1, u_2, u_3\}$. Il costo c_{hj} di attivazione del magazzino j dipende dalla capacità u_h scelta ($j = 1, \ldots, n$, $h = 1, 2, 3$). Sapendo che il punto vendita i ha una domanda d_i e che ogni punto vendita deve essere servito da un solo magazzino, la direzione deve stabilire quali magazzini aprire (e le relative capacità) ed assegnare i punti vendita ai magazzini aperti in modo da rispettare le capacità dei magazzini e minimizzando il costo totale di attivazione.

Esercizi di riepilogo - facility location

Variabili: $x_{ij} = \begin{cases} 1 & \text{se il punto vendita } i \text{ è assegnato al magazzino } j, \\ 0 & \text{altrimenti.} \end{cases}$

$y_{hj} = \begin{cases} 1 & \text{se il magazzino } j \text{ viene aperto con capacità } u_h, \\ 0 & \text{altrimenti.} \end{cases}$

Modello:

$$\min \sum_{h=1}^{3} \sum_{j=1}^{n} c_{hj} y_{hj}$$

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

$$\sum_{h=1}^{3} y_{hj} \leq 1 \qquad \forall\ j = 1, \ldots, n$$

$$\sum_{i=1}^{m} d_i x_{ij} \leq \sum_{h=1}^{3} u_h y_{hj} \qquad \forall\ j = 1, \ldots, n$$

$$x_{ij}, y_{hj} \in \{0, 1\}$$

Esercizi di riepilogo - distribuzione merce

Un'azienda produttrice di pasta dispone di n depositi, aventi ciascuno una capacità mensile pari a U_j scatole di pasta ($j = 1, \ldots, n$), e m centri di distribuzione, ognuno dei quali richiede d_i scatole di pasta al mese ($i = 1, \ldots, m$). Il rifornimento della pasta dai depositi ai centri di distribuzione è organizzato in modo che ogni centro deve essere rifornito da un unico deposito. Il deposito j ha un costo di spedizione c_j per ogni scatola inviata ai centri di distribuzione ed un costo fisso di gestione mensile f_j nel caso in cui rifornisca almeno un centro. L'azienda deve stabilire come rifornire mensilmente i centri di distribuzione in modo da minimizzare il costo totale.

Esercizi di riepilogo - distribuzione merce

Variabili: $x_{ij} = \begin{cases} 1 & \text{se il centro } i \text{ è servito dal deposito } j, \\ 0 & \text{altrimenti,} \end{cases}$

$y_j = \begin{cases} 1 & \text{se il deposito } j \text{ rifornisce almeno un centro,} \\ 0 & \text{altrimenti.} \end{cases}$

Modello:

$$\min \sum_{j=1}^{n} c_j \sum_{i=1}^{m} d_i x_{ij} + \sum_{j=1}^{n} f_j y_j$$

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

$$\sum_{i=1}^{m} d_i x_{ij} \leq U_j y_j \qquad \forall\ j = 1, \ldots, m$$

$$x_{ij} \in \{0, 1\}$$

$$y_j \in \{0, 1\}$$

Esercizi di riepilogo - presidi sanitari

L'amministratore sanitario di una città costituita da m quartieri, ognuno dei quali abitato da p_i persone ($i = 1, \ldots, m$), vuole organizzare n presidi sanitari ed assegnare ad essi gli m quartieri. Per ragioni di equità, il rapporto tra il minimo ed il massimo numero di abitanti assegnati ad un presidio deve essere almeno $1/2$. Sapendo che c_{ij} è il costo dovuto all'assegnamento del quartiere i al presidio j, l'amministratore deve decidere come assegnare i quartieri ai presidi, rispettando il vincolo di equità, in modo da minimizzare il costo totale.

Esercizi di riepilogo - presidi sanitari

Variabili: $x_{ij} = \begin{cases} 1 & \text{se il quartiere } i \text{ è assegnato al presidio } j, \\ 0 & \text{altrimenti,} \end{cases}$

e due variabili ausiliarie $u, v \in \mathbb{R}$.

Modello:

$$
\begin{aligned}
\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} & \\
\sum_{j=1}^{n} x_{ij} = 1 & \qquad \forall\ i = 1, \ldots, m \\
u \leq \sum_{i=1}^{m} p_i x_{ij} & \qquad \forall\ j = 1, \ldots, n \\
v \geq \sum_{i=1}^{m} p_i x_{ij} & \qquad \forall\ j = 1, \ldots, n \\
u \geq v/2 & \\
x_{ij} \in \{0, 1\} &
\end{aligned}
$$

Esercizi di riepilogo - presidi sanitari (variante)

L'amministratore sanitario di una città costituita da m quartieri, ognuno dei quali abitato da p_i persone ($i = 1, \ldots, m$), vuole organizzare n presidi sanitari ed assegnare ad essi gli m quartieri. L'amministratore deve decidere come assegnare i quartieri ai presidi in modo che il numero di abitanti assegnati ai presidi sia il più uniforme possibile, ossia vuole minimizzare la massima differenza di abitanti tra un presidio e l'altro.

Esercizi di riepilogo - presidi sanitari (variante)

Variabili: $x_{ij} = \begin{cases} 1 & \text{se il quartiere } i \text{ è assegnato al presidio } j, \\ 0 & \text{altrimenti,} \end{cases}$

e una variabile ausiliaria $u \in \mathbb{R}$.

Modello:

$$\min \ u$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\ i = 1, \dots, m$$

$$u \geq \sum_{i=1}^{m} p_i x_{ij} - \sum_{i=1}^{m} p_i x_{ik} \qquad \forall\ j, k = 1, \dots, n$$

$$x_{ij} \in \{0, 1\}$$

Albero di copertura di costo minimo

Mauro Passacantando

Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2021/22

Problema

Dato un grafo $G = (N, A)$ non orientato, in cui ad ogni arco $\{i, j\}$ è associato un costo c_{ij}, trovare un albero di copertura T di costo minimo, dove il costo di T è definito come $\sum_{\{i,j\} \in T} c_{ij}$.

Senza perdita di generalità, possiamo supporre che il grafo G sia completo, eventualmente aggiungendo archi di costo M, dove M è una costante sufficientemente grande (ad esempio $M > \max_{\{i,j\} \in A} c_{ij}$).

Per semplicità, nel seguito ogni arco $\{i, j\}$ con $i < j$ sarà indicato con (i, j).

Modello 1

Sia $N = \{1, \ldots, n\}$.

Per ogni $i, j \in N$ con $i < j$, definiamo le variabili decisionali:

$$x_{ij} = \begin{cases} 1 & \text{se l'albero contiene l'arco } (i,j), \\ 0 & \text{altrimenti.} \end{cases}$$

Modello di ottimizzazione:

$$\begin{cases} \min \displaystyle\sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} c_{ij} x_{ij} & \\ \displaystyle\sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} x_{ij} = n - 1 & \text{(l'albero contiene } n-1 \text{ archi)} \\ \displaystyle\sum_{i \in S} \sum_{\substack{j \in S \\ j > i}} x_{ij} \leq |S| - 1 & \forall\ S \subset N \text{ tale che } |S| \geq 3 \\ & \text{(eliminazione dei cicli)} \\ x_{ij} \in \{0, 1\} & i, j \in N \text{ con } i < j \end{cases}$$

Modello 2

Variabili decisionali: $x_{ij} = \begin{cases} 1 & \text{se l'albero contiene l'arco } (i,j), \\ 0 & \text{altrimenti.} \end{cases}$

Modello di ottimizzazione:

$$\begin{cases} \min \displaystyle\sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} c_{ij} x_{ij} & \\ \displaystyle\sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} x_{ij} = n - 1 & \text{(l'albero contiene } n-1 \text{ archi)} \\ \displaystyle\sum_{i \in S} \sum_{\substack{j \notin S \\ j > i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j > i}} x_{ij} \geq 1 & \forall\, S \subset N \text{ tale che } |S| \geq 1 \\ & \text{(vincoli di connessione)} \\ x_{ij} \in \{0,1\} & i,j \in N \text{ con } i < j \end{cases}$$

Condizione di ottimalità basata sui cicli

Osservazione. Se ad un albero di copertura T viene aggiunto un arco $(u, v) \notin T$, si forma un ciclo (perché in T esiste un cammino da u a v).

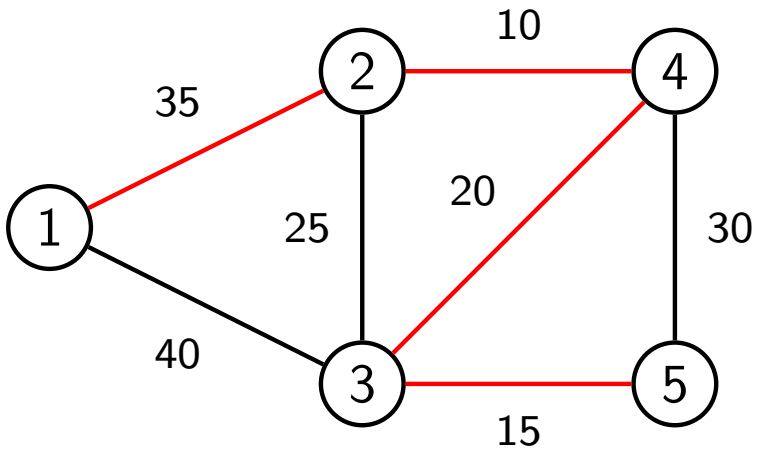
Teorema

Sia T un albero di copertura.
T è un albero di copertura di costo minimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:

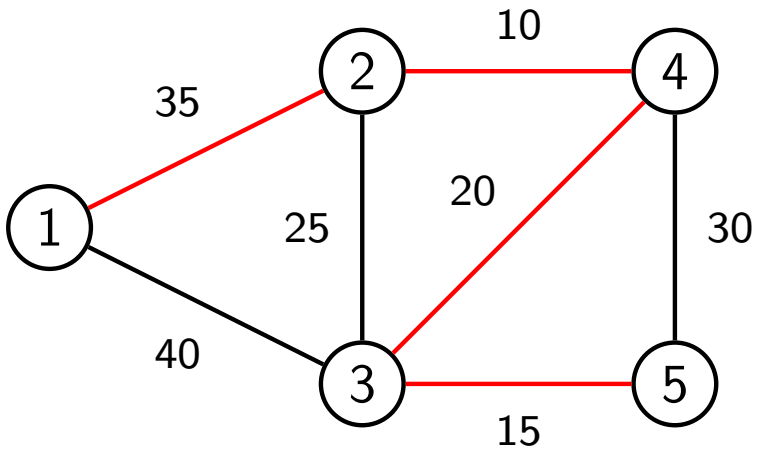


Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



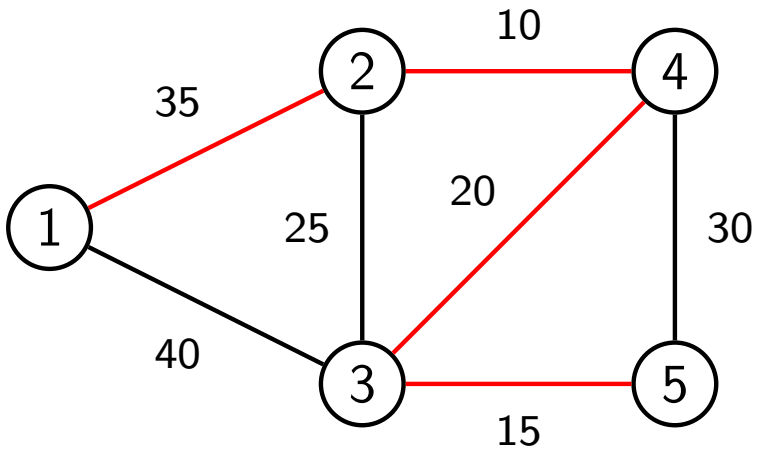
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \notin T_1$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-4-3-1	(1,2) (2,4) (3,4)	35, 10, 20	si

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



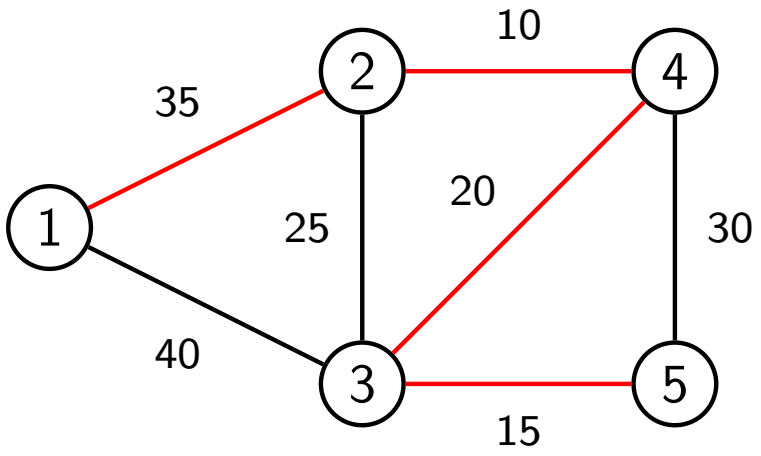
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \notin T_1$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-4-3-1	(1,2) (2,4) (3,4)	35, 10, 20	si
(2,3)	25	2-4-3-2	(2,4) (3,4)	10, 20	si

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

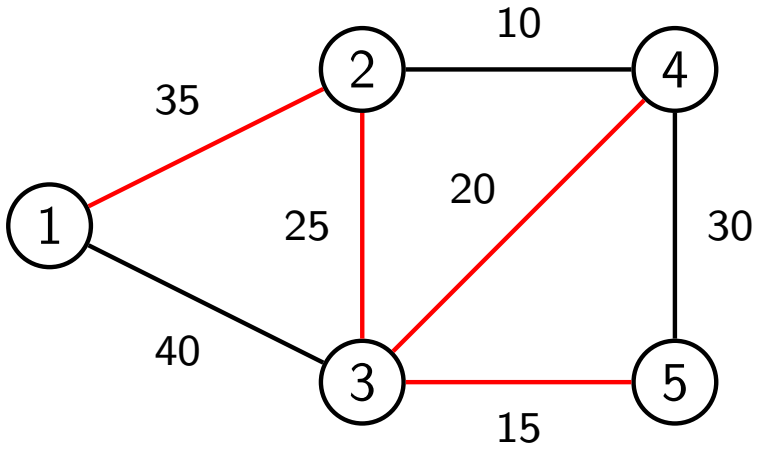
$(u, v) \notin T_1$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-4-3-1	(1,2) (2,4) (3,4)	35, 10, 20	si
(2,3)	25	2-4-3-2	(2,4) (3,4)	10, 20	si
(4,5)	30	4-3-5-4	(3,4) (3,5)	20, 15	si

Quindi T_1 è un albero di copertura di costo minimo.

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:

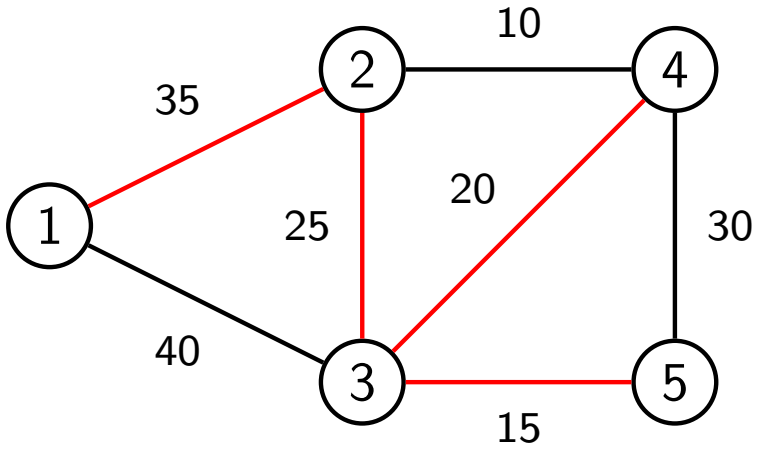


Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



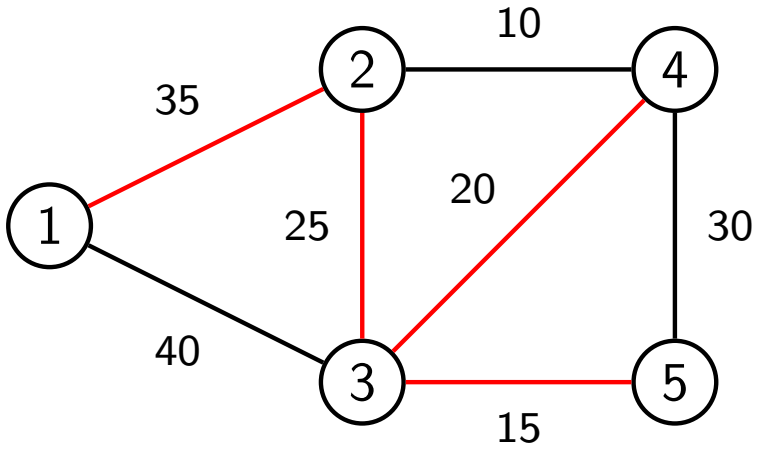
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \notin T_2$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-3-1	(1,2) (2,3)	35, 25	si

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



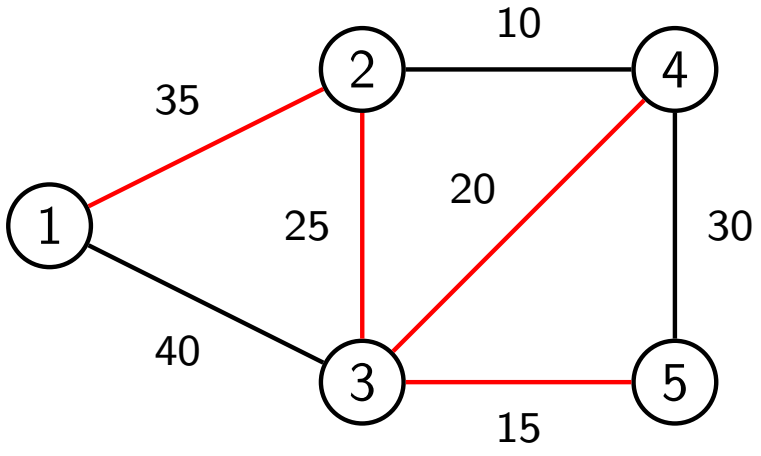
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \notin T_2$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-3-1	(1,2) (2,3)	35, 25	si
(2,4)	10	2-3-4-2	(2,3) (3,4)	25, 20	no

Condizione di ottimalità basata sui cicli

T è ottimo se e solo se per ogni arco $(u, v) \notin T$ si ha $c_{uv} \geq c_{ij}$ per ogni arco (i, j) appartenente al ciclo ottenuto aggiungendo a T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \notin T_2$	c_{uv}	ciclo C	altri archi di C	costi	cond. ott. vera
(1,3)	40	1-2-3-1	(1,2) (2,3)	35, 25	si
(2,4)	10	2-3-4-2	(2,3) (3,4)	25, 20	no
(4,5)	30	4-3-5-4	(3,4) (3,5)	20, 15	si

Quindi T_2 non è un albero di copertura di costo minimo.

Condizione di ottimalità basata sui tagli

Osservazione. Se da un albero di copertura T viene eliminato un arco $(u, v) \in T$, si formano 2 componenti connesse N' e N'', e quindi un taglio (N', N'') tale che (u, v) è l'unico arco del taglio che appartiene a T.

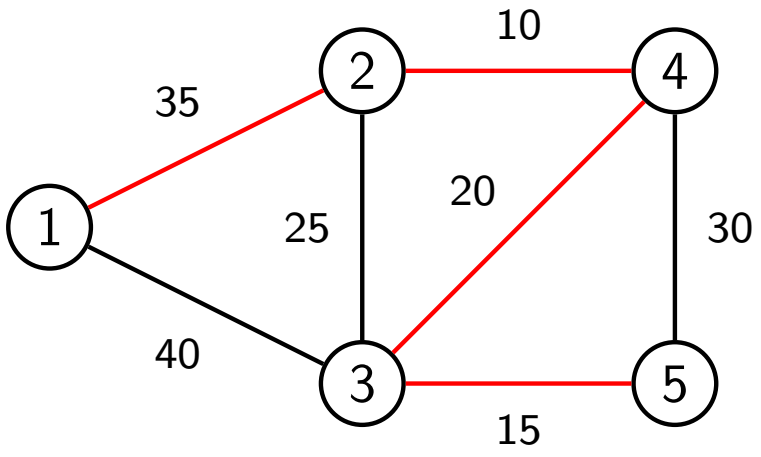
Teorema

Sia T un albero di copertura.
T è un albero di copertura di costo minimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) del taglio ottenuto eliminando da T l'arco (u, v).

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:

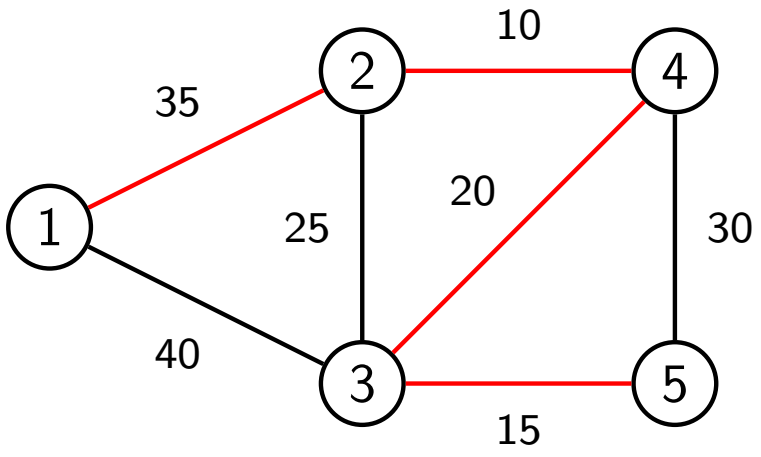


Applicando la condizione di ottimalità basata sui tagli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



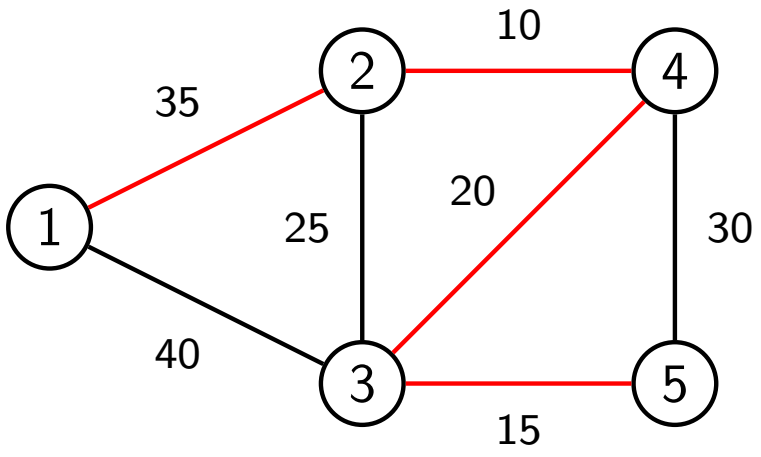
Applicando la condizione di ottimalità basata sui tagli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_1$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



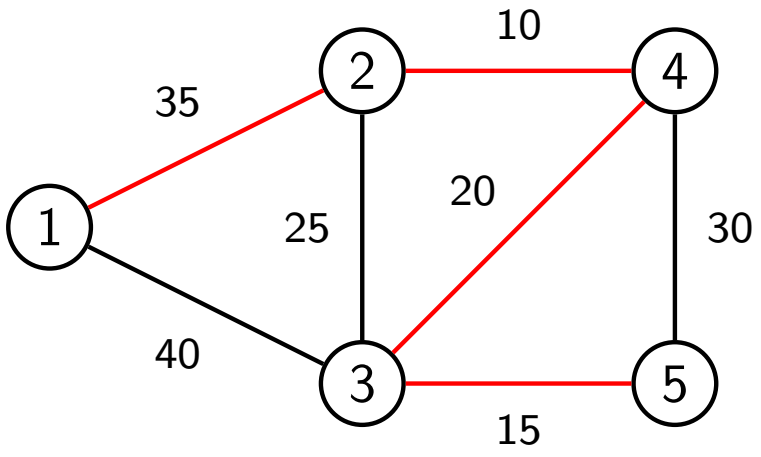
Applicando la condizione di ottimalità basata sui tagli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_1$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,4)	10	({1,2}, {3,4,5})	(1,3) (2,3)	40, 25	si

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



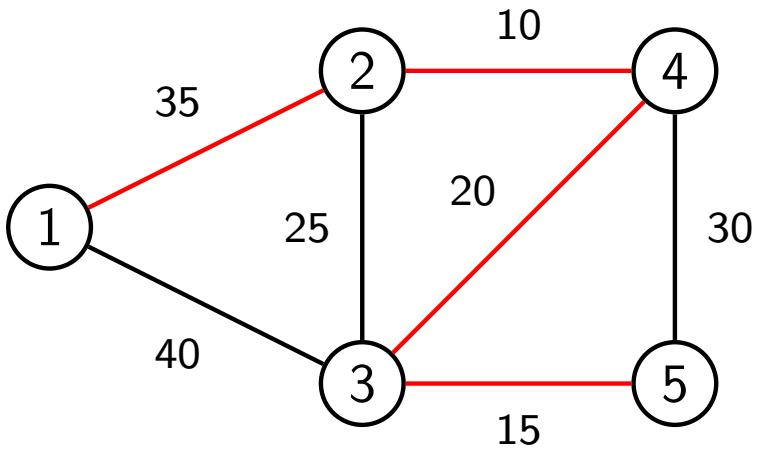
Applicando la condizione di ottimalità basata sui tagli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_1$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,4)	10	({1,2}, {3,4,5})	(1,3) (2,3)	40, 25	si
(3,4)	20	({1,2,4}, {3,5})	(1,3) (2,3) (4,5)	40, 25, 30	si

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 1. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



Applicando la condizione di ottimalità basata sui tagli, dire se l'albero di copertura $T_1 = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$ è ottimo.

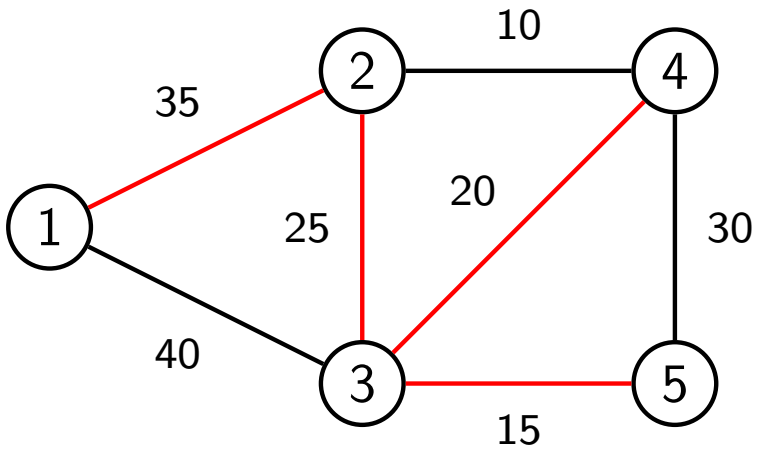
$(u, v) \in T_1$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,4)	10	({1,2}, {3,4,5})	(1,3) (2,3)	40, 25	si
(3,4)	20	({1,2,4}, {3,5})	(1,3) (2,3) (4,5)	40, 25, 30	si
(3,5)	15	({1,2,3,4}, {5})	(4,5)	30	si

Quindi T_1 è un albero di copertura di costo minimo.

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:

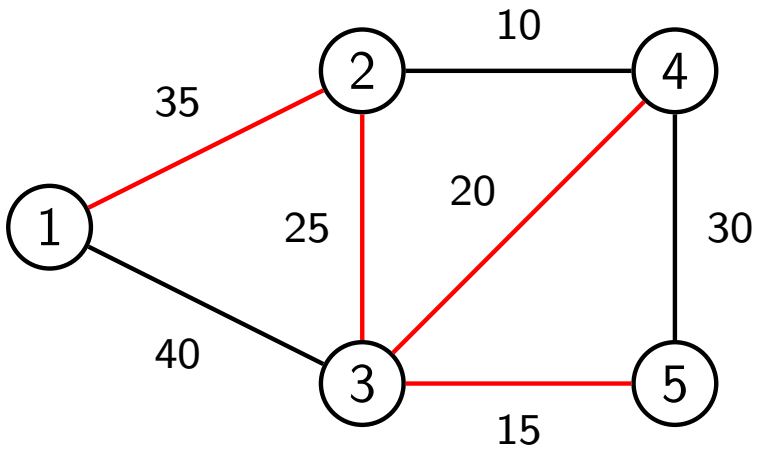


Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



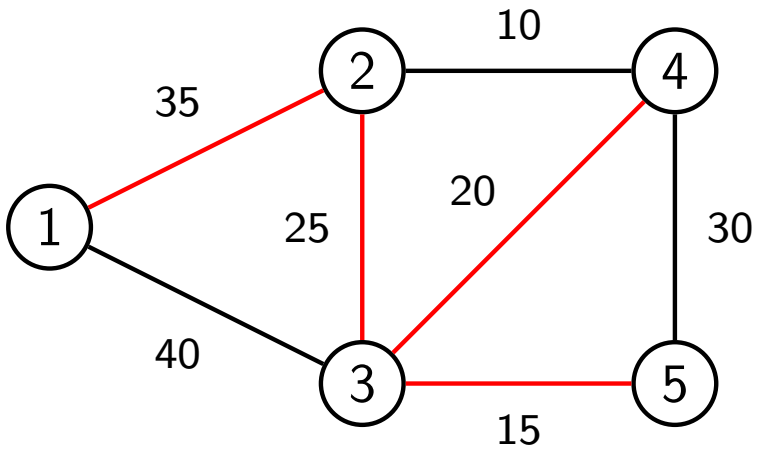
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_2$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



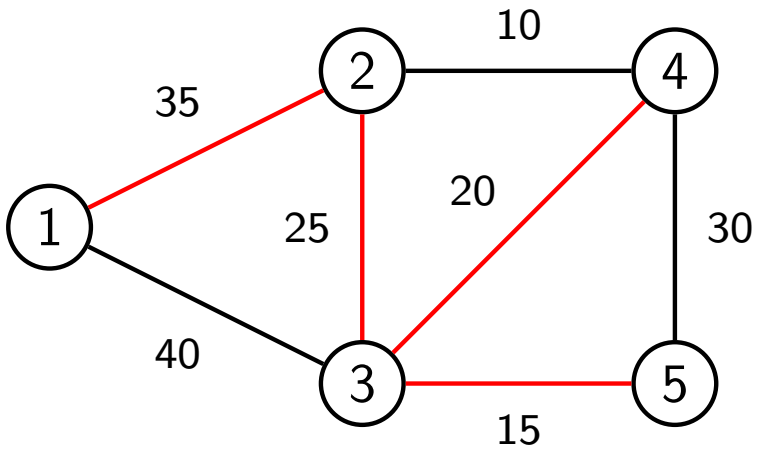
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_2$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,3)	25	({1,2}, {3,4,5})	(1,3) (2,4)	40, 10	no

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



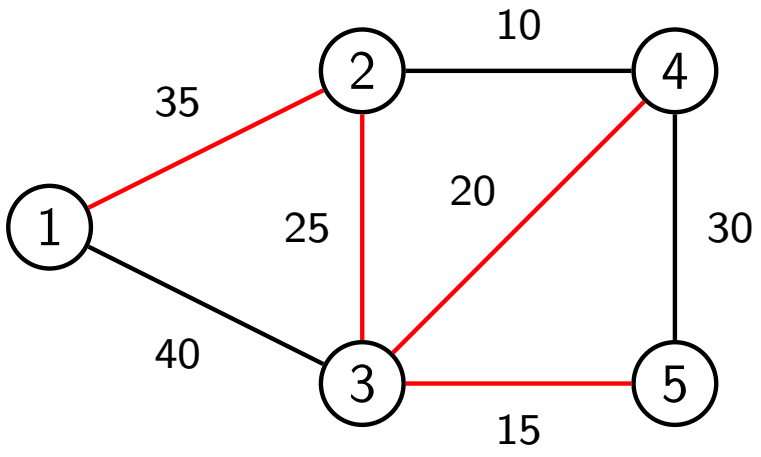
Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_2$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,3)	25	({1,2}, {3,4,5})	(1,3) (2,4)	40, 10	no
(3,4)	20	({1,2,3,5}, {4})	(1,3) (2,4) (4,5)	10, 30	no

Condizione di ottimalità basata sui tagli

T è ottimo se e solo se per ogni arco $(u, v) \in T$ si ha $c_{uv} \leq c_{ij}$ per ogni arco (i, j) appartenente al taglio ottenuto eliminando da T l'arco (u, v).

Esempio 2. Si consideri il problema dell'albero di copertura di costo minimo sul seguente grafo:



Applicando la condizione di ottimalità basata sui cicli, dire se l'albero di copertura $T_2 = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ è ottimo.

$(u, v) \in T_2$	c_{uv}	taglio (N', N'')	altri archi taglio	costi	cond. vera
(1,2)	35	({1}, {2,3,4,5})	(1,3)	40	si
(2,3)	25	({1,2}, {3,4,5})	(1,3) (2,4)	40, 10	no
(3,4)	20	({1,2,3,5}, {4})	(1,3) (2,4) (4,5)	10, 30	no
(3,5)	15	({1,2,3,4}, {5})	(4,5)	30	si

Quindi T_2 non è un albero di copertura di costo minimo.

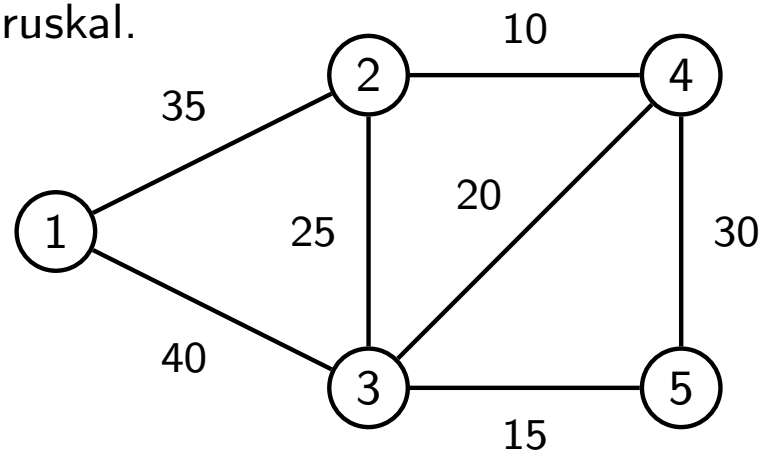
Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

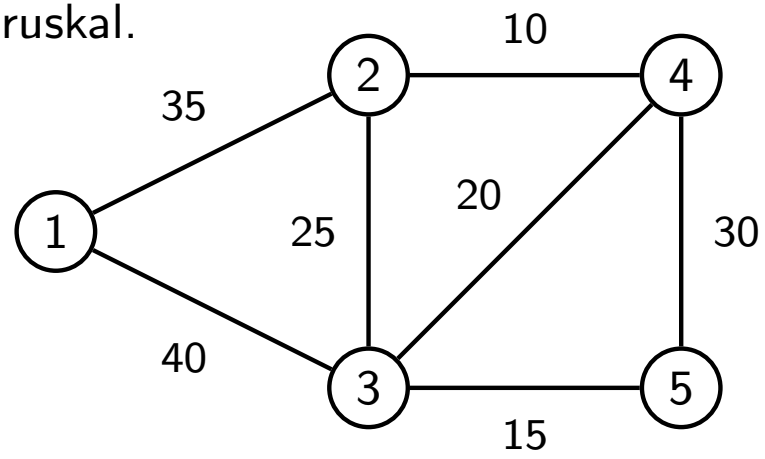


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.
$T = \emptyset$,

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

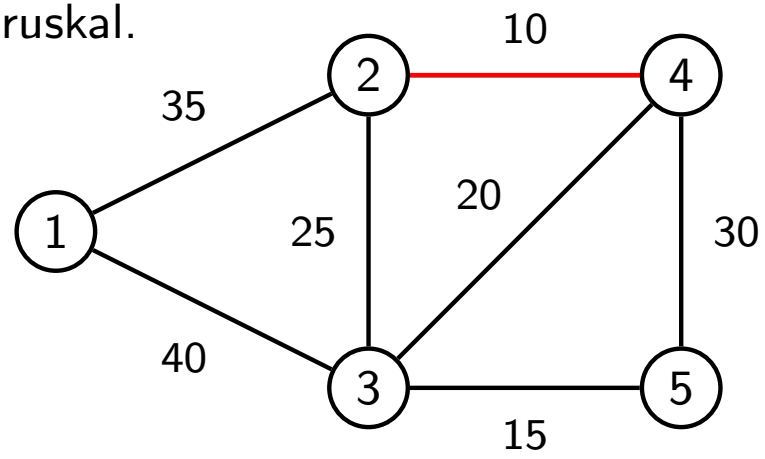


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.
$T = \emptyset$, $(2,4)$ forma un ciclo con gli archi di T? NO

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

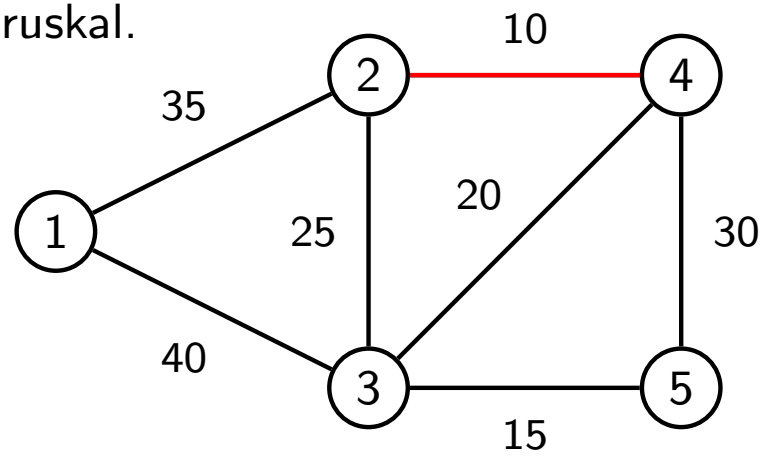


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.
$T = \emptyset$, $(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

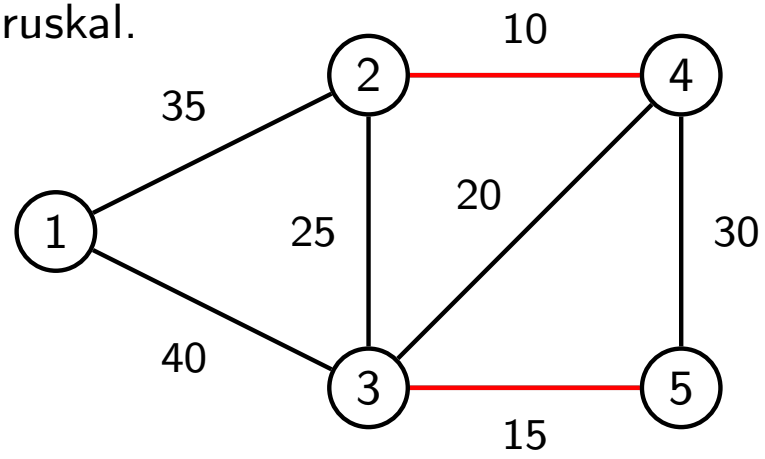
$T = \emptyset$, $(2,4)$ forma un ciclo con gli archi di T? NO

$T = \{(2,4)\}$, $(3,5)$ forma un ciclo con gli archi di T? NO

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



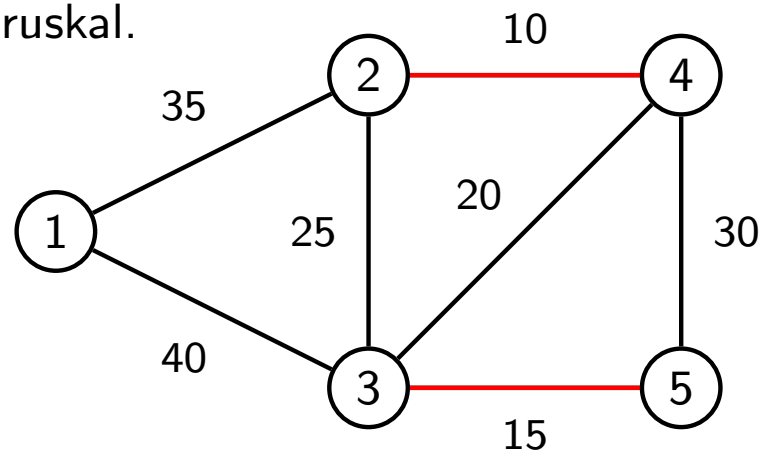
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$, $(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$, $(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



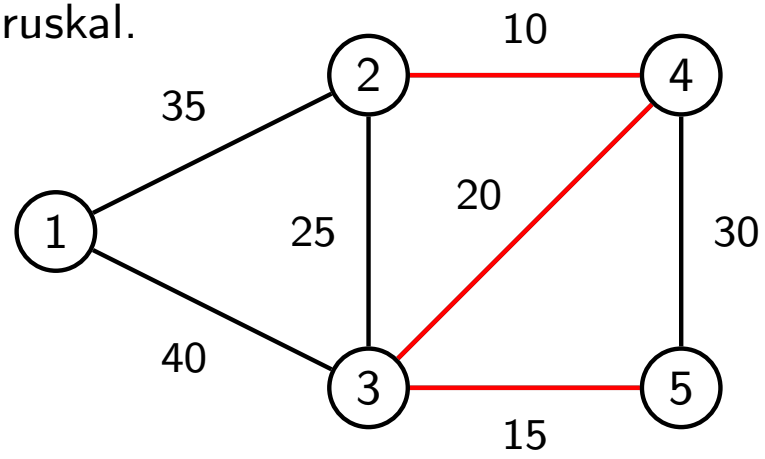
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



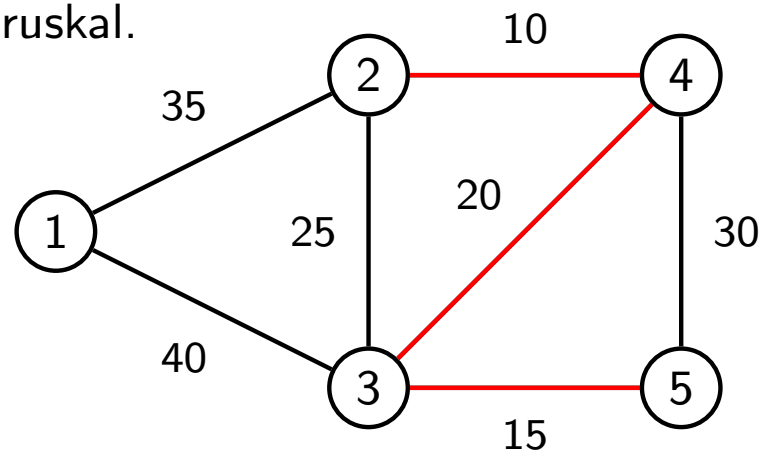
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$, $(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$, $(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$, $(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



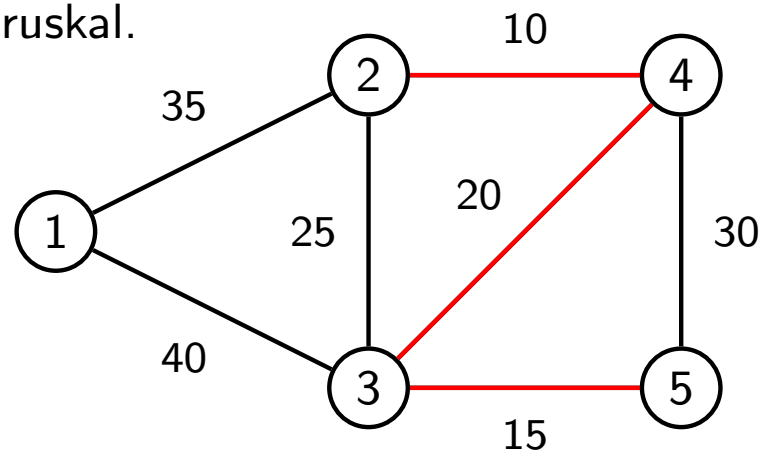
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



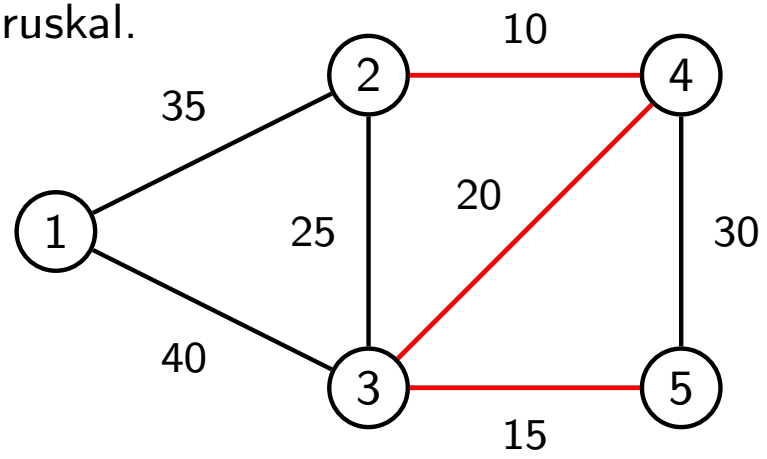
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



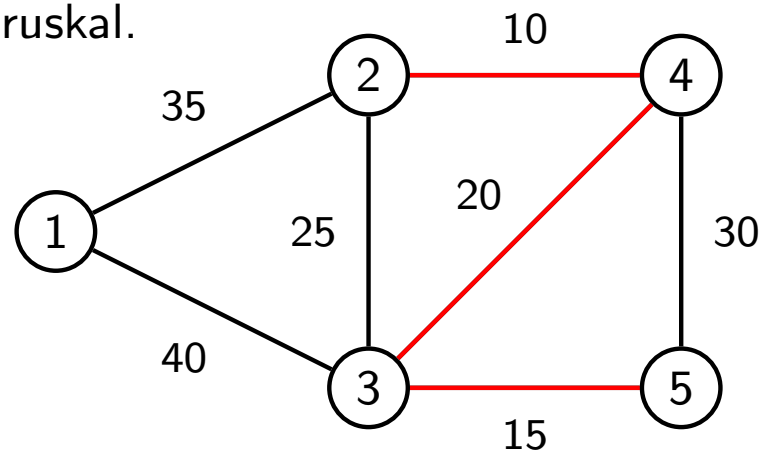
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(4,5)$ forma un ciclo con gli archi di T? SI

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



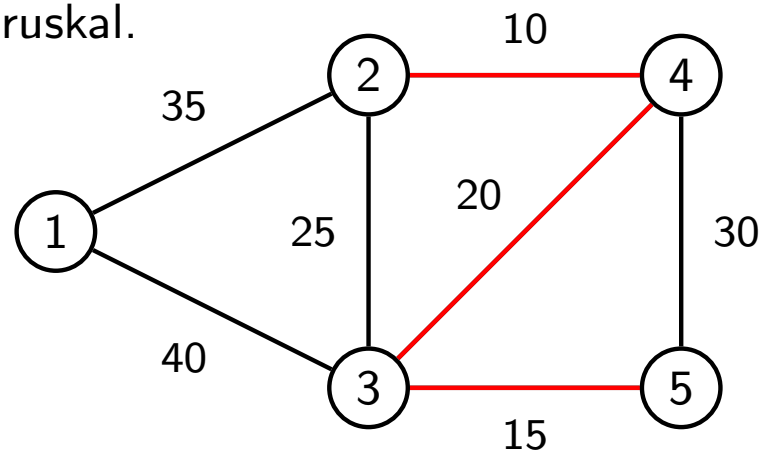
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(4,5)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



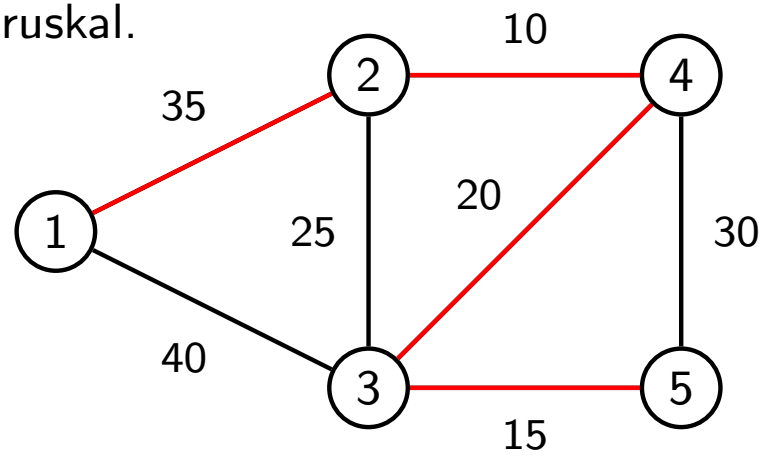
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(4,5)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(1,2)$ forma un ciclo con gli archi di T? NO

Algoritmo di Kruskal

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$.
1. Se $|T| = n - 1$ allora stop
2. Se a_k non forma un ciclo con gli archi di T allora $T = T \cup \{a_k\}$
3. $k = k + 1$ e torna al passo 1.

Esempio 3. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	$(2,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4)\}$,	$(3,5)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5)\}$,	$(3,4)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4)\}$,	$(2,3)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(4,5)$ forma un ciclo con gli archi di T? SI
$T = \{(2,4),(3,5),(3,4)\}$,	$(1,2)$ forma un ciclo con gli archi di T? NO
$T = \{(2,4),(3,5),(3,4),(1,2)\}$	STOP

Algoritmo di Kruskal

Come possiamo implementare in modo rigoroso la condizione del passo 2

"Se a_k non forma un ciclo con gli archi di T"?

Algoritmo di Kruskal

Come possiamo implementare in modo rigoroso la condizione del passo 2

"Se a_k non forma un ciclo con gli archi di T"?

Utilizzando opportunamente delle <u>etichette</u> sui nodi.

0. Ordina gli archi $a_1, \ldots, a_m$ in ordine crescente di costo. $T = \emptyset$, $k = 1$. Associa ad ogni nodo i un'etichetta $d_i = i$.

1. Se $|T| = n - 1$ allora stop

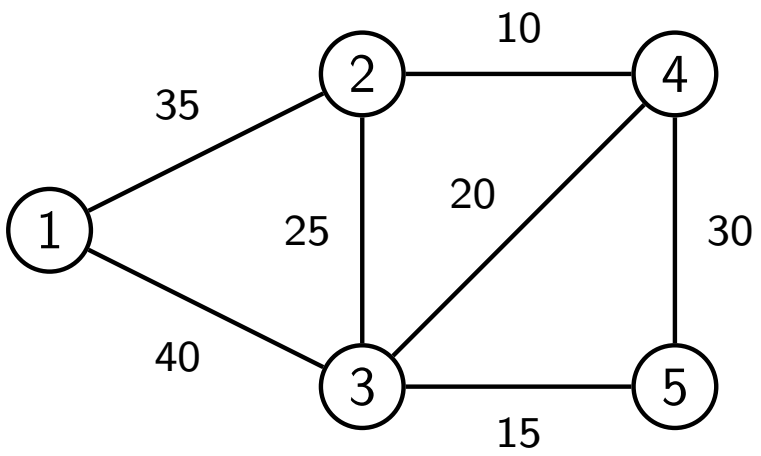
2. Sia $a_k = (p, q)$.
Se $d_p \neq d_q$ (cioè a_k non forma un ciclo con gli archi di T) allora
[aggiorna l'albero] $T = T \cup \{a_k\}$
[aggiorna le etichette] per ogni nodo i con $d_i = \max\{d_p, d_q\}$ poni $d_i = \min\{d_p, d_q\}$

3. $k = k + 1$ e torna al passo 1.

Le etichette sui nodi individuano le componenti connesse di T, cioè due nodi appartengono alla stessa componente connessa se e solo se hanno la stessa etichetta.

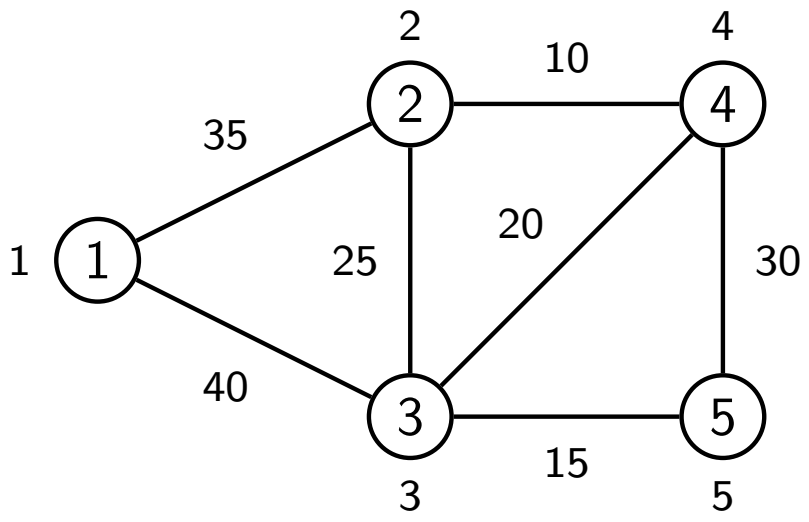
Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Algoritmo di Kruskal

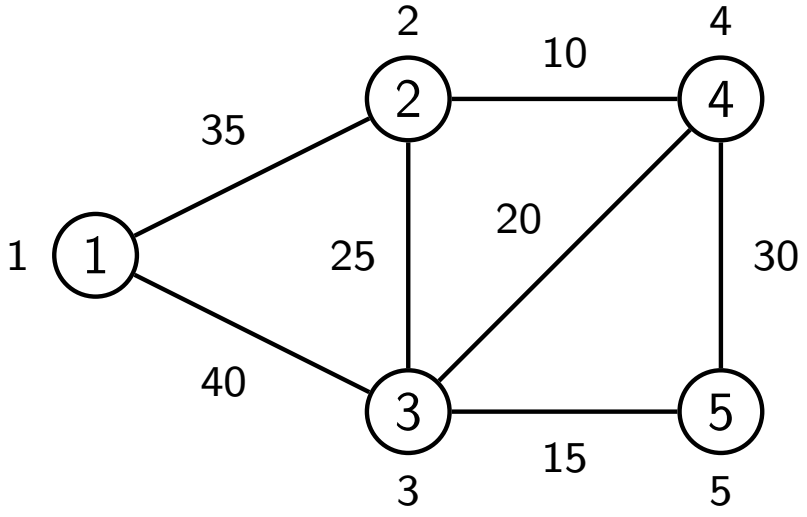
Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.
$T = \emptyset$,

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

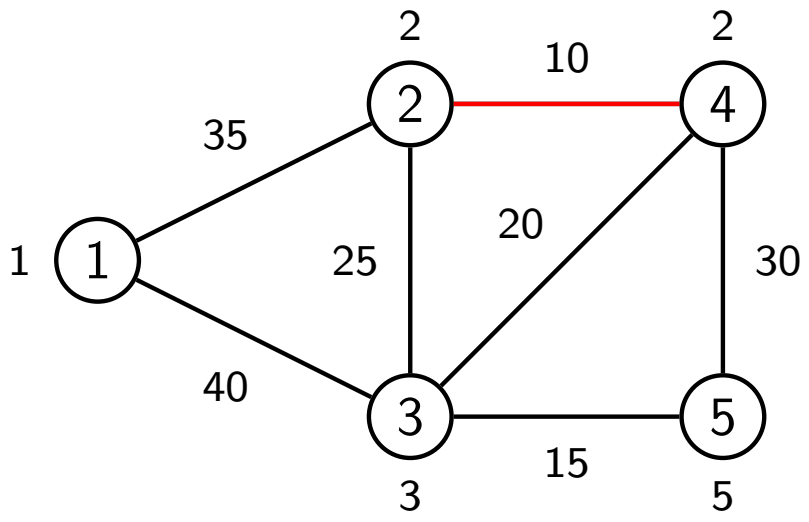


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$, analizzo $(2,4)$: $d_2 \neq d_4$? SI

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



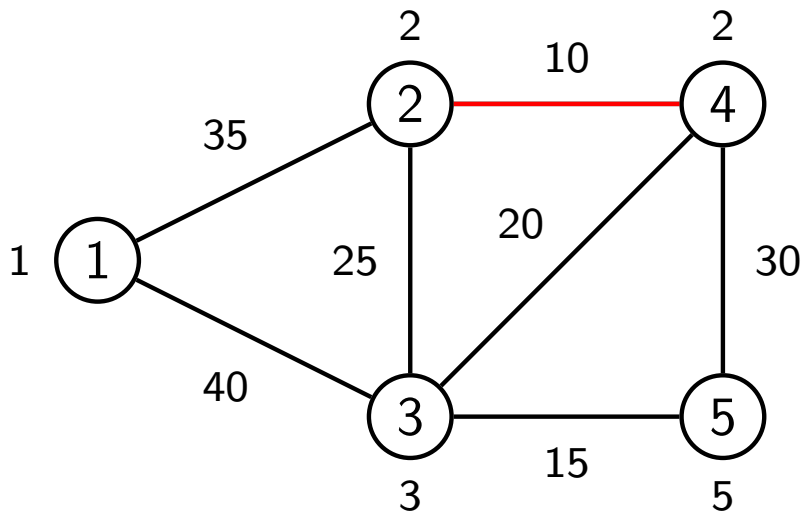
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$, analizzo $(2,4)$: $d_2 \neq d_4$? SI

$T = \{(2,4)\}$,

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



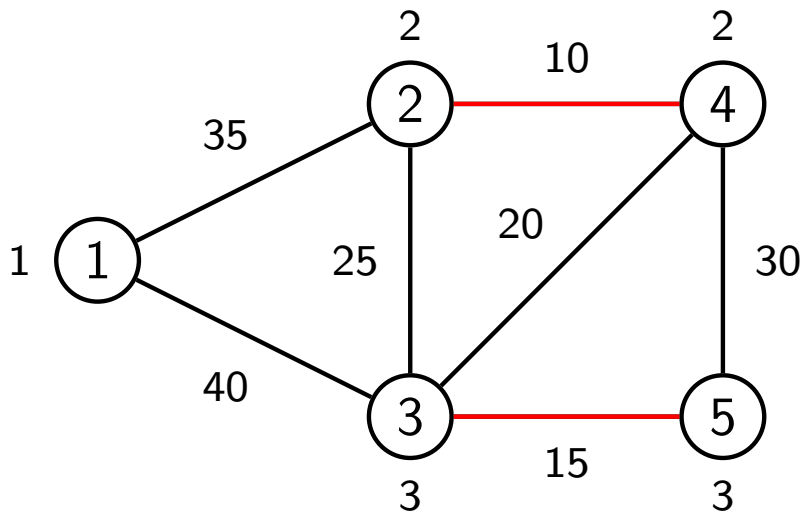
Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$, analizzo $(2,4)$: $d_2 \neq d_4$? SI

$T = \{(2,4)\}$, analizzo $(3,5)$: $d_3 \neq d_5$? SI

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

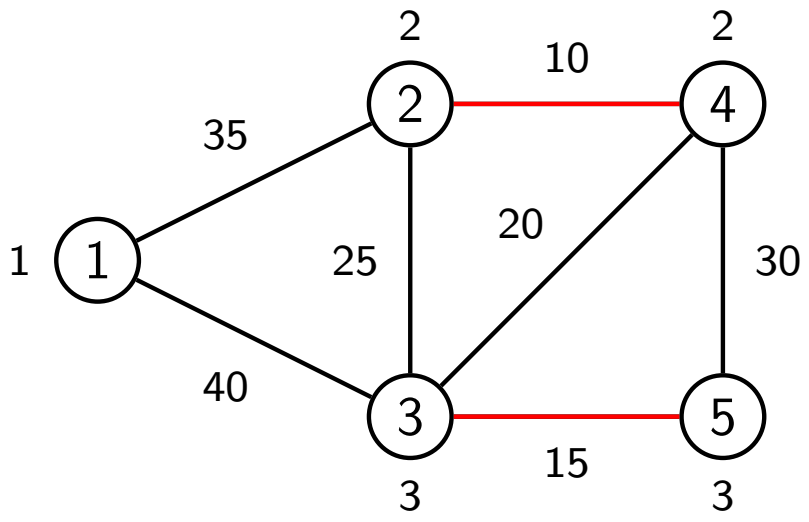
$T = \emptyset$, analizzo $(2,4)$: $d_2 \neq d_4$? SI

$T = \{(2,4)\}$, analizzo $(3,5)$: $d_3 \neq d_5$? SI

$T = \{(2,4),(3,5)\}$,

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

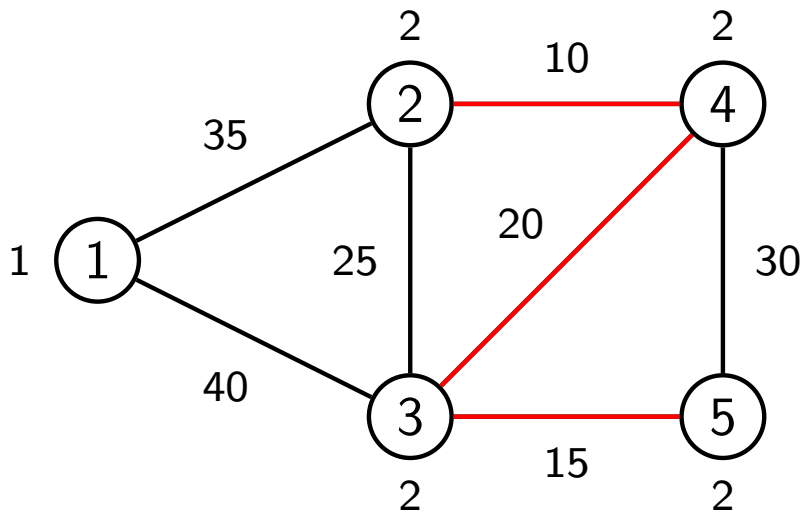


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

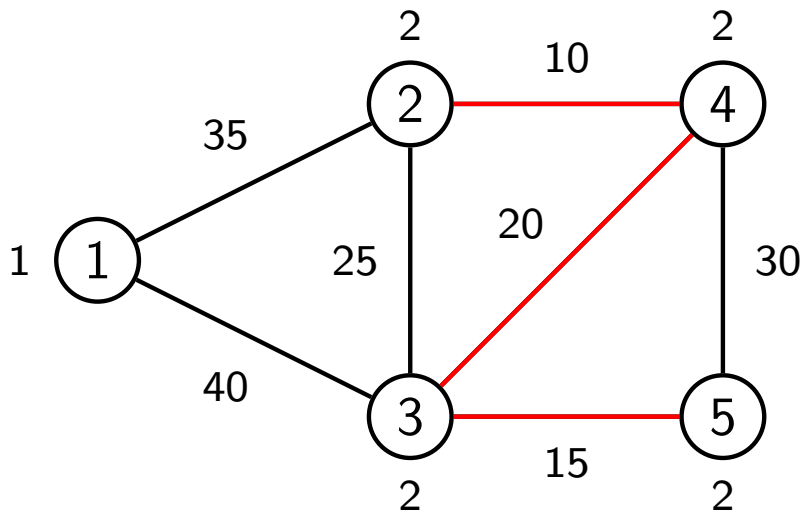


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

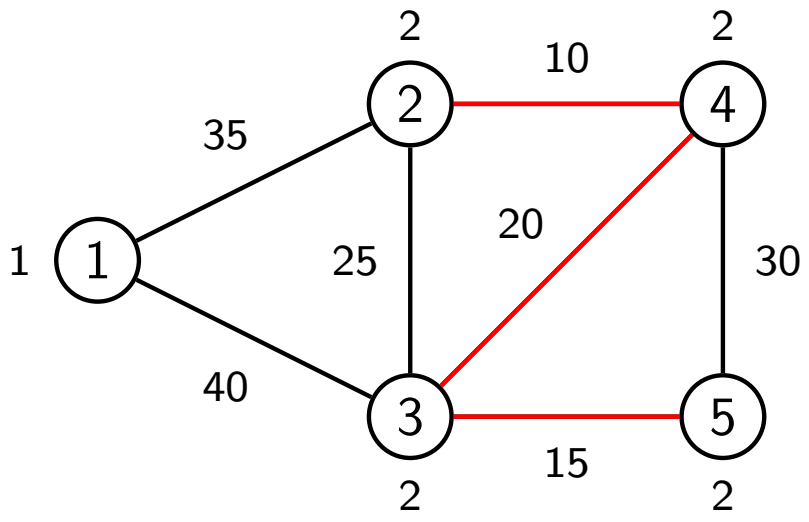


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

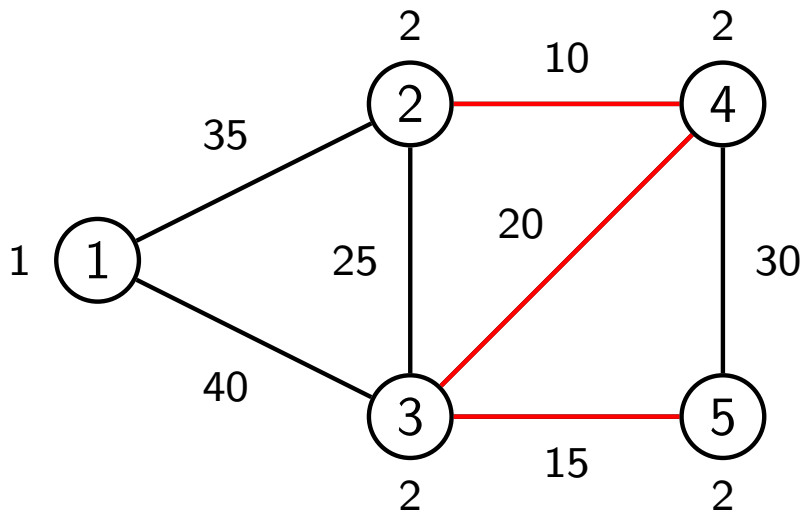


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

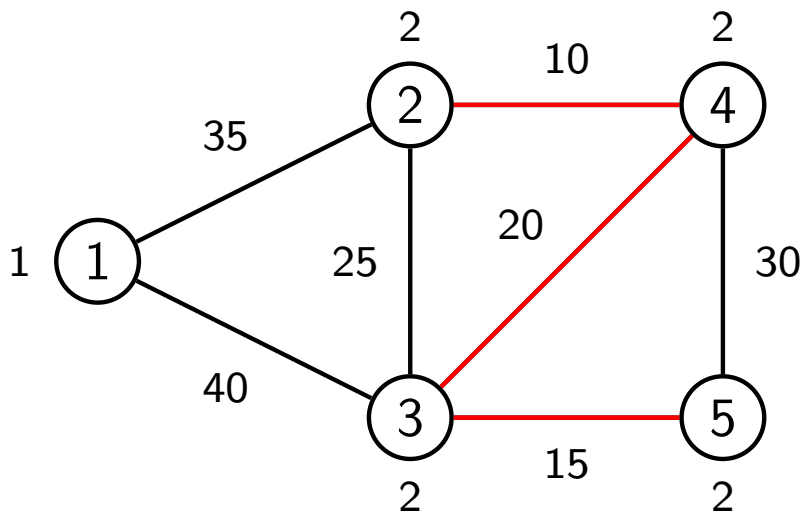


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(4,5)$: $d_4 \neq d_5$? NO

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

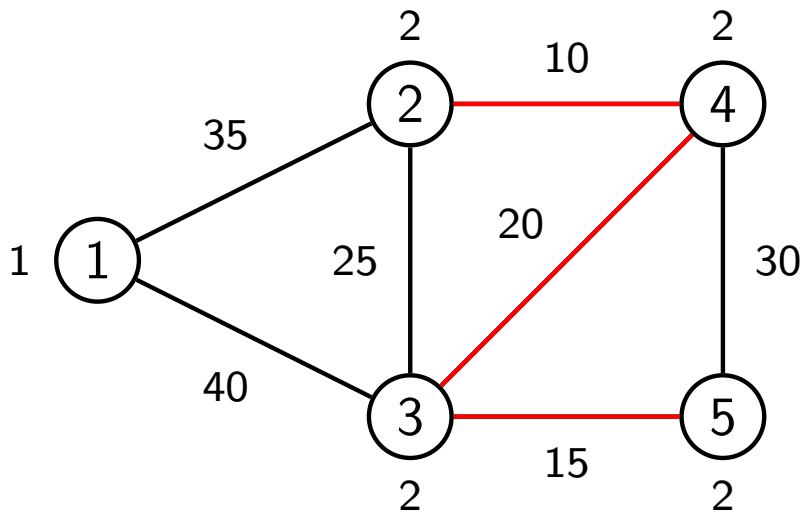


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(4,5)$: $d_4 \neq d_5$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.

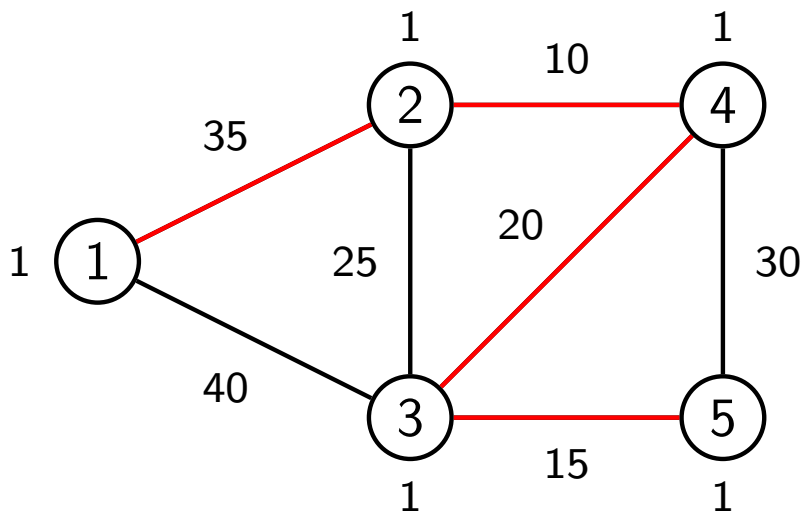


Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(4,5)$: $d_4 \neq d_5$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(1,2)$: $d_1 \neq d_2$? SI

Algoritmo di Kruskal

Esempio 3 (segue). Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Kruskal.



Archi in ordine crescente di costo: $\{(2,4),(3,5),(3,4),(2,3),(4,5),(1,2),(1,3)\}$.

$T = \emptyset$,	analizzo $(2,4)$: $d_2 \neq d_4$? SI
$T = \{(2,4)\}$,	analizzo $(3,5)$: $d_3 \neq d_5$? SI
$T = \{(2,4),(3,5)\}$,	analizzo $(3,4)$: $d_3 \neq d_4$? SI
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(2,3)$: $d_2 \neq d_3$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(4,5)$: $d_4 \neq d_5$? NO
$T = \{(2,4),(3,5),(3,4)\}$,	analizzo $(1,2)$: $d_1 \neq d_2$? SI
$T = \{(2,4),(3,5),(3,4),(1,2)\}$	STOP

Algoritmo di Kruskal

Teorema

L'algoritmo di Kruskal trova un albero di copertura di costo minimo in tempo polinomiale.

Dimostrazione

Segue dalla condizione di ottimalità basata sui cicli.

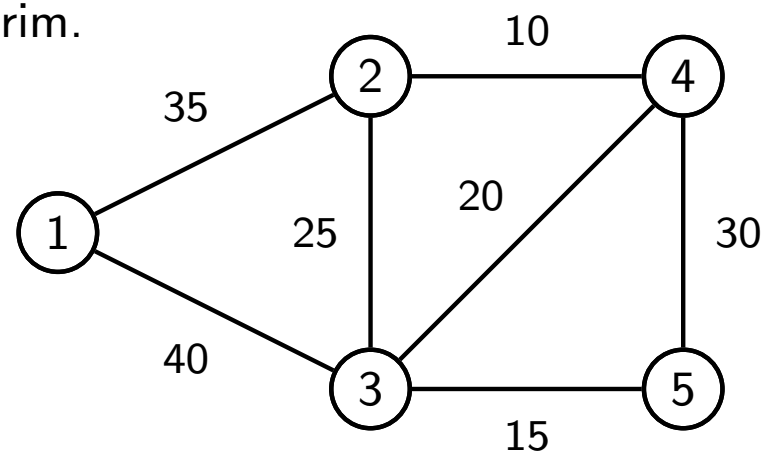
Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

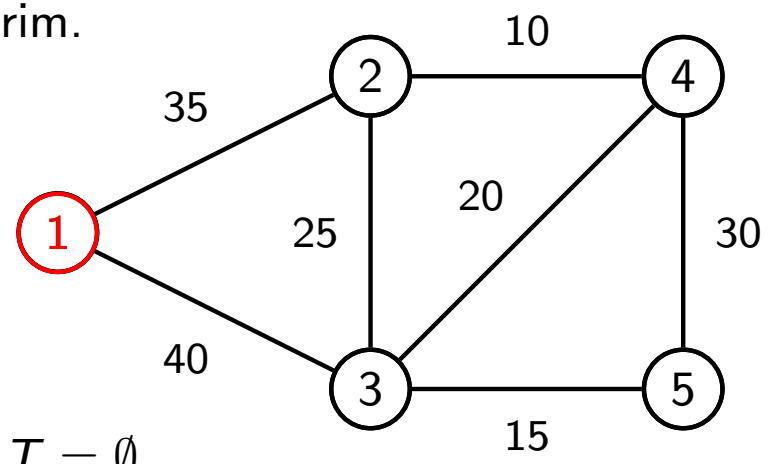
Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.



Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

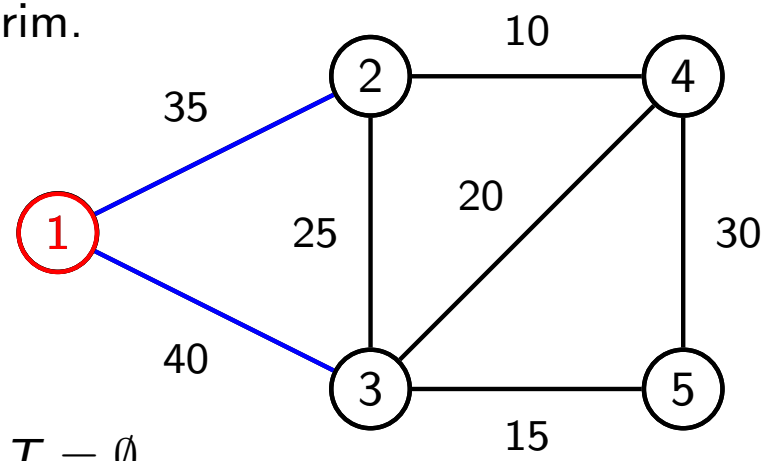


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

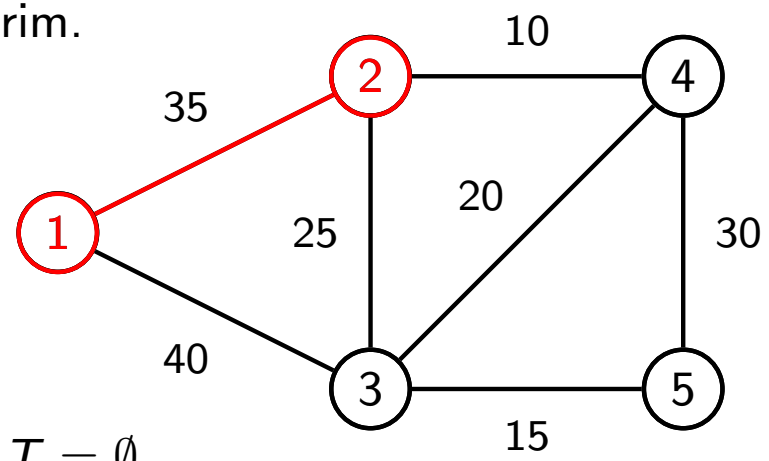


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$,

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

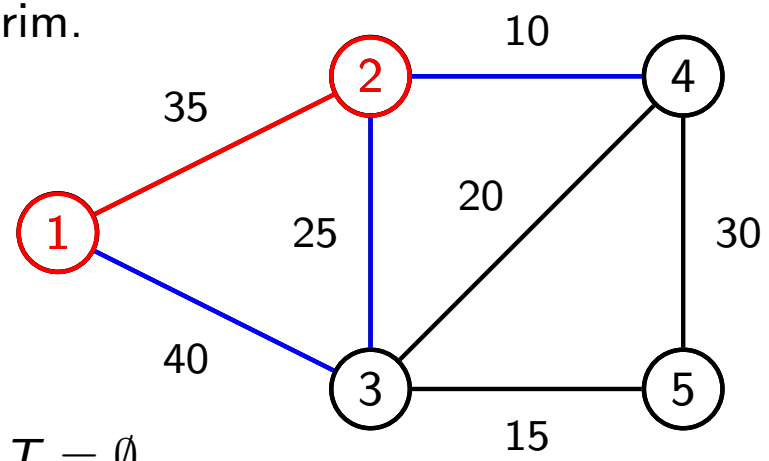


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

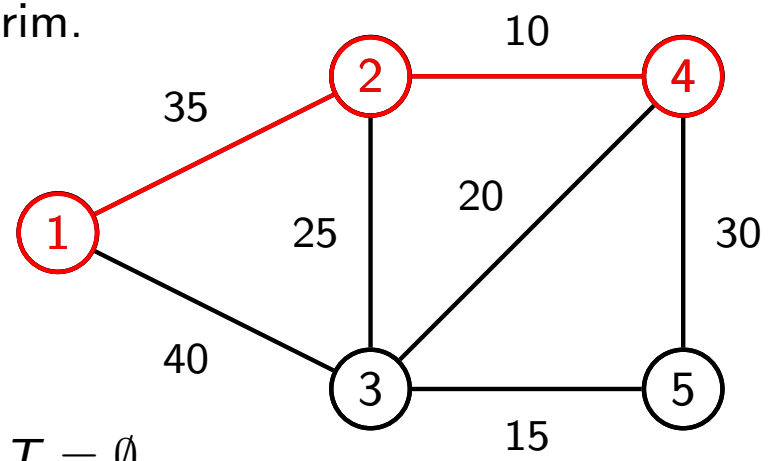


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$,

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

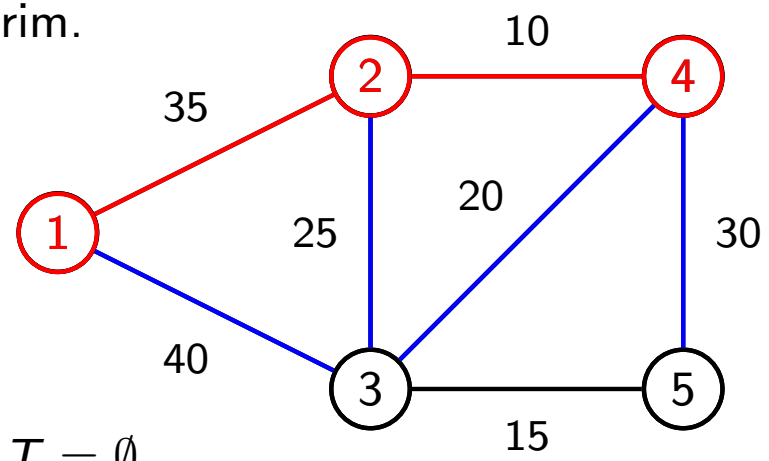


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$, $T = \{(1, 2), (2, 4)\}$, $S = \{1, 2, 4\}$

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

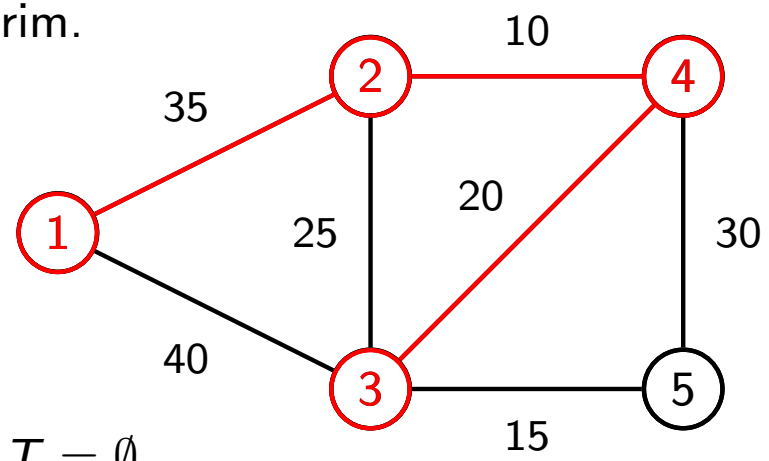


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$, $T = \{(1, 2), (2, 4)\}$, $S = \{1, 2, 4\}$
Taglio $(\{1, 2, 4\}, \{3, 5\})$, $(3, 4) = \arg\min\{c_{13}, c_{23}, c_{34}, c_{45}\}$,

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

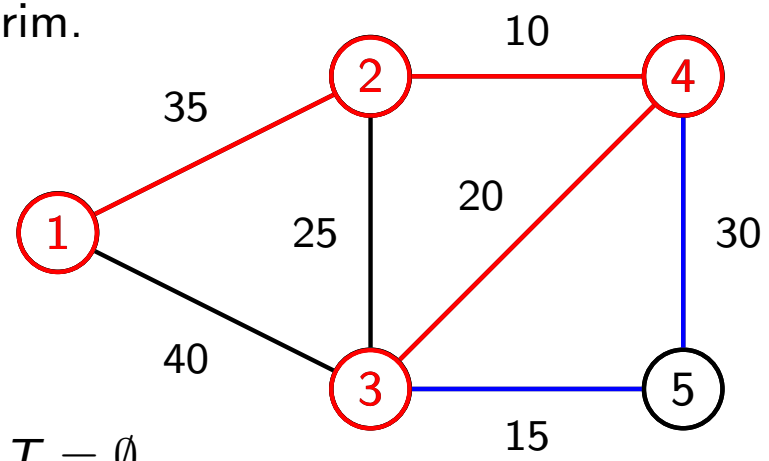


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$, $T = \{(1, 2), (2, 4)\}$, $S = \{1, 2, 4\}$
Taglio $(\{1, 2, 4\}, \{3, 5\})$, $(3, 4) = \arg\min\{c_{13}, c_{23}, c_{34}, c_{45}\}$, $T = \{(1, 2), (2, 4), (3, 4)\}$, $S = \{1, 2, 3, 4\}$

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.

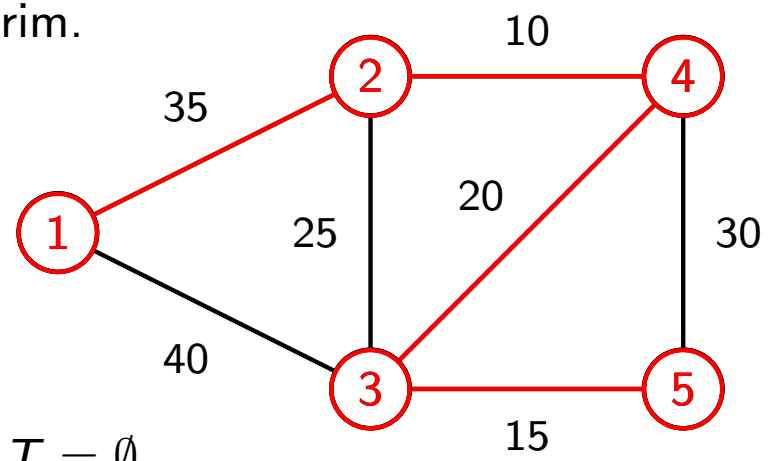


Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$, $T = \{(1, 2), (2, 4)\}$, $S = \{1, 2, 4\}$
Taglio $(\{1, 2, 4\}, \{3, 5\})$, $(3, 4) = \arg\min\{c_{13}, c_{23}, c_{34}, c_{45}\}$, $T = \{(1, 2), (2, 4), (3, 4)\}$, $S = \{1, 2, 3, 4\}$
Taglio $(\{1, 2, 3, 4\}, \{5\})$, $(3, 5) = \arg\min\{c_{35}, c_{45}\}$,

Algoritmo di Prim

0. Scegli un nodo $i \in N$, poni $S = \{i\}$ e $T = \emptyset$.
1. Se $|T| = n - 1$ allora stop
2. Trova un arco (u, v) di costo minimo tra gli archi del taglio $(S, N \setminus S)$. Inserisci (u, v) in T.
3. Inserisci in S l'estremo di (u, v) che non appartiene ad S e torna al passo 1.

Esempio 4. Trovare un albero di copertura di costo minimo sul seguente grafo utilizzando l'algoritmo di Prim.



Scegliamo $i = 1$, $S = \{1\}$, $T = \emptyset$.
Taglio $(\{1\}, \{2, 3, 4, 5\})$, $(1, 2) = \arg\min\{c_{12}, c_{13}\}$, $T = \{(1, 2)\}$, $S = \{1, 2\}$
Taglio $(\{1, 2\}, \{3, 4, 5\})$, $(2, 4) = \arg\min\{c_{13}, c_{23}, c_{24}\}$, $T = \{(1, 2), (2, 4)\}$, $S = \{1, 2, 4\}$
Taglio $(\{1, 2, 4\}, \{3, 5\})$, $(3, 4) = \arg\min\{c_{13}, c_{23}, c_{34}, c_{45}\}$, $T = \{(1, 2), (2, 4), (3, 4)\}$, $S = \{1, 2, 3, 4\}$
Taglio $(\{1, 2, 3, 4\}, \{5\})$, $(3, 5) = \arg\min\{c_{35}, c_{45}\}$, $T = \{(1, 2), (2, 4), (3, 4), (3, 5)\}$, $S = \{1, 2, 3, 4, 5\}$ STOP

Algoritmo di Prim

Teorema

L'algoritmo di Prim trova un albero di copertura di costo minimo in tempo polinomiale.

Dimostrazione

Segue dalla condizione di ottimalità basata sui tagli.

Algoritmo di Prim

Teorema

L'algoritmo di Prim trova un albero di copertura di costo minimo in tempo polinomiale.

Dimostrazione

Segue dalla condizione di ottimalità basata sui tagli.

Osservazione.

- Ad ogni iterazione dell'algoritmo di Kruskal l'insieme T non è necessariamente un albero.
- Ad ogni iterazione dell'algoritmo di Prim l'insieme T è un albero.

3 - Problema dei cammini minimi

Mauro Passacantando

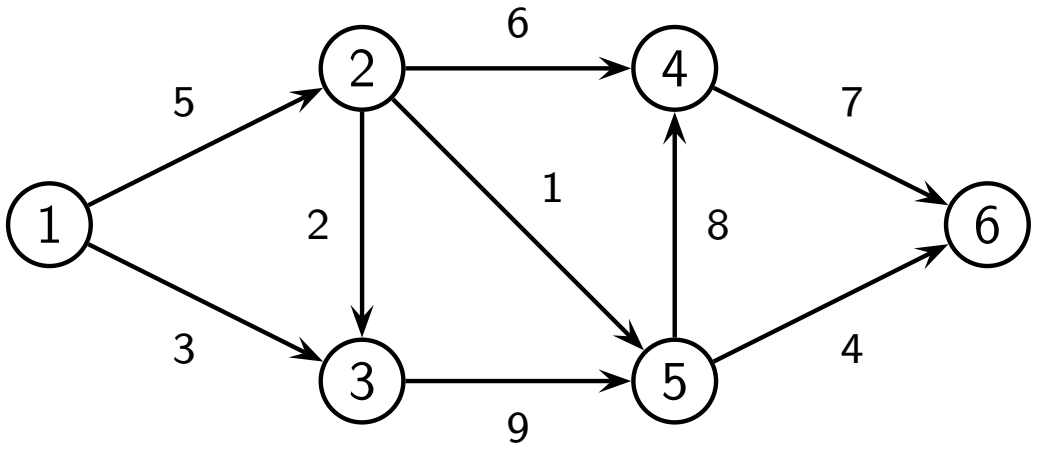
Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2021/22

Problema del cammino minimo – definizione

Sia (N, A) un grafo orientato in cui è definito un costo c_{ij} per ogni arco $(i, j) \in A$. Dati un nodo origine $s \in N$ e un nodo destinazione $t \in N$, trovare un cammino orientato da s a t di costo minimo, dove il costo di un cammino è definito come la somma dei costi degli archi da cui è formato.

Esempio 1. Trovare un cammino di costo minimo dal nodo 1 al nodo 6 sul seguente grafo (sugli archi sono riportati i costi).



Problema del cammino minimo – modello

Variabili: per ogni $(i,j) \in A$, definiamo x_{ij} = numero di volte che si attraversa l'arco (i,j) [flusso sull'arco (i,j)].

Funzione obiettivo: $\displaystyle\sum_{(i,j)\in A} c_{ij}x_{ij}$

Vincoli:

$$\sum_{(i,s)\in A} x_{is} - \sum_{(s,j)\in A} x_{sj} = -1 \qquad \text{(1 unità di flusso deve uscire da } s\text{)}$$

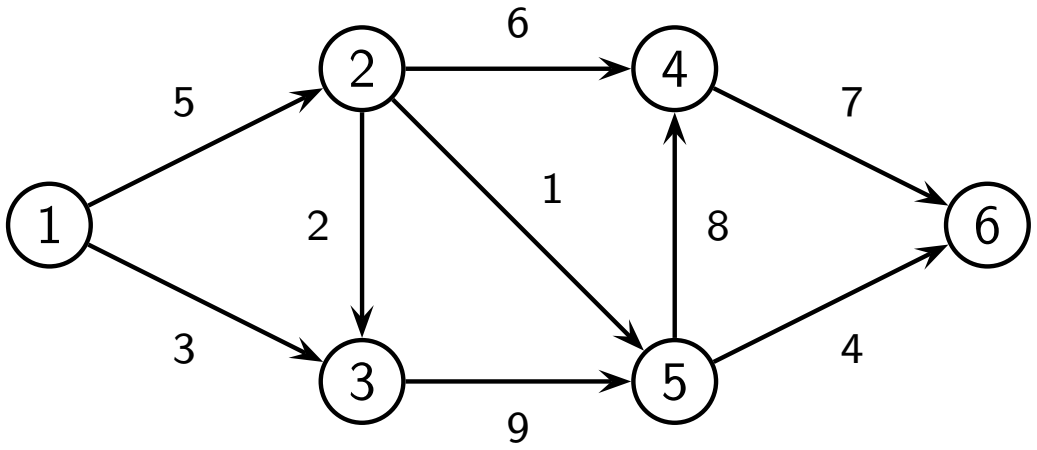
$$\sum_{(i,t)\in A} x_{it} - \sum_{(t,j)\in A} x_{tj} = 1 \qquad \text{(1 unità di flusso deve entrare in } t\text{)}$$

$$\sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = 0 \qquad \forall\ k \in N \setminus \{s,t\}$$

(i nodi diversi da s e t sono nodi di transito)

Problema del cammino minimo – modello

Esempio 1. Trovare un cammino di costo minimo dal nodo 1 al nodo 6:



Modello:

$$
\begin{cases}
\min \displaystyle\sum_{(i,j)\in A} c_{ij}x_{ij} & \\
-x_{12} - x_{13} = -1 & \text{(nodo 1)} \\
x_{46} + x_{56} = 1 & \text{(nodo 6)} \\
x_{12} - x_{23} - x_{24} - x_{25} = 0 & \text{(nodo 2)} \\
x_{13} + x_{23} - x_{35} = 0 & \text{(nodo 3)} \\
x_{24} + x_{54} - x_{46} = 0 & \text{(nodo 4)} \\
x_{25} + x_{35} - x_{54} - x_{56} = 0 & \text{(nodo 5)} \\
x_{ij} \geq 0, \quad x_{ij} \in \mathbb{Z} &
\end{cases}
$$

Problema del cammino minimo – modello

Il modello può essere scritto in modo più compatto utilizzando la matrice di incidenza del grafo.

La matrice di incidenza E di un grafo orientato (N, A) ha dimensione $|N| \times |A|$, cioè ha una riga per ogni nodo $k \in N$ ed una colonna per ogni arco $(i, j) \in A$, ed è così definita:

$$E_{k,(i,j)} = \begin{cases} -1 & \text{se } k = i, \\ 1 & \text{se } k = j, \\ 0 & \text{altrimenti.} \end{cases}$$

Il modello può essere scritto nella forma compatta

$$\begin{cases} \min \ c^{\mathsf{T}} x \\ Ex = b \\ x \geq 0, \quad x \in \mathbb{Z}^{|A|} \end{cases}$$

dove c è il vettore dei costi, x è il vettore dei flussi e b il vettore dei bilanci ai nodi

$$b_k = \begin{cases} -1 & \text{se } k = s, \\ 1 & \text{se } k = t, \\ 0 & \text{altrimenti.} \end{cases}$$

Problema dei cammini minimi – modello

Consideriamo ora il problema più generale di trovare un cammino di costo minimo da un nodo $r \in N$ a tutti gli altri nodi.

[Possiamo sempre supporre che esista un cammino da r ad ogni altro nodo $i \neq r$, eventualmente aggiungendo al grafo un arco fittizio (r, i) di costo $M > (n-1) \max_{(i,j) \in A} c_{ij}$]

Come deve essere modificato il modello di ottimizzazione?

Problema dei cammini minimi – modello

Consideriamo ora il problema più generale di trovare un cammino di costo minimo da un nodo $r \in N$ a tutti gli altri nodi.

[Possiamo sempre supporre che esista un cammino da r ad ogni altro nodo $i \neq r$, eventualmente aggiungendo al grafo un arco fittizio (r, i) di costo $M > (n-1) \max_{(i,j) \in A} c_{ij}$]

Come deve essere modificato il modello di ottimizzazione?
Basta modificare il vettore b dei bilanci ponendo

$$b_i = \begin{cases} -(n-1) & \text{se } i = r, \\ 1 & \text{se } i \neq r, \end{cases}$$

e considerare il modello

$$\begin{cases} \min \ c^{\mathsf{T}} x \\ Ex = b \\ x \geq 0, \quad x \in \mathbb{Z}^{|A|} \end{cases}$$

Esistenza di soluzioni ottime

Teorema
Il problema dei cammini minimi ammette (almeno) una soluzione ottima se e solo se non esistono cicli orientati di costo negativo.

[In seguito vedremo in modo algoritmico come fare a verificare se un grafo contiene un ciclo orientato di costo negativo (vedi algoritmo di Bellman-Ford)]

Se non esistono cicli orientati di costo negativo, allora esiste un insieme di cammini minimi da r ad ogni altro nodo che forma un albero di copertura radicato in r e orientato.

Condizioni di ottimalità

Come riconoscere una soluzione ottima?

Se T è un albero di copertura radicato in r e orientato, allora esiste un unico cammino da r ad ogni nodo $i \neq r$.
Indichiamo con π_i il costo di tale cammino. [π_i è chiamato potenziale del nodo i]
Per calcolare i potenziali π basta fare una visita dell'albero T e porre $\pi_j = \pi_i + c_{ij}$ se $\text{pred}(j) = i$.

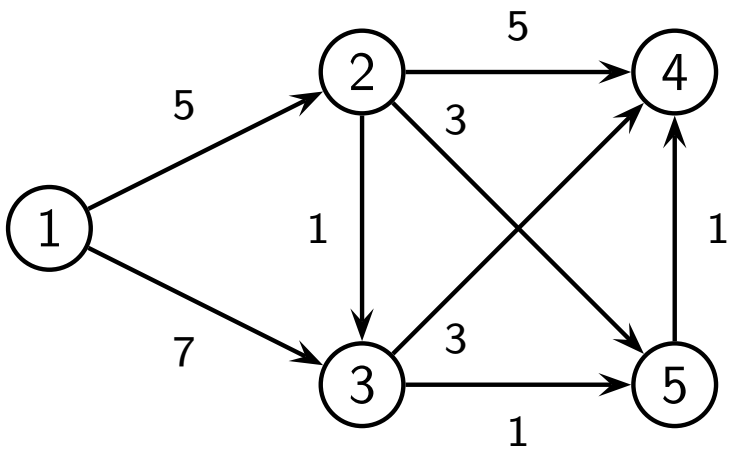
Teorema (condizioni di Bellman)

Sia T un albero di copertura radicato in r e orientato, π il corrispondente vettore dei potenziali dei nodi.

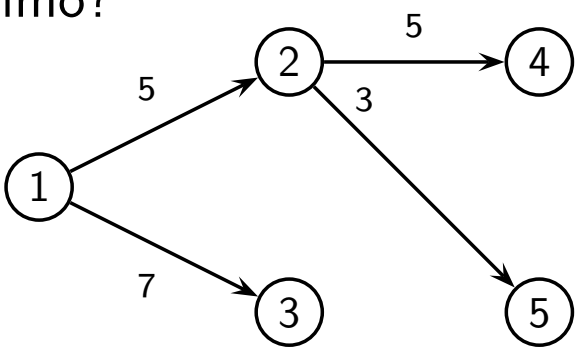
$$T \text{ è un albero dei cammini minimi} \quad \text{se e solo se} \quad \pi_j \leq \pi_i + c_{ij}, \quad \forall\, (i,j) \notin T.$$

Condizioni di ottimalità

Esempio 2. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



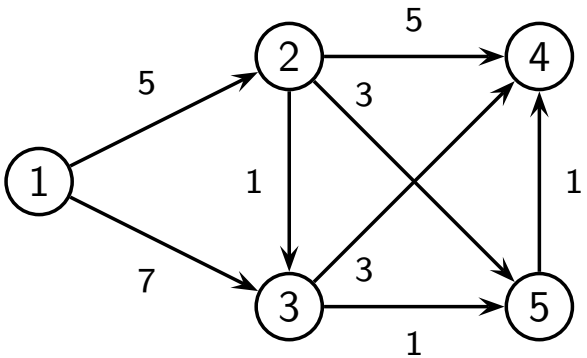
Il seguente albero è ottimo?



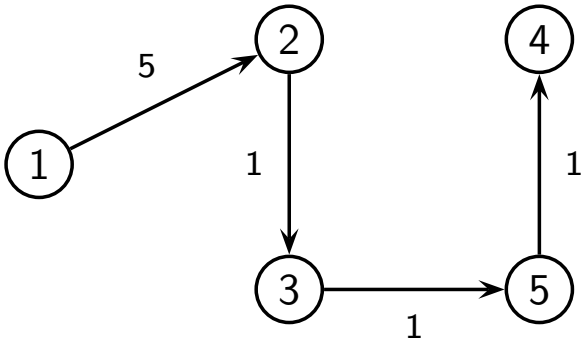
Calcoliamo i potenziali dei nodi: $\pi = (0, 5, 7, 10, 8)$
e controlliamo le condizioni di Bellman:
arco $(2, 3)$: $7 = \pi_3 \leq \pi_2 + c_{23} = 5 + 1$? NO, quindi l'albero non è ottimo.

Condizioni di ottimalità

Esempio 3. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



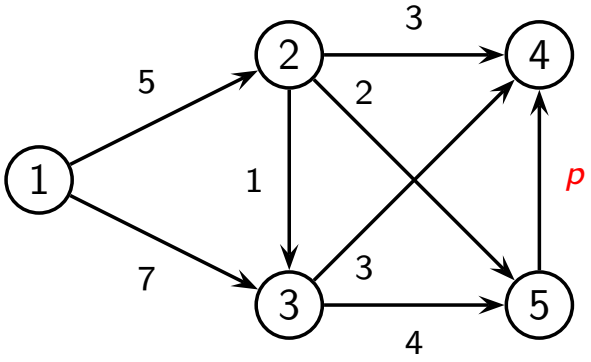
Il seguente albero è ottimo?



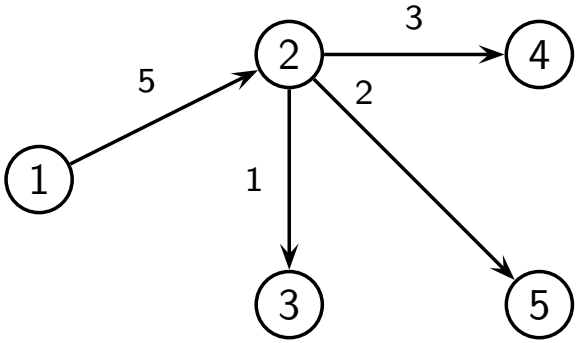
I potenziali dei nodi sono $\pi = (0, 5, 6, 8, 7)$ Controlliamo le condizioni di Bellman:
arco $(1, 3)$: $6 = \pi_3 \leq \pi_1 + c_{13} = 0 + 7$? SI.
arco $(2, 4)$: $8 = \pi_4 \leq \pi_2 + c_{24} = 5 + 5$? SI.
arco $(2, 5)$: $7 = \pi_5 \leq \pi_2 + c_{25} = 5 + 3$? SI.
arco $(3, 4)$: $8 = \pi_4 \leq \pi_3 + c_{34} = 6 + 3$? SI. Pertanto l'albero è ottimo.

Condizioni di ottimalità

Esempio 4. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



Per quali valori del parametro p il seguente albero è ottimo?



I potenziali dei nodi sono $\pi = (0, 5, 6, 8, 7)$ Condizioni di Bellman:
arco $(1, 3)$: $6 = \pi_3 \leq \pi_1 + c_{13} = 0 + 7$ sempre
arco $(3, 4)$: $8 = \pi_4 \leq \pi_3 + c_{34} = 6 + 3$ sempre
arco $(3, 5)$: $7 = \pi_5 \leq \pi_3 + c_{35} = 6 + 4$ sempre
arco $(5, 4)$: $8 = \pi_4 \leq \pi_5 + c_{54} = 7 + p$ per $p \geq 1$.
Pertanto l'albero è ottimo se e solo se $p \geq 1$.

Unicità della soluzione ottima

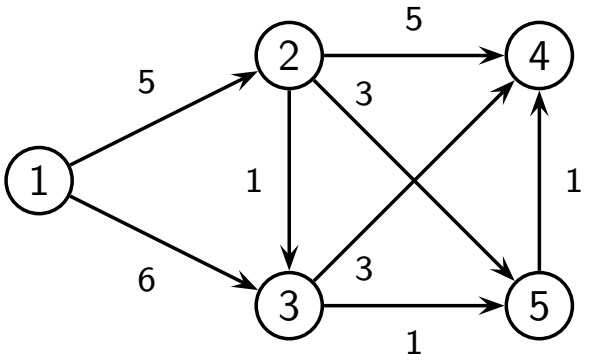
Teorema

Supponiamo che nel grafo non esistano cicli orientati di costo ≤ 0. Sia T un albero di copertura radicato in r e orientato, π il corrispondente vettore dei potenziali dei nodi. Allora

T è l'unico albero dei cammini minimi se e solo se $\pi_j < \pi_i + c_{ij}, \quad \forall\,(i,j) \notin T.$

Nell'esempio 3 le condizioni di Bellman valgono tutte con il segno $<$ quindi l'albero ottimo è unico.

Mentre nel grafo



esistono due alberi ottimi:

$T_1 = \{(1,2),(2,3),(3,5),(5,4)\}$ e $T_2 = \{(1,2),(1,3),(3,5),(5,4)\}$.

Algoritmo di Dijkstra

L'algoritmo di Dijkstra trova un albero ottimo nel caso in cui $c_{ij} \geq 0$ per ogni $(i, j) \in A$.

Ad ogni iterazione l'algoritmo mantiene:

- un albero di copertura radicato in r e orientato (memorizzato in un vettore p di predecessori)
- un vettore π di potenziali dei nodi corrispondente all'albero memorizzato in p
- un insieme U di nodi tale che gli archi che potrebbero violare le condizioni di Bellman devono essere uscenti da nodi di U.

Algoritmo di Dijkstra

Algoritmo

0. Inizializza

$$p_i = \begin{cases} 0 & \text{se } i = r \\ -1 & \text{se } i \neq r \end{cases} \qquad \pi_i = \begin{cases} 0 & \text{se } i = r \\ +\infty & \text{se } i \neq r \end{cases} \qquad U = N$$

(l'albero inizialmente è costituito solo da archi fittizi da r agli altri nodi)

1. Se $U = \emptyset$ allora stop

2. Seleziona un nodo $u \in U$ con potenziale minimo: $u = \arg\min_{i \in U} \pi_i$

3. (Controlla le condizioni di Bellman sugli archi uscenti da u)
Per ogni arco $(u, v) \in A$:
se $\pi_v > \pi_u + c_{uv}$ allora $p_v = u$, $\pi_v = \pi_u + c_{uv}$

4. $U = U \setminus \{u\}$ e torna al passo 1.

Algoritmo di Dijkstra

Teorema

Se $c_{ij} \geq 0$ per ogni $(i, j) \in A$, allora l'algoritmo di Dijkstra trova un albero dei cammini minimi dopo $|N|$ iterazioni.

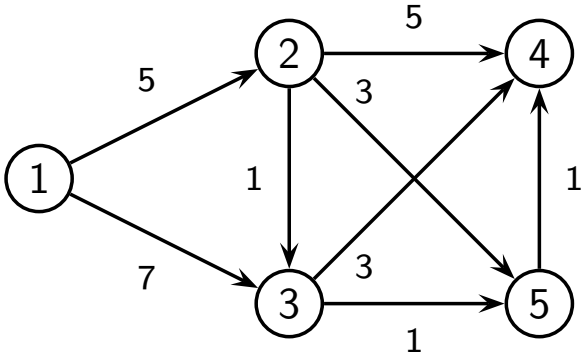
Dimostrazione (sketch)

- durante l'esecuzione dell'algoritmo π è il vettore dei potenziali dei nodi associato all'albero memorizzato nel vettore p
- il potenziale di ogni nodo non può mai aumentare durante l'esecuzione dell'algoritmo
- quando un nodo u viene estratto da U, il suo potenziale diventa definitivo e gli archi uscenti da u soddisfano le condizioni di Bellman fino alla fine dell'algoritmo

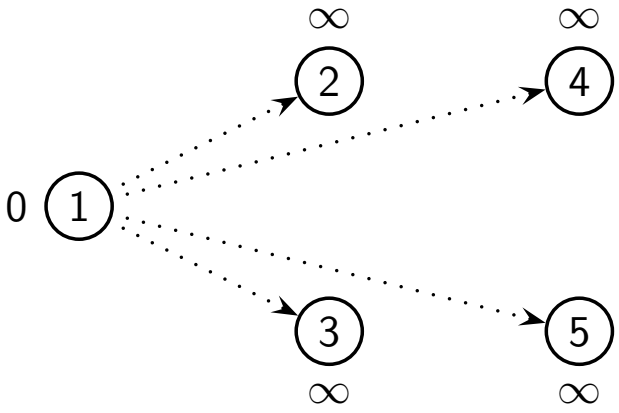
$\square$

Algoritmo di Dijkstra

Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



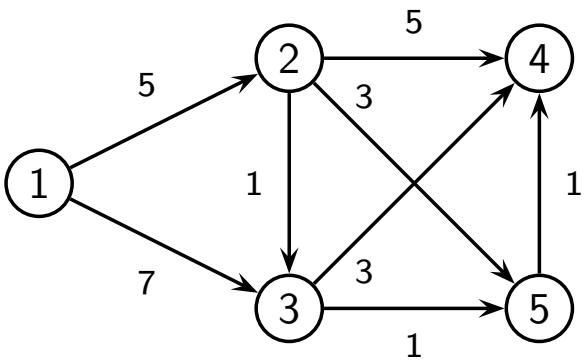
Iterazione 0.



$$U = \{1, 2, 3, 4, 5\}$$

Algoritmo di Dijkstra

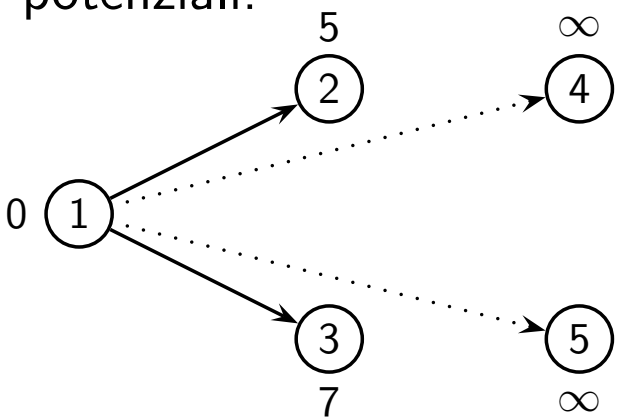
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 1. Estrai nodo 1 da U ed esamina gli archi uscenti da 1:

$$\begin{cases} \infty = \pi_2 > \pi_1 + c_{12} = 5 \Rightarrow p_2 = 1, \pi_2 = 5 \\ \infty = \pi_3 > \pi_1 + c_{13} = 7 \Rightarrow p_3 = 1, \pi_3 = 7 \end{cases}$$

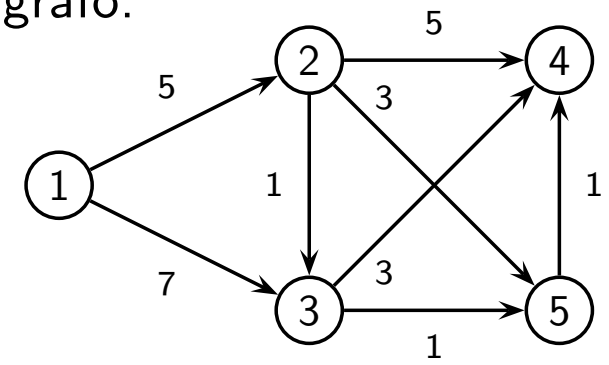
Aggiorno l'albero ed i potenziali:



$$U = \{2, 3, 4, 5\}$$

Algoritmo di Dijkstra

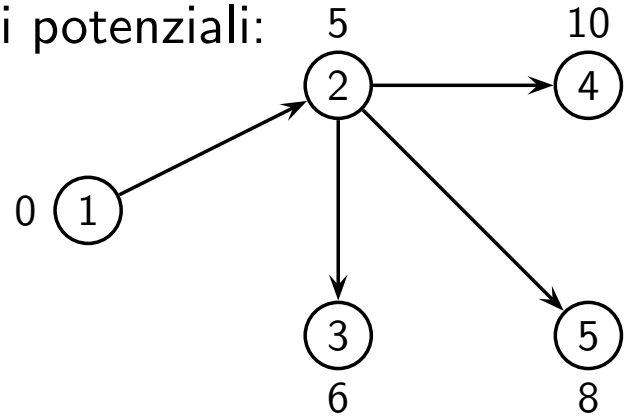
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 2. Estrai nodo 2 da U ed esamina gli archi uscenti da 2:

$$\begin{cases} 7 = \pi_3 > \pi_2 + c_{23} = 6 \Rightarrow p_3 = 2, \pi_3 = 6 \\ \infty = \pi_4 > \pi_2 + c_{24} = 10 \Rightarrow p_4 = 2, \pi_4 = 10 \\ \infty = \pi_5 > \pi_2 + c_{25} = 8 \Rightarrow p_5 = 2, \pi_5 = 8 \end{cases}$$

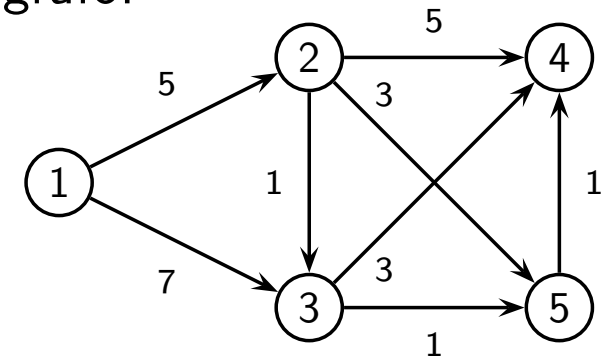
Aggiorno l'albero ed i potenziali:



$U = \{3, 4, 5\}$

Algoritmo di Dijkstra

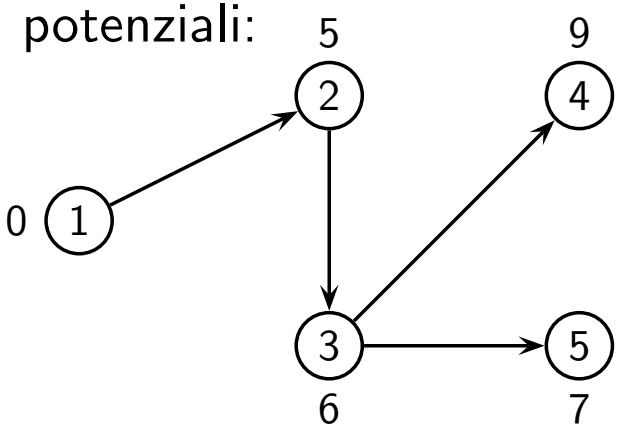
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 3. Estrai nodo 3 da U ed esamina gli archi uscenti da 3:

$$\begin{cases} 10 = \pi_4 > \pi_3 + c_{34} = 9 \Rightarrow p_4 = 3, \pi_4 = 9 \\ 8 = \pi_5 > \pi_3 + c_{35} = 7 \Rightarrow p_5 = 3, \pi_5 = 7 \end{cases}$$

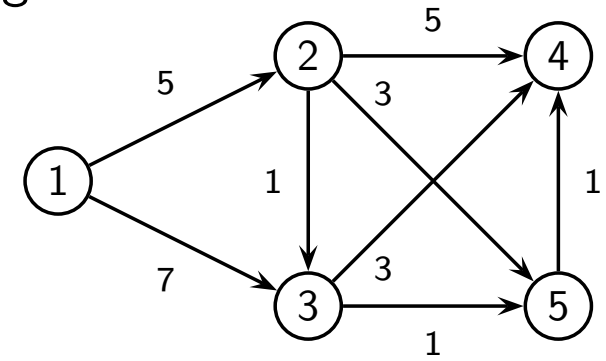
Aggiorno l'albero ed i potenziali:



$U = \{4, 5\}$

Algoritmo di Dijkstra

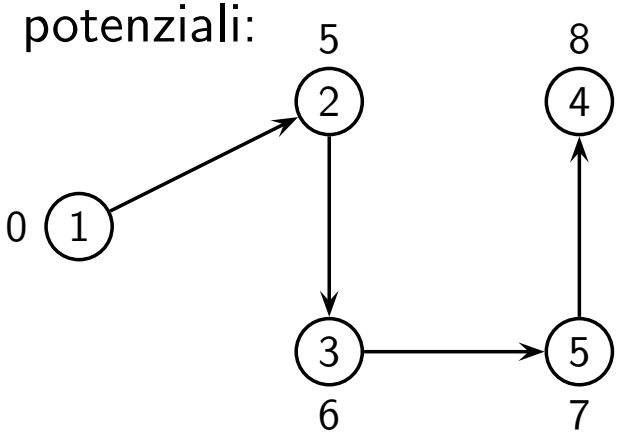
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 4. Estrai nodo 5 da U ed esamina gli archi uscenti da 5:

$$9 = \pi_4 > \pi_5 + c_{54} = 8 \Rightarrow p_4 = 5, \pi_4 = 8$$

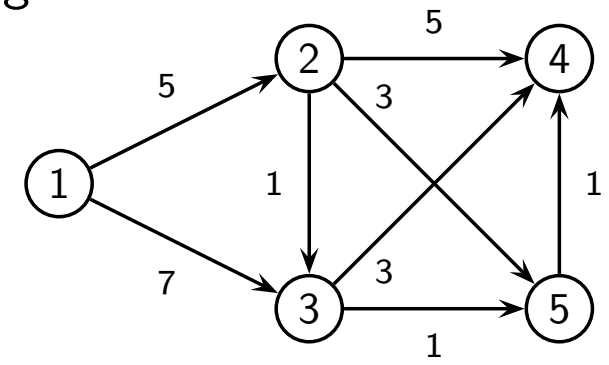
Aggiorno l'albero ed i potenziali:



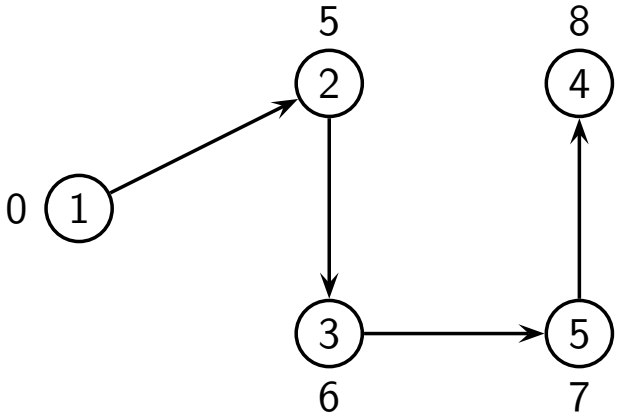
$U = \{4\}$

Algoritmo di Dijkstra

Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



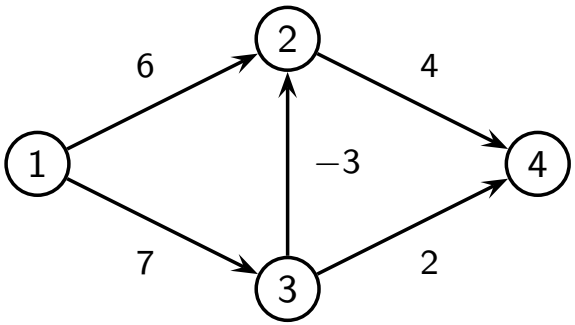
Iterazione 5. Estrai nodo 4 da U, non ci sono archi uscenti da 4, stop.
Un albero dei cammini minimi è



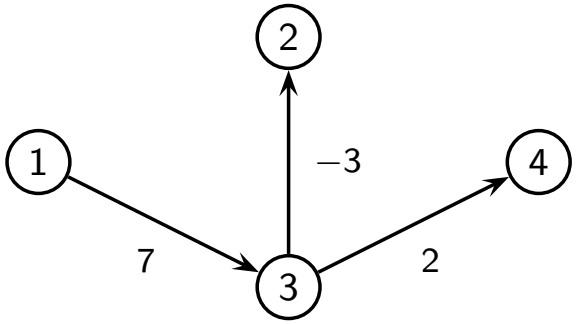
Algoritmo di Dijkstra

Se nel grafo esistono archi di costo negativo, l'algoritmo di Dijkstra non trova necessariamente un albero dei cammini minimi.

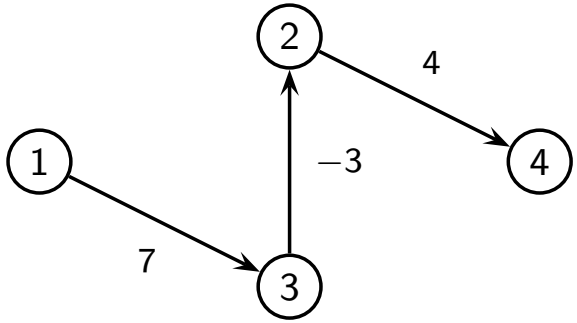
Esempio 6. Dato il grafo



l'algoritmo di Dijkstra trova l'albero



che non è ottimo, perché l'albero dei cammini minimi è



Algoritmo di Bellman-Ford

L'algoritmo di Bellman-Ford trova un albero ottimo (se esiste) anche nel caso in cui nel grafo ci siano archi di costo negativo.

Ad ogni iterazione k l'algoritmo mantiene:

- un albero di copertura radicato in r e orientato (memorizzato in un vettore p di predecessori)
- un vettore π^k di etichette associate ai nodi (che non è necessariamente uguale al vettore dei potenziali dei nodi associato all'albero)

Algoritmo di Bellman-Ford

Algoritmo

0. Inizializza $\quad p_i = \begin{cases} 0 & \text{se } i = r \\ -1 & \text{se } i \neq r \end{cases} \qquad \pi_i^0 = \begin{cases} 0 & \text{se } i = r \\ +\infty & \text{se } i \neq r \end{cases} \qquad k = 1$

(l'albero inizialmente è costituito solo da archi fittizi da r agli altri nodi)

1. (Aggiorniamo le etichette di tutti i nodi)
Per ogni nodo $j \in N$:

trova $u = \arg\min_{i \in BS(j)} \{\pi_i^{k-1} + c_{ij}\}$

$[BS(j) = \{i \in N : \text{ esiste un arco } (i,j) \in A\}$ è la stella entrante in $j]$

se $\pi_j^{k-1} > \pi_u^{k-1} + c_{uj}$ allora $p_j = u$, $\pi_j^k = \pi_u^{k-1} + c_{uj}$
altrimenti $\pi_j^k = \pi_j^{k-1}$

2. Se $\pi^k = \pi^{k-1}$ allora stop (p fornisce un albero ottimo)

3. Se $k = |N|$ allora stop (p fornisce un ciclo orientato di costo negativo)
altrimenti $k = k + 1$ e torna al passo 1.

Algoritmo di Bellman-Ford

Teorema

L'algoritmo di Bellman-Ford trova un albero dei cammini minimi oppure un ciclo orientato di costo negativo dopo al più $|N|$ iterazioni.

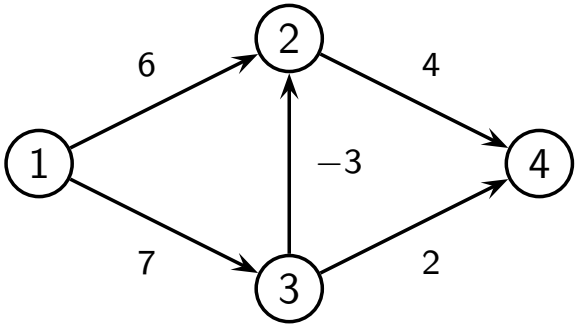
Dimostrazione (sketch)

- Per ogni nodo j l'etichetta π_j^k è uguale al costo del cammino minimo da r a j che contiene al più k archi.
- Se ad una iterazione $k \leq |N|$ le etichette dei nodi non sono state modificate ($\pi^k = \pi^{k-1}$), allora rimarranno costanti in tutte le iterazioni successive, quindi π_j^k è il costo del cammino minimo da r a j senza limitazioni sul numero di archi e p fornisce un albero ottimo.
- Se all'iterazione $|N|$ viene modificata l'etichetta di un nodo j, allora è stato trovato un nuovo cammino da r a j contenente $|N|$ archi (cioè passante due volte su uno stesso nodo) di costo inferiore rispetto a tutti i cammini semplici da r a j. Pertanto non può esistere un cammino minimo da r a j, mentre nel nuovo cammino trovato esiste un ciclo di costo negativo che può essere ricavato dal vettore p.

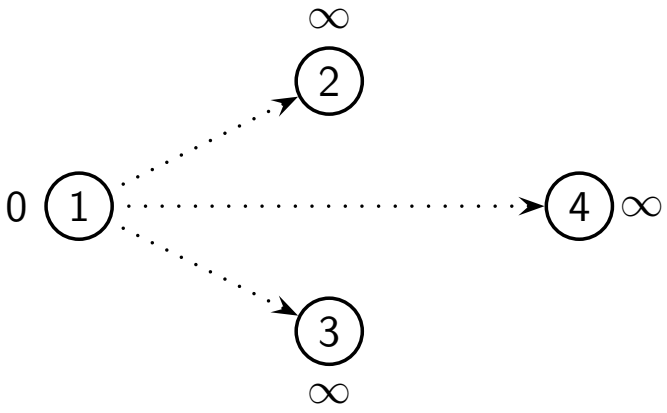
$\square$

Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:

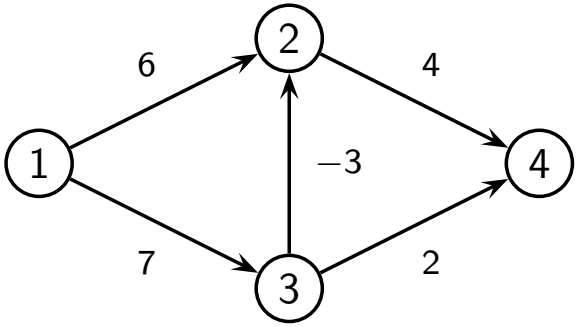


Iterazione 0.



Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione $k = 1$.

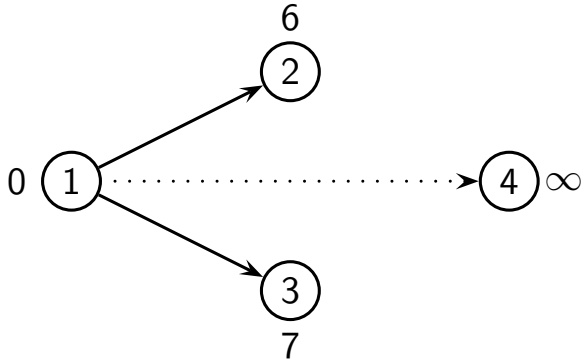
Nodo 1: non ci sono archi entranti in 1, $\pi_1^1 = 0$.

Nodo 2: $u = \arg\min\{\pi_1^0 + c_{12}, \pi_3^0 + c_{32}\} = 1$,

$\infty = \pi_2^0 > \pi_1^0 + c_{12} = 6 \Longrightarrow p_2 = 1$, $\pi_2^1 = 6$

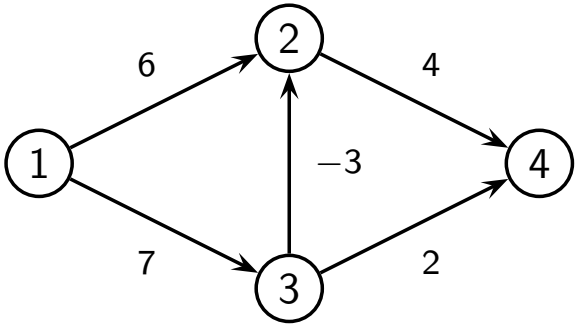
Nodo 3: $u = 1$, $\quad \infty = \pi_3^0 > \pi_1^0 + c_{13} = 7 \Longrightarrow p_3 = 1$, $\pi_3^1 = 7$

Nodo 4: $u = \arg\min\{\pi_2^0 + c_{24}, \pi_3^0 + c_{34}\} = 2$. $\quad \infty = \pi_4^0 = \pi_2^0 + c_{24} = \infty \Longrightarrow \pi_4^1 = \infty$

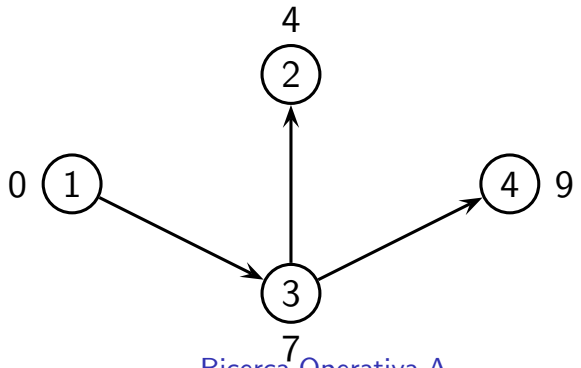


Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:

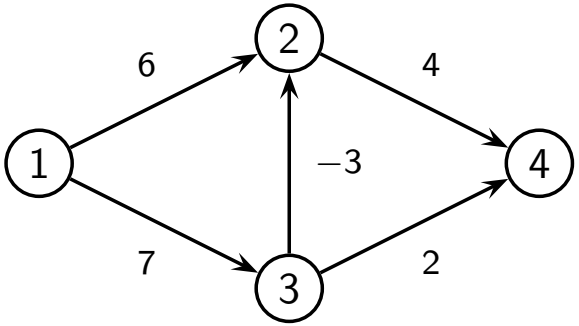


Iterazione $k = 2$.
Nodo 1: non ci sono archi entranti in 1, $\pi_1^2 = 0$
Nodo 2: $u = \arg\min\{\pi_1^1 + c_{12}, \pi_3^1 + c_{32}\} = 3, \quad 6 = \pi_2^1 > \pi_3^1 + c_{32} = 4 \Longrightarrow p_2 = 3$, $\pi_2^2 = 4$
Nodo 3: $u = 1, \quad 7 = \pi_3^1 = \pi_1^1 + c_{13} = 7 \Longrightarrow \pi_3^2 = 7$
Nodo 4: $u = \arg\min\{\pi_2^1 + c_{24}, \pi_3^1 + c_{34}\} = 3 \quad \infty = \pi_4^1 > \pi_3^1 + c_{34} = 9 \Longrightarrow p_4 = 3$, $\pi_4^2 = 9$

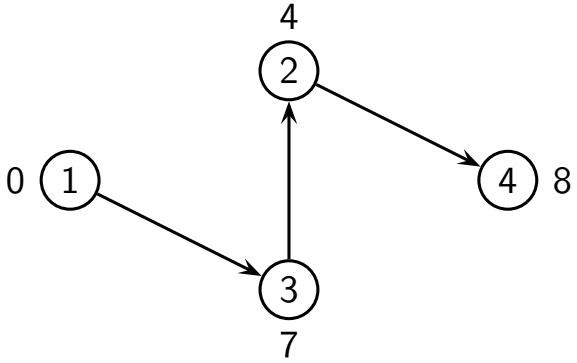


Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:

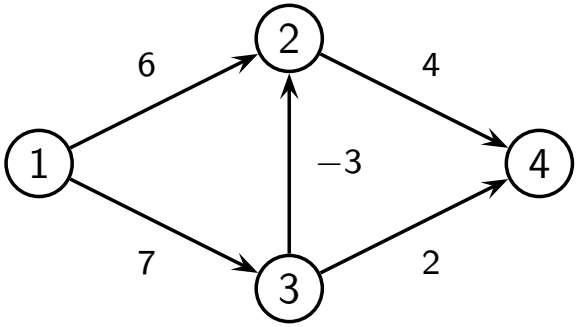


Iterazione $k = 3$.
Nodo 1: non ci sono archi entranti in 1, $\pi_1^3 = 0$
Nodo 2: $u = \arg\min\{\pi_1^2 + c_{12}, \pi_3^2 + c_{32}\} = 3 \quad 4 = \pi_2^2 = \pi_3^2 + c_{32} = 4 \Longrightarrow \pi_2^3 = 4$
Nodo 3: $u = 1, \quad 7 = \pi_3^2 = \pi_1^2 + c_{13} = 7 \Longrightarrow \pi_3^3 = 7$
Nodo 4: $u = \arg\min\{\pi_2^2 + c_{24}, \pi_3^2 + c_{34}\} = 2 \quad 9 = \pi_4^2 > \pi_2^2 + c_{24} = 8 \Longrightarrow p_4 = 2$, $\pi_4^3 = 8$

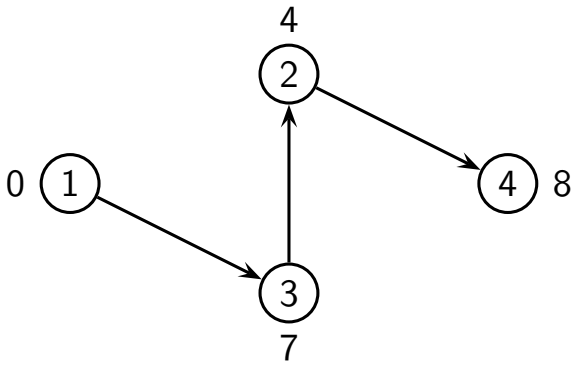


Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione $k = 4$.
Nodo 1: non ci sono archi entranti in 1, $\pi_1^4 = 0$
Nodo 2: $u = \arg\min\{\pi_1^3 + c_{12}, \pi_3^3 + c_{32}\} = 3 \quad 4 = \pi_2^3 = \pi_3^3 + c_{32} = 4 \Longrightarrow \pi_2^4 = 4$
Nodo 3: $u = 1 \quad 7 = \pi_3^3 = \pi_1^3 + c_{13} = 7 \Longrightarrow \pi_3^4 = 7$
Nodo 4: $u = \arg\min\{\pi_2^3 + c_{24}, \pi_3^3 + c_{34}\} = 2. \quad 8 = \pi_4^3 = \pi_2^3 + c_{24} = 8 \Longrightarrow \pi_4^4 = 8$
Poiché nessuna etichetta è cambiata, cioè $\pi^4 = \pi^3$, l'algoritmo si ferma e l'albero seguente è ottimo



Algoritmo di programmazione dinamica

Se il grafo (N, A) è aciclico, cioè non esistono cicli orientati, allora un albero dei cammini minimi di radice r si può trovare utilizzando un algoritmo di programmazione dinamica.

Prima di tutto, è necessario fare un ordinamento topologico dei nodi, cioè numerare i nodi in modo che

$$(i, j) \in A \implies i < j.$$

Tale ordinamento può essere effettuato con il seguente algoritmo:

1. Assegno 1 alla radice r, elimino dal grafo r e tutti gli archi uscenti da r. Pongo $k = 2$.
2. Assegno k ad un nodo i che non ha archi entranti.
3. Elimino il nodo i e tutti gli archi uscenti da i.
4. Se il grafo è vuoto, allora stop
   altrimenti $k = k + 1$ e torna al passo 2.

Algoritmo di programmazione dinamica

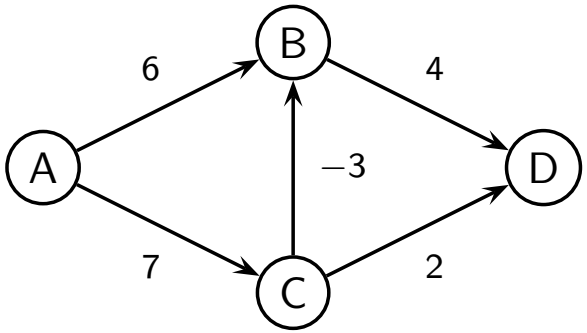
Dopo aver definito un ordinamento topologico dei nodi, l'algoritmo di programmazione dinamica per trovare un albero dei cammini minimi di radice 1 è il seguente:

- Poni $\pi_1 = 0$
- Per ogni nodo $j = 2, \ldots, n$:
  trova $u = \arg\min_{i<j} \{\pi_i + c_{ij}\}$
  poni $p_j = u, \;\; \pi_j = \pi_u + c_{uj}$

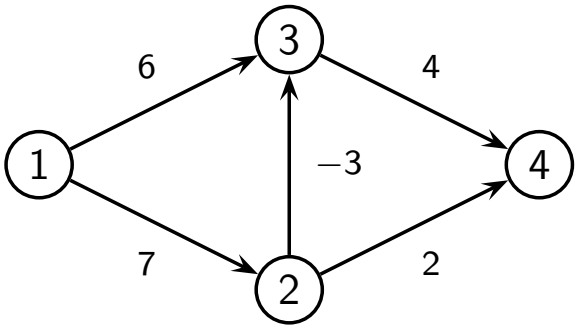
dove p è il vettore dei predecessori dei nodi che rappresenta l'albero e π è il corrispondente vettore dei potenziali dei nodi.

Algoritmo di programmazione dinamica

Esempio 8. Applicare l'algoritmo di programmazione dinamica per trovare l'albero dei cammini minimi di radice A sul seguente grafo aciclico:



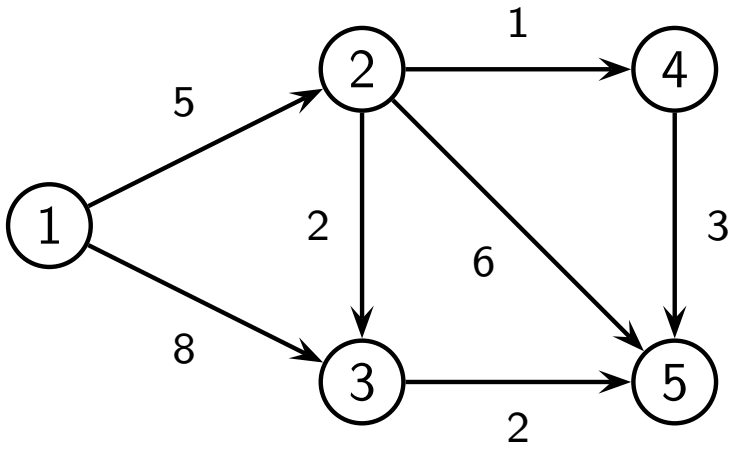
L'ordinamento topologico dei nodi è il seguente:



Algoritmo di programmazione dinamica:
Nodo 1: $\pi_1 = 0$
Nodo 2: $u = 1$, $p_2 = 1$, $\pi_2 = 7$
Nodo 3: $u = \arg\min\{\pi_1 + c_{13}, \pi_2 + c_{23}\} = 2$, $p_3 = 2$, $\pi_3 = 4$
Nodo 4: $u = \arg\min\{\pi_2 + c_{24}, \pi_3 + c_{34}\} = 3$, $p_4 = 3$, $\pi_4 = 8$

Esercizio di riepilogo

Applicare l'algoritmo di Dijkstra, l'algoritmo di Bellman-Ford e l'algoritmo di programmazione dinamica per trovare un albero dei cammini minimi di radice 1 sul seguente grafo e confrontare le soluzioni ottenute.



4 - Problema del flusso massimo

Mauro Passacantando

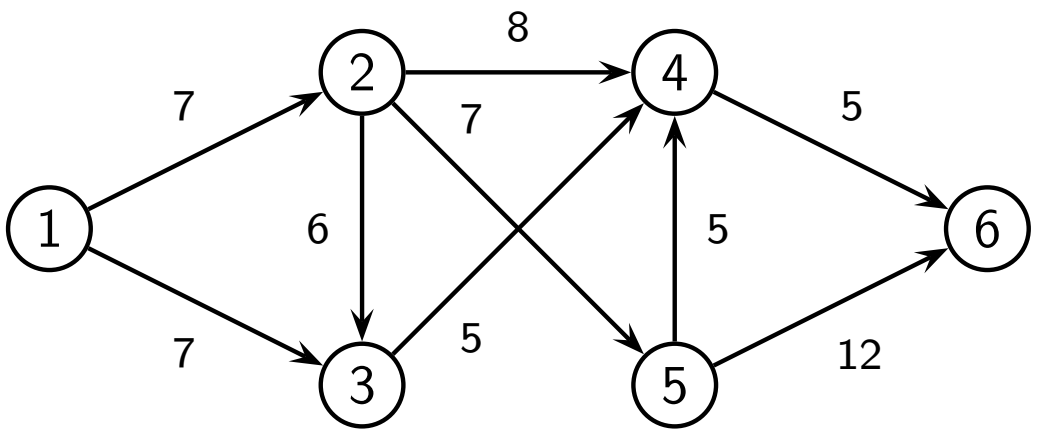
Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2021/22

Problema

Sia (N, A) un grafo orientato in cui è definita una capacità superiore u_{ij} per ogni arco $(i, j) \in A$. Dati un nodo origine $s \in N$ ed un nodo destinazione $t \in N$, vogliamo spedire la massima quantità possibile di flusso da s a t in modo da rispettare le capacità superiori degli archi.

Esempio 1. Trovare il flusso di valore massimo dal nodo 1 al nodo 6 sul seguente grafo (sugli archi sono riportate le capacità superiori).



Modello di ottimizzazione

Variabili:
x_{ij} = flusso sull'arco (i,j), per ogni $(i,j) \in A$
v = flusso totale uscente dal nodo s (o flusso totale entrante nel nodo t)

Modello di PL:

$$\max \; v$$

$$\left.\begin{array}{ll} \displaystyle\sum_{(i,s)\in A} x_{is} - \sum_{(s,j)\in A} x_{sj} = -v & \\ \displaystyle\sum_{(i,t)\in A} x_{it} - \sum_{(t,j)\in A} x_{tj} = v & \\ \displaystyle\sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = 0 & \forall\, k \in N \setminus \{s,t\} \end{array}\right\} \text{vincoli di bilancio}$$

$$0 \le x_{ij} \le u_{ij} \qquad \forall\, (i,j) \in A \qquad \text{vincoli di capacità}$$

Flussi e tagli

Un taglio è una partizione di N in due sottoinsiemi (N_s, N_t)

Un taglio ammissibile è un taglio (N_s, N_t) tale che $s \in N_s$ e $t \in N_t$

Dato un taglio ammissibile (N_s, N_t) ed un flusso x, si definiscono:

$A^+ = \{(i,j) \in A : \ i \in N_s, \ j \in N_t\}$ insieme degli archi diretti del taglio

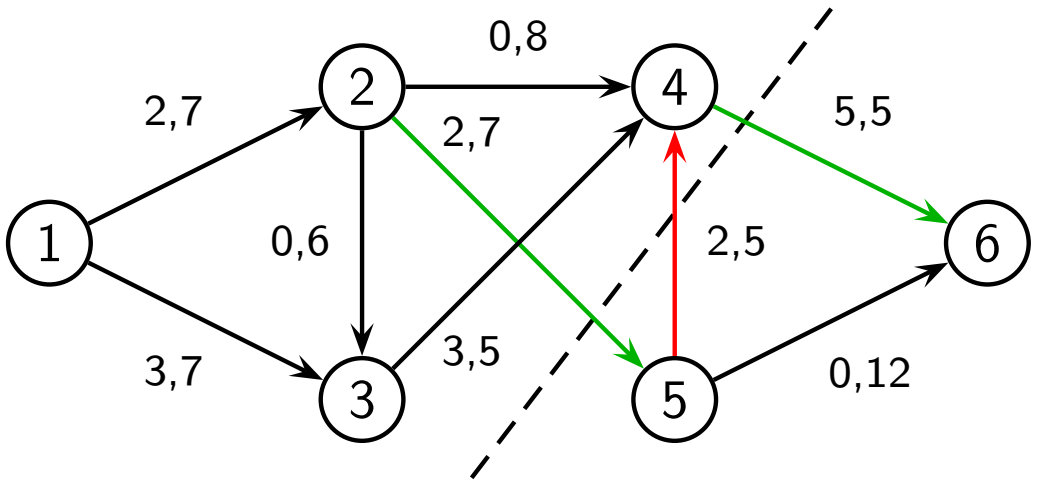
$A^- = \{(i,j) \in A : \ i \in N_t, \ j \in N_s\}$ insieme degli archi inversi del taglio

$$u(N_s, N_t) = \sum_{(i,j) \in A^+} u_{ij}$$ capacità del taglio

$$x(N_s, N_t) = \sum_{(i,j) \in A^+} x_{ij} - \sum_{(i,j) \in A^-} x_{ij}$$ flusso sul taglio

Flussi e tagli

Esempio 2. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



Il taglio (N_s, N_t), dove $N_s = \{1, 2, 3, 4\}$ e $N_t = \{5, 6\}$, è ammissibile
archi diretti: $A^+ = \{(2, 5), (4, 6)\}$
archi inversi: $A^- = \{(5, 4)\}$
capacità del taglio: $u(N_s, N_t) = u_{25} + u_{46} = 7 + 5 = 12$
flusso sul taglio: $x(N_s, N_t) = x_{25} + x_{46} - x_{54} = 2 + 5 - 2 = 5$.

Flussi e tagli

Lemma

Se x è un flusso ammissibile e (N_s, N_t) è un taglio ammissibile, allora

a) $v = x(N_s, N_t)$ (valore del flusso = flusso sul taglio)

b) $x(N_s, N_t) \leq u(N_s, N_t)$ (flusso sul taglio $\leq$ capacità del taglio)

Dimostrazione

a) Consideriamo i vincoli di bilancio corrispondenti ai nodi di N_t:

$$\sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = 0 \quad \forall\ k \in N_t \setminus \{t\}$$

$$\sum_{(i,t)\in A} x_{it} - \sum_{(t,j)\in A} x_{tj} = v$$

Sommando tra loro le equazioni, si ottiene $v = x(N_s, N_t)$.

b) Utilizzando i vincoli di capacità $0 \leq x \leq u$, si ottiene

$$x(N_s, N_t) = \sum_{(i,j)\in A^+} x_{ij} - \sum_{(i,j)\in A^-} x_{ij} \leq \sum_{(i,j)\in A^+} u_{ij} = u(N_s, N_t).$$

Flussi e tagli

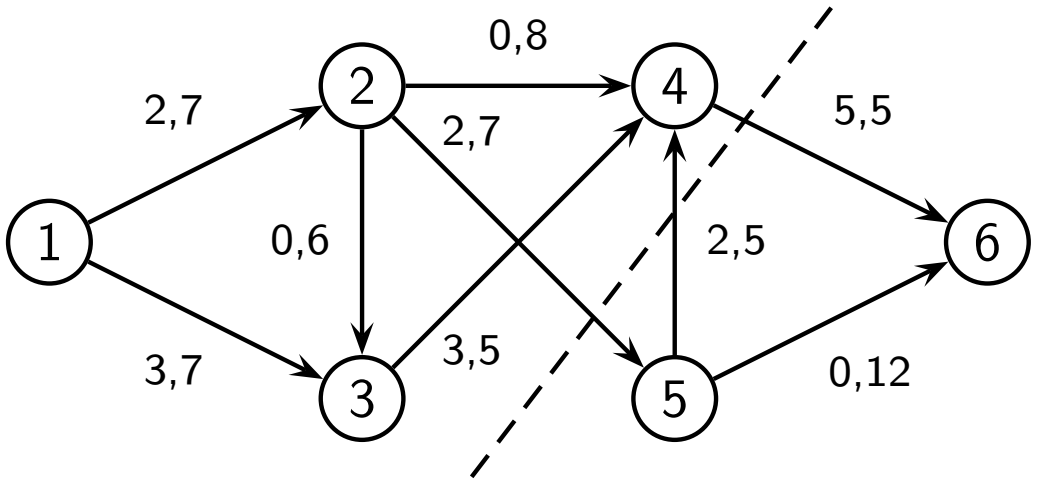
Lemma

Se x è un flusso ammissibile e (N_s, N_t) è un taglio ammissibile, allora

a) $v = x(N_s, N_t)$ (valore del flusso = flusso sul taglio)

b) $x(N_s, N_t) \leq u(N_s, N_t)$ (flusso sul taglio $\leq$ capacità del taglio)

Esempio 2 (segue). Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



valore del flusso: $v = x_{12} + x_{13} = 2 + 3 = 5$

flusso sul taglio: $x(N_s, N_t) = x_{25} + x_{46} - x_{54} = 2 + 5 - 2 = 5.$

capacità del taglio: $u(N_s, N_t) = u_{25} + u_{46} = 7 + 5 = 12$

Condizione sufficiente di ottimalità

Teorema (max flow – min cut)

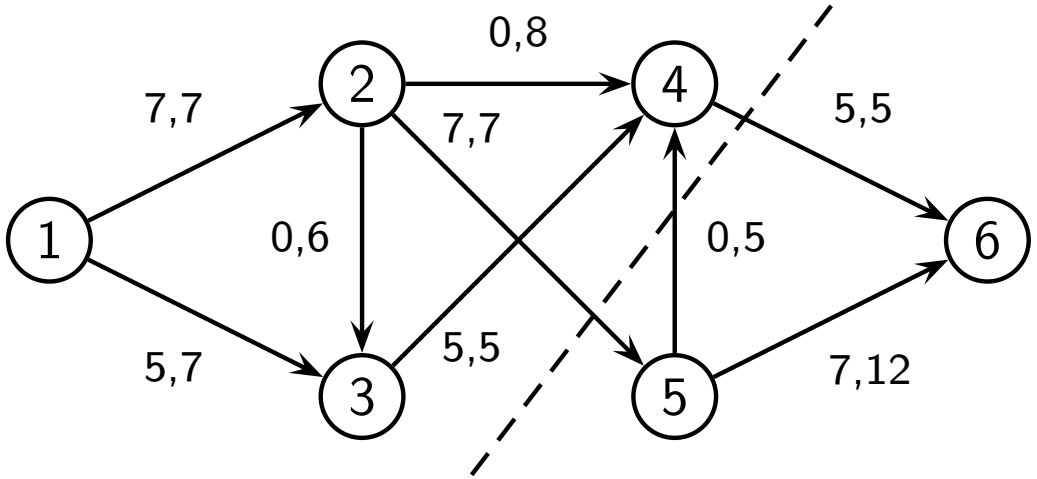
Se esistono un flusso ammissibile x ed un taglio ammissibile (N_s, N_t) tali che $x(N_s, N_t) = u(N_s, N_t)$, allora x è un flusso di valore massimo e (N_s, N_t) è un taglio di capacità minima.

Condizione sufficiente di ottimalità

Teorema (max flow – min cut)

Se esistono un flusso ammissibile x ed un taglio ammissibile (N_s, N_t) tali che $x(N_s, N_t) = u(N_s, N_t)$, allora x è un flusso di valore massimo e (N_s, N_t) è un taglio di capacità minima.

Esempio 3. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



Valore del flusso: $v = x_{12} + x_{13} = 7 + 5 = 12$.
Flusso sul taglio: $x(N_s, N_t) = x_{25} + x_{46} - x_{54} = 7 + 5 - 0 = 12$.
Capacità del taglio: $u(N_s, N_t) = u_{25} + u_{46} = 7 + 5 = 12$.
Quindi il flusso è di valore massimo ed il taglio è di capacità minima.

Grafo residuo

Il teorema max flow – min cut fornisce una condizione sufficiente di ottimalità. Ma come si trova (se esiste) una taglio ammissibile (N_s, N_t) tale che $x(N_s, N_t) = u(N_s, N_t)$?

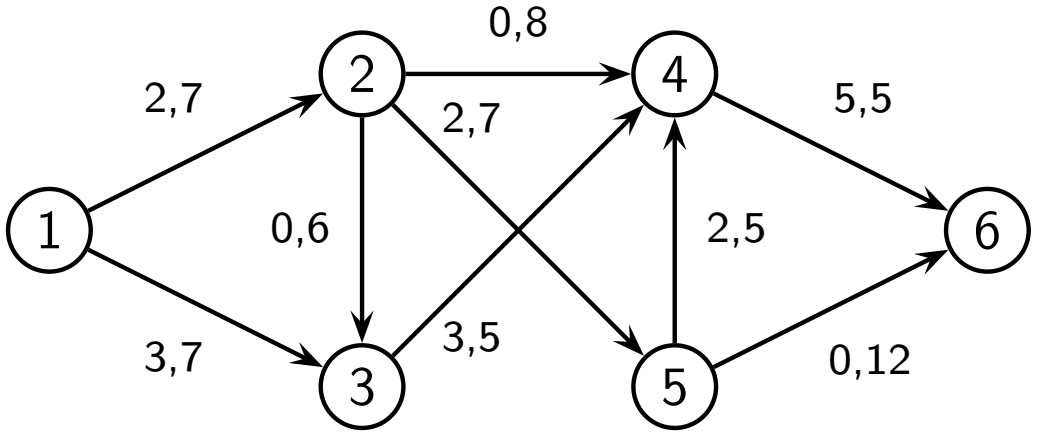
Dato un flusso ammissibile x, il grafo residuo relativo ad x è il grafo $G(x) = (N, A(x))$ avente gli stessi nodi del grafo G, mentre gli archi e le loro capacità residue r_{ij} sono definiti come segue:

se $(i, j) \in A$ con $x_{ij} < u_{ij}$ (arco non saturo) allora $(i, j) \in A(x)$ con $r_{ij} = u_{ij} - x_{ij}$

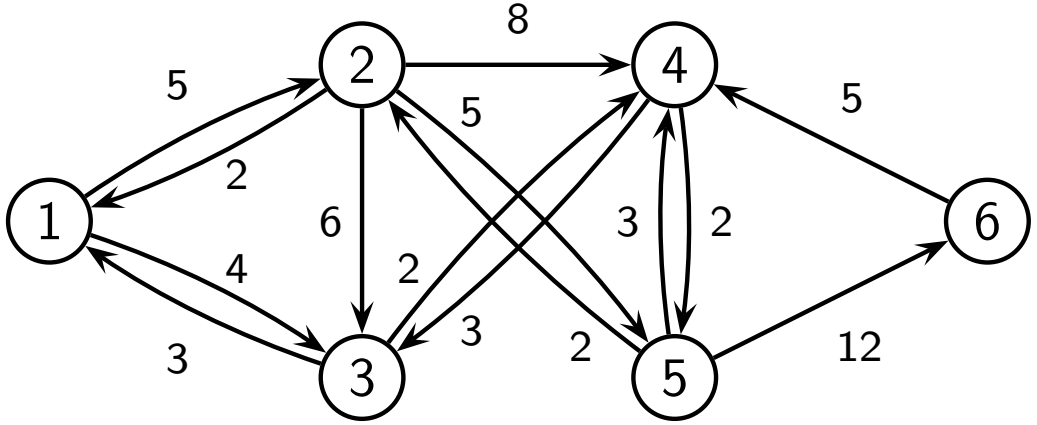
se $(i, j) \in A$ con $x_{ij} > 0$ (arco non vuoto) allora $(j, i) \in A(x)$ con $r_{ji} = x_{ij}$

Grafo residuo

Esempio 4. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



Il grafo residuo associato al flusso indicato sugli archi è il seguente (sugli archi sono indicate le capacità residue):



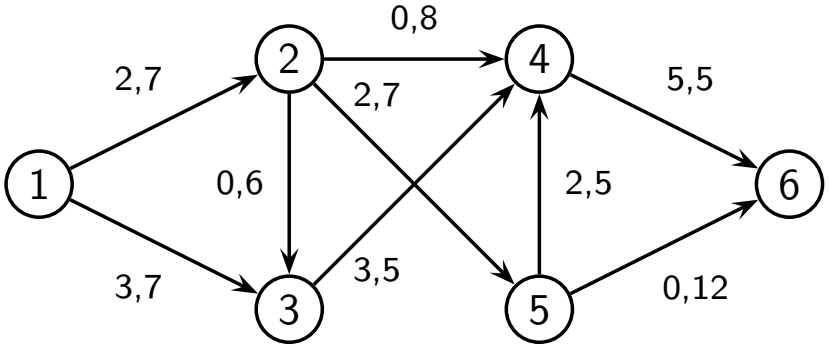
Cammino aumentante

Dato un flusso ammissibile x, un cammino aumentante rispetto ad x è un cammino orientato da s a t nel grafo residuo $G(x)$.

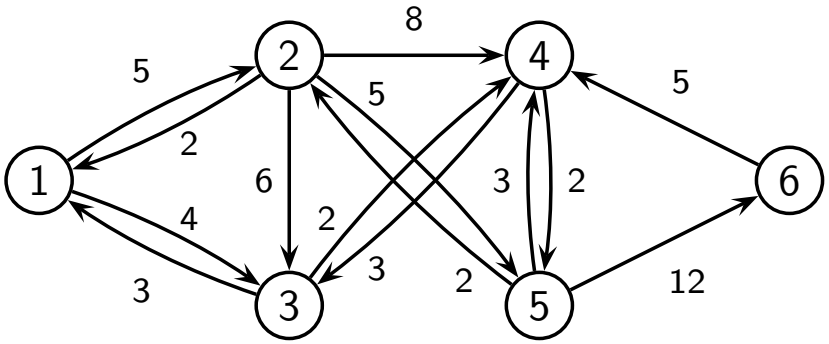
Cammino aumentante

Dato un flusso ammissibile x, un cammino aumentante rispetto ad x è un cammino orientato da s a t nel grafo residuo $G(x)$.

Esempio 4. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



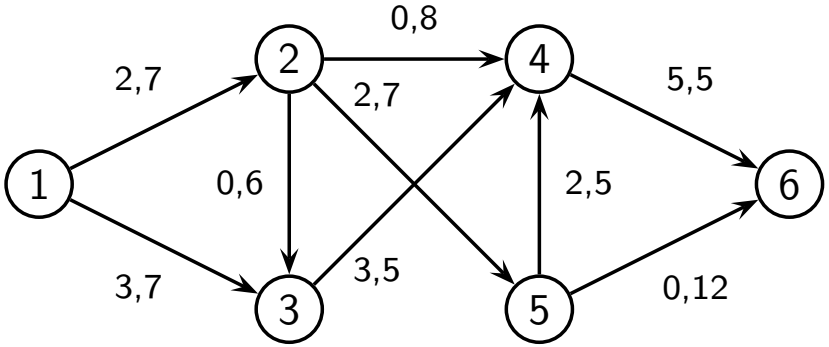
Il grafo residuo è:



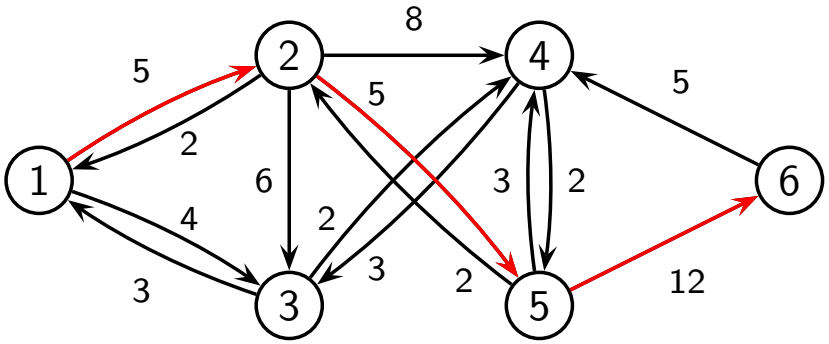
Cammino aumentante

Dato un flusso ammissibile x, un cammino aumentante rispetto ad x è un cammino orientato da s a t nel grafo residuo $G(x)$.

Esempio 4. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



Il grafo residuo è:

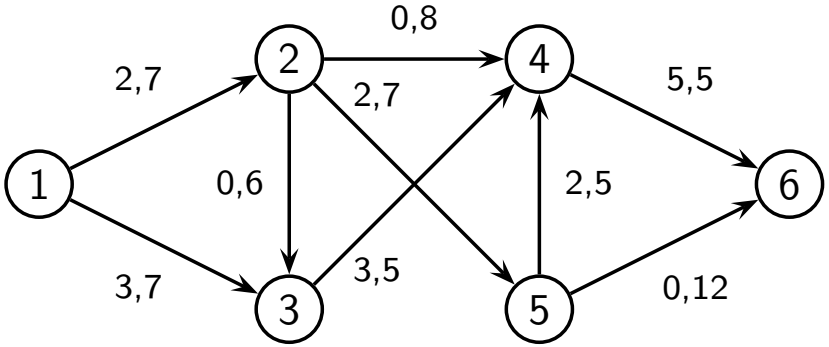


1-2-5-6 è un cammino aumentante (ed è un cammino orientato nel grafo iniziale)

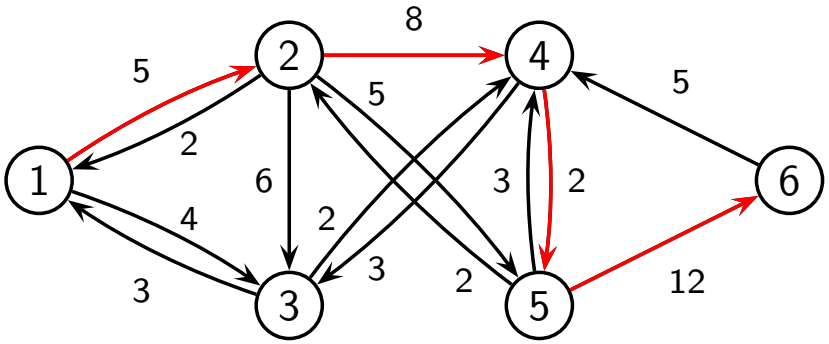
Cammino aumentante

Dato un flusso ammissibile x, un cammino aumentante rispetto ad x è un cammino orientato da s a t nel grafo residuo $G(x)$.

Esempio 4. Si consideri il problema del flusso massimo da 1 a 6 sul seguente grafo in cui su ogni arco sono indicati, in ordine, il flusso e la capacità superiore:



Il grafo residuo è:



1-2-5-6 è un cammino aumentante (ed è un cammino orientato nel grafo iniziale)
1-2-4-5-6 è un cammino aumentante (ma non è un cammino orientato nel grafo iniziale)

Condizione necessaria e sufficiente di ottimalità

Teorema

Sia x un flusso ammissibile. Allora x è un flusso di valore massimo se e solo se non esiste un cammino aumentante rispetto ad x.

Dimostrazione

Se esiste un cammino aumentante rispetto ad x, allora spedendo lungo tale cammino un flusso che rispetta le capacità residue, si ottiene un altro flusso ammissibile x' di valore maggiore rispetto ad x. Pertanto x non è di valore massimo.

Se non esiste un cammino aumentante rispetto ad x, allora si può definire il taglio (N_s, N_t), dove

$$\begin{aligned} N_s &= \{i \in N : \quad \text{esiste in } G(x) \text{ un cammino orientato da } s \text{ a } i\}, \\ N_t &= \{i \in N : \quad \text{non esiste in } G(x) \text{ un cammino orientato da } s \text{ a } i\}. \end{aligned}$$

Il taglio è ammissibile perché $s \in N_s$ e $t \in N_t$. Inoltre, dalla definizione del taglio si ottiene che $x(N_s, N_t) = u(N_s, N_t)$. Quindi, per il teorema max flow – min cut, il flusso x è di valore massimo e (N_s, N_t) è un taglio di capacità minima. $\square$

Algoritmo di Ford-Fulkerson

Ad ogni iterazione cerca un cammino aumentante rispetto al flusso corrente.

Se esiste un cammino aumentante, l'algoritmo aggiorna il flusso spedendo su tale cammino il massimo flusso possibile (pari alla minima capacità residua degli archi di tale cammino).

Se non esiste un cammino aumentate, l'algoritmo si ferma perché il flusso corrente è di valore massimo. Inoltre l'algoritmo trova anche un taglio di capacità minima.

Algoritmo di Ford-Fulkerson

Algoritmo

0. Poni $x = 0$ (flusso nullo su tutti gli archi) e $G(x) = G$.

1. Se esiste un cammino aumentante C rispetto ad x allora
aggiorna il flusso:
calcola $\delta = \min\{r_{ij} : (i,j) \in C\}$ (max quantità da spedire lungo C)
per ogni $(i,j) \in A$:
se $(i,j) \in C$ allora $x_{ij} = x_{ij} + \delta$
se $(j,i) \in C$ allora $x_{ij} = x_{ij} - \delta$
altrimenti stop.

2. Aggiorna il grafo residuo $G(x)$ e torna al passo 1.

Algoritmo di Ford-Fulkerson

Teorema

Se le capacità degli archi sono intere, allora l'algoritmo di Ford-Fulkerson trova un flusso di valore massimo dopo un numero finito di iterazioni.
All'ultima iterazione un taglio di capacità minima (N_s, N_t) si può calcolare nel modo seguente:

$$N_s = \{i \in N : \quad \text{esiste in } G(x) \text{ un cammino orientato da } s \text{ a } i \},$$

$$N_t = \{i \in N : \quad \text{non esiste in } G(x) \text{ un cammino orientato da } s \text{ a } i \},$$

Un altro taglio di capacità minima (N'_s, N'_t) si può ottenere nel modo seguente:

$$N'_s = \{i \in N : \quad \text{non esiste in } G(x) \text{ un cammino orientato da } i \text{ a } t \},$$

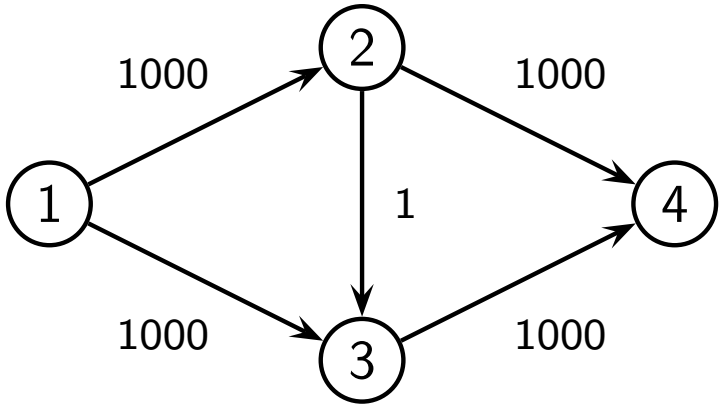
$$N'_t = \{i \in N : \quad \text{esiste in } G(x) \text{ un cammino orientato da } i \text{ a } t \},$$

Il taglio di capacità minima è unico se e solo se i due tagli precedenti coincidono.

Algoritmo di Ford-Fulkerson

Se non viene data una regola precisa per la ricerca del cammino aumentante, la convergenza dell'algoritmo può essere molto lenta.

Esempio 5. Consideriamo il problema di flusso massimo da 1 a 4 sul seguente grafo:



Se scegliamo come cammini aumentanti 1-2-3-4, 1-3-2-4, 1-2-3-4, ecc. ad ogni iterazione spediamo solo 1 unità di flusso lungo il cammino aumentante e l'algoritmo termina dopo 2000 iterazioni.

Variante di Edmonds-Karp

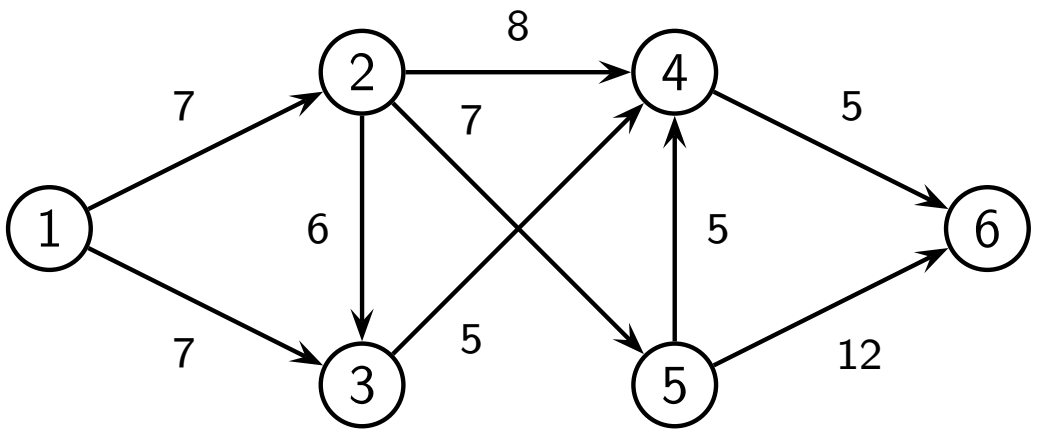
Se ad ogni iterazione si cerca un cammino aumentante con il minimo numero di archi, allora l'algoritmo di Ford-Fulkerson ha una complessità polinomiale pari a $O(nm^2)$, dove $n = |N|$ e $m = |A|$.

Per trovare un cammino aumentante con il minimo numero di archi basta fare una visita a ventaglio del grafo residuo. Sia p un vettore di predecessori dei nodi e Q un insieme di nodi (Q rappresenta una fila i cui elementi si estraggono dalla testa e si aggiungono in coda).

1. Inizializza $p_i := \begin{cases} -1 & \text{se } i \neq s \\ 0 & \text{se } i = s \end{cases}$, $Q := \boxed{s}$
2. **if** $Q = \emptyset$
   then STOP (p fornisce un taglio di capacità minima
   $N_s = \{i \in N : \ p_i \neq -1\}, N_t = \{i \in N : \ p_i = -1\}$)
   else estrai il primo elemento i di Q.
3. **if** $(i, t) \in A(x)$ **then** $p_t := i$, STOP (p fornisce un cammino aumentante).
4. In $G(x)$ analizza gli archi uscenti da i in ordine lessicografico
   Per ogni $(i, j) \in A(x)$:
   if $p_j := -1$ **then** $p_j := i$, aggiungi j in fondo a Q.
   Torna al passo 2.

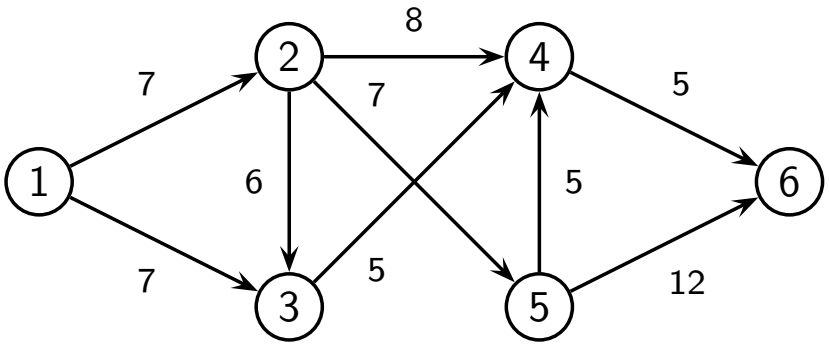
Algoritmo di Ford-Fulkerson

Esempio 6. Applichiamo l'algoritmo di Ford-Fulkerson, con la variante di Edmonds-Karp per la ricerca del cammino aumentante, per trovare un flusso di valore massimo dal nodo 1 al nodo 6 sul seguente grafo (sugli archi sono riportate le capacità superiori).



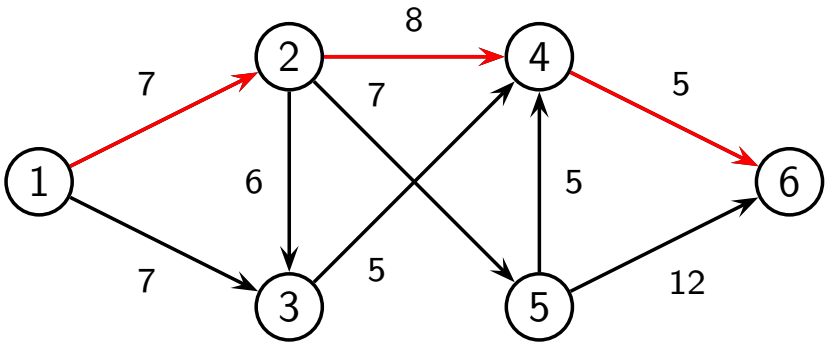
Algoritmo di Ford-Fulkerson

Iterazione 1. Partiamo con $x = 0$, quindi il grafo residuo $G(x) = G$:



Algoritmo di Ford-Fulkerson

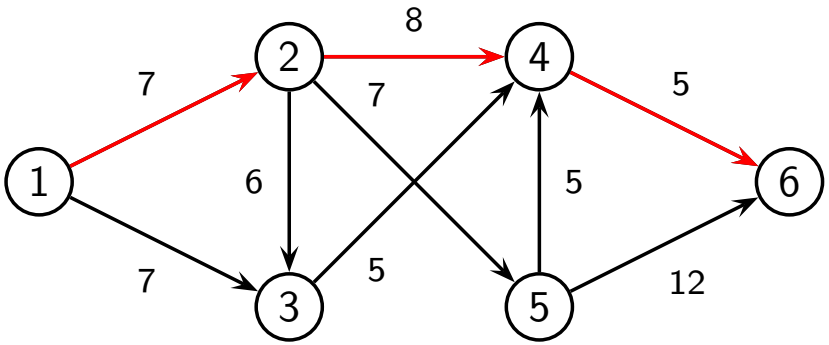
Iterazione 1. Partiamo con $x = 0$, quindi il grafo residuo $G(x) = G$:



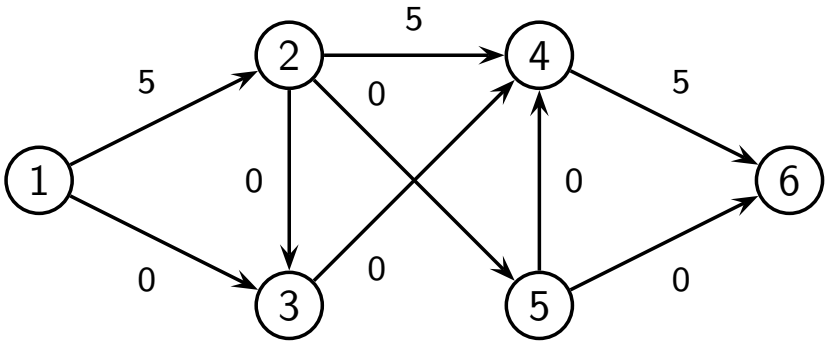
L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 1, 1, 2, 2, 4)$ che fornisce il cammino aumentante 1-2-4-6. Si ha $\delta = \min\{7, 8, 5\} = 5$.

Algoritmo di Ford-Fulkerson

Iterazione 1. Partiamo con $x = 0$, quindi il grafo residuo $G(x) = G$:

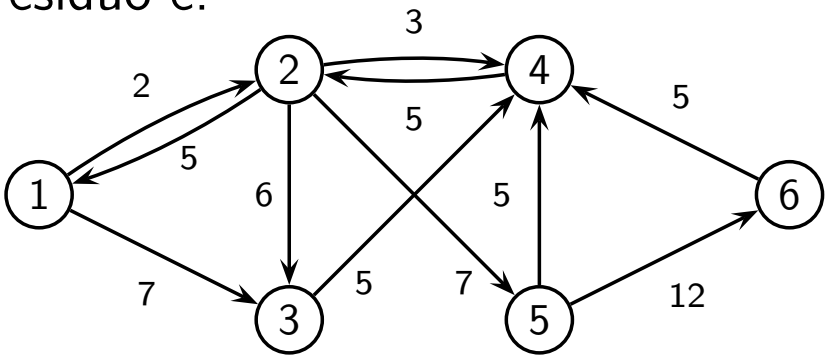


L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 1, 1, 2, 2, 4)$ che fornisce il cammino aumentante 1-2-4-6. Si ha $\delta = \min\{7, 8, 5\} = 5$.
Il nuovo flusso è



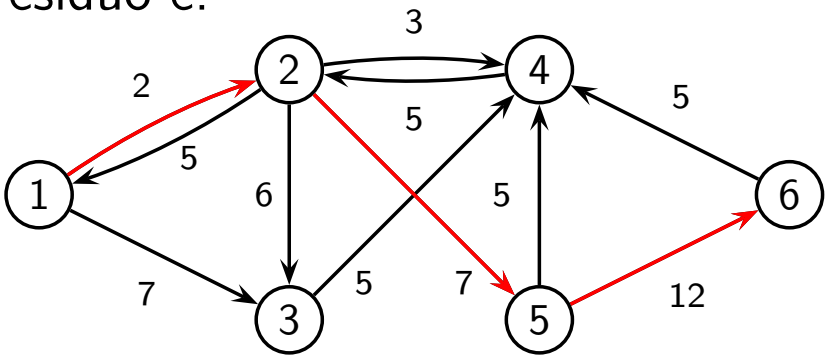
Algoritmo di Ford-Fulkerson

Iterazione 2. Il grafo residuo è:



Algoritmo di Ford-Fulkerson

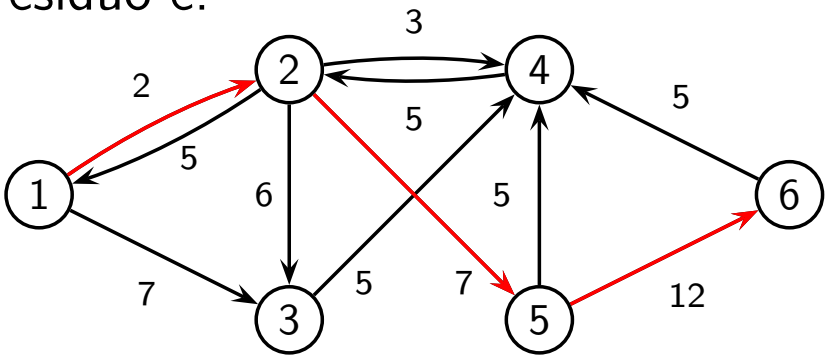
Iterazione 2. Il grafo residuo è:



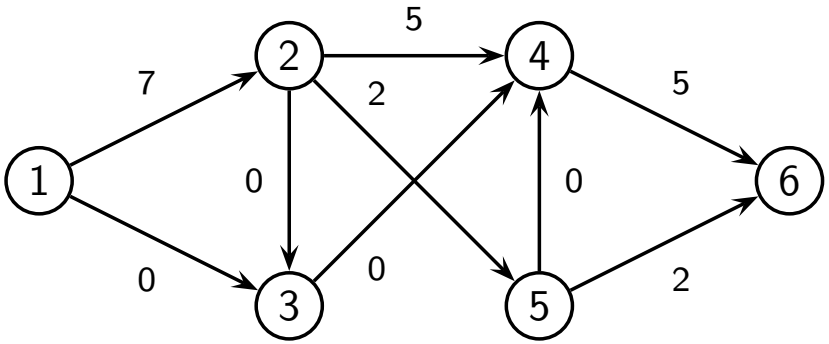
L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 1, 1, 2, 2, 5)$ che fornisce il cammino aumentante 1-2-5-6. Si ha $\delta = \min\{2, 7, 12\} = 2$.

Algoritmo di Ford-Fulkerson

Iterazione 2. Il grafo residuo è:

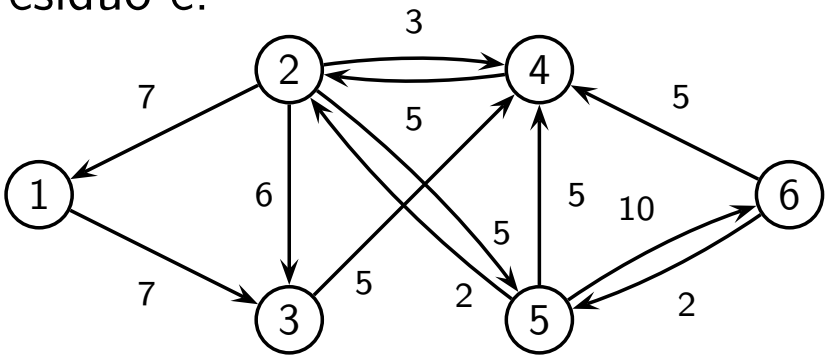


L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 1, 1, 2, 2, 5)$ che fornisce il cammino aumentante 1-2-5-6. Si ha $\delta = \min\{2, 7, 12\} = 2$.
Il nuovo flusso è



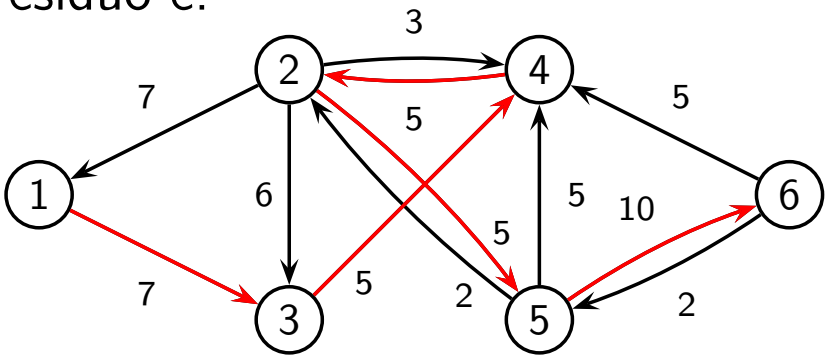
Algoritmo di Ford-Fulkerson

Iterazione 3. Il grafo residuo è:



Algoritmo di Ford-Fulkerson

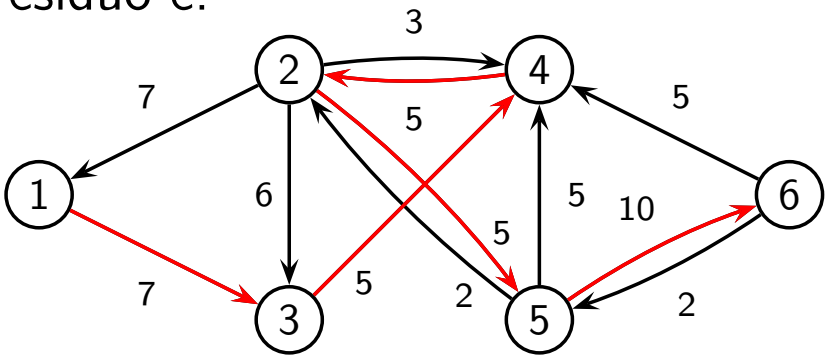
Iterazione 3. Il grafo residuo è:



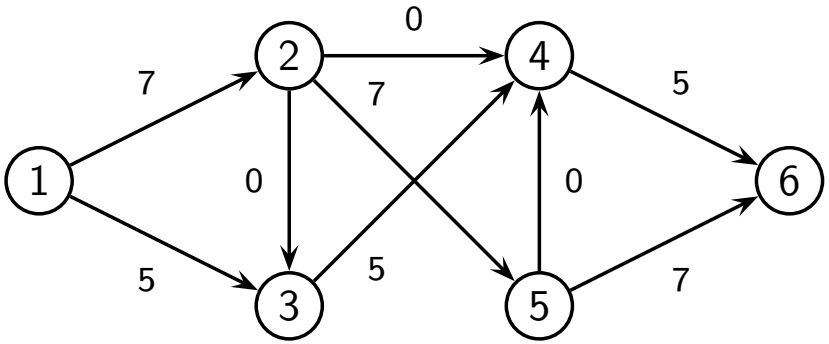
L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 4, 1, 3, 2, 5)$ che fornisce il cammino aumentante 1-3-4-2-5-6. Si ha $\delta = \min\{7, 5, 5, 5, 10\} = 5$.

Algoritmo di Ford-Fulkerson

Iterazione 3. Il grafo residuo è:

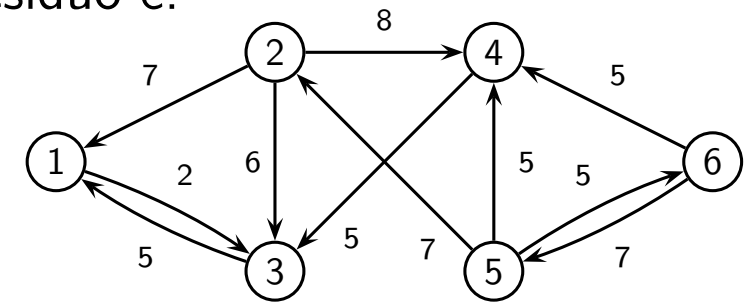


L'algoritmo di Edmonds-Karp trova il vettore $p = (0, 4, 1, 3, 2, 5)$ che fornisce il cammino aumentante 1-3-4-2-5-6. Si ha $\delta = \min\{7, 5, 5, 5, 10\} = 5$.
Il nuovo flusso è



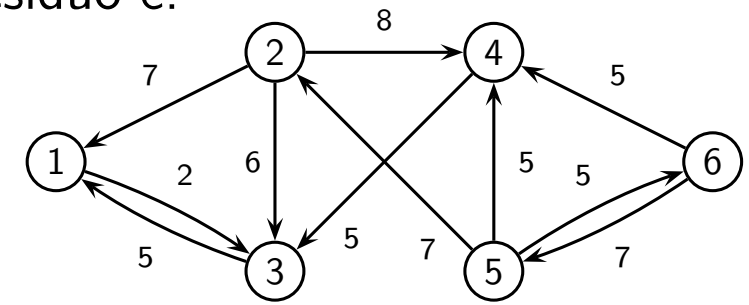
Algoritmo di Ford-Fulkerson

Iterazione 4. Il grafo residuo è:



Algoritmo di Ford-Fulkerson

Iterazione 4. Il grafo residuo è:



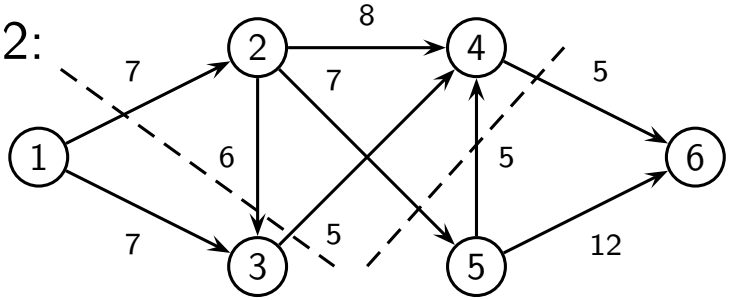
L'algoritmo di Edmonds-Karp trova il vettore $p = (0, -1, 1, -1, -1, -1)$, pertanto il flusso trovato all'iterazione precedente è ottimo ($v = 12$) ed un taglio di capacità minima è

$$N_s = \{1, 3\} \text{ (nodi raggiungibili da 1)}, \; N_t = \{2, 4, 5, 6\} \text{ (nodi non raggiungibili da 1)}.$$

Notiamo che un altro taglio di capacità minima è

$$N'_s = \{1, 2, 3, 4\} \text{ (nodi dai quali non è possibile raggiungere 6)},$$
$$N'_t = \{5, 6\} \text{ (nodi dai quali è possibile raggiungere 6)}.$$

Osserviamo che i due tagli hanno capacità 12:



Esercizio 4.1

La Scuola di Ingegneria si divide in 4 dipartimenti (Informatica, Elettronica, Meccanica, Civile). Ogni dipartimento ha ricevuto il seguente numero di domande di studenti per partecipare al programma Erasmus: 20 per Informatica, 16 per Elettronica, 15 per Meccanica e 10 per Civile. Sono disponibili varie convenzioni con diverse università straniere. In Spagna non si possono mandare più di 30 studenti, in Germania non più di 20, in Norvegia non più di 5, in Francia non più di 25, in Gran Bretagna non più di 15. Le convenzioni attivate non permettono di mandare più di 10 studenti di un dipartimento nello stesso paese. Inoltre, gli studenti di Informatica non possono andare in Francia, quelli di Elettronica in Spagna, quelli di Meccanica in Norvegia e quelli di Civile in Germania.

a) Formulare il problema di massimizzare il numero di studenti da mandare nei programmi Erasmus come un problema di flusso massimo su un opportuno grafo.

b) Trovare la soluzione ottima del problema.

c) Trovare un taglio di capacità minima.

Problema dell'accoppiamento di massima cardinalità

Un grafo orientato (N, A) è detto bipartito se esistono due insiemi disgiunti $O, D \subset N$ tali che $N = O \cup D$ e $A \subseteq O \times D$.

In un grafo bipartito (N, A) un accoppiamento (*matching*) è un insieme di archi $M \subseteq A$ tale che su ogni nodo di N incide al più un arco di M.

Il problema dell'accoppiamento di massima cardinalità in un grafo bipartito (N, A) consiste nel trovare un accoppiamento con il massimo numero di archi.

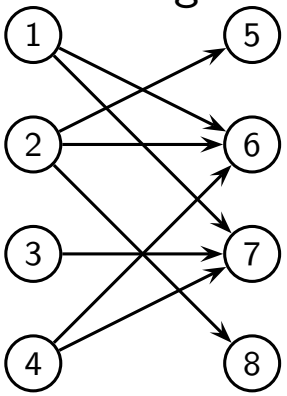
Problema dell'accoppiamento di massima cardinalità

Un grafo orientato (N, A) è detto bipartito se esistono due insiemi disgiunti $O, D \subset N$ tali che $N = O \cup D$ e $A \subseteq O \times D$.

In un grafo bipartito (N, A) un accoppiamento (*matching*) è un insieme di archi $M \subseteq A$ tale che su ogni nodo di N incide al più un arco di M.

Il problema dell'accoppiamento di massima cardinalità in un grafo bipartito (N, A) consiste nel trovare un accoppiamento con il massimo numero di archi.

Esempio. Supponiamo che O sia un insieme di operai, D un insieme di lavori ed ogni arco $(i, j) \in A$ indichi che l'operaio i è in grado di svolgere il lavoro j. Ogni operaio può eseguire al più un lavoro ed ogni lavoro può essere eseguito da al più un operaio. Come assegniamo gli operai ai lavori in modo da eseguire il maggior numero di lavori possibile?



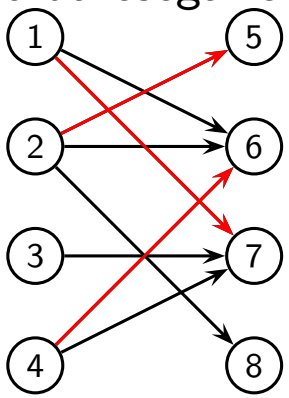
Problema dell'accoppiamento di massima cardinalità

Un grafo orientato (N, A) è detto bipartito se esistono due insiemi disgiunti $O, D \subset N$ tali che $N = O \cup D$ e $A \subseteq O \times D$.

In un grafo bipartito (N, A) un accoppiamento (*matching*) è un insieme di archi $M \subseteq A$ tale che su ogni nodo di N incide al più un arco di M.

Il problema dell'accoppiamento di massima cardinalità in un grafo bipartito (N, A) consiste nel trovare un accoppiamento con il massimo numero di archi.

Esempio. Supponiamo che O sia un insieme di operai, D un insieme di lavori ed ogni arco $(i, j) \in A$ indichi che l'operaio i è in grado di svolgere il lavoro j. Ogni operaio può eseguire al più un lavoro ed ogni lavoro può essere eseguito da al più un operaio. Come assegniamo gli operai ai lavori in modo da eseguire il maggior numero di lavori possibile?



$M = \{(1, 7), (2, 5), (4, 6)\}$ è un accoppiamento. È anche di massima cardinalità?

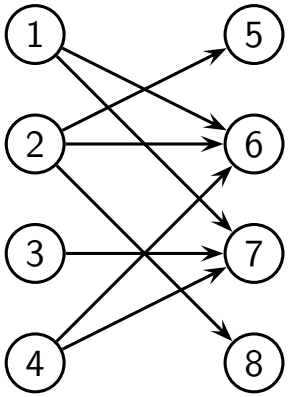
Problema dell'accoppiamento di massima cardinalità

Il problema dell'accoppiamento di massima cardinalità su un grafo bipartito (N, A) è equivalente al problema del flusso massimo da s a t sul grafo (N', A'), dove $N' = N \cup \{s, t\}$, $A' = A \cup \{(s, i) : \ i \in O\} \cup \{(j, t) : \ j \in D\}$ e le capacità superiori di tutti gli archi sono uguali a 1.

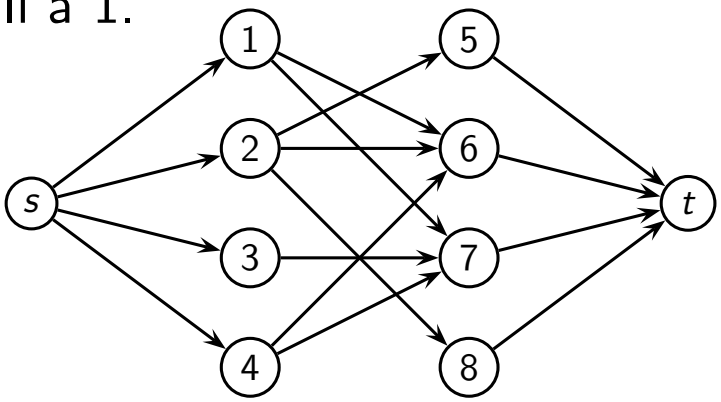
Problema dell'accoppiamento di massima cardinalità

Il problema dell'accoppiamento di massima cardinalità su un grafo bipartito (N, A) è equivalente al problema del flusso massimo da s a t sul grafo (N', A'), dove $N' = N \cup \{s, t\}$, $A' = A \cup \{(s, i) : \ i \in O\} \cup \{(j, t) : \ j \in D\}$ e le capacità superiori di tutti gli archi sono uguali a 1.

Esempio. L'accoppiamento di massima cardinalità sul grafo



è equivalente al problema del flusso massimo sul grafo seguente in cui le capacità superiori degli archi sono uguali a 1.



Qual è la soluzione ottima?

5 - Problema del flusso di costo minimo

Mauro Passacantando

Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2020/21

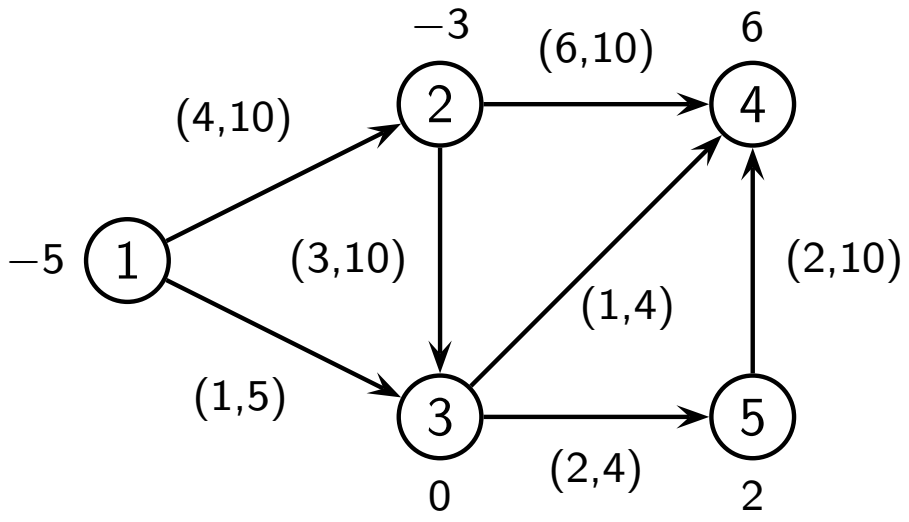
Problema

Dato un grafo orientato (N, A) in cui sono definiti

- un costo c_{ij} per spedire una unità di flusso sull'arco $(i, j) \in A$
- una capacità superiore u_{ij} per ogni arco $(i, j) \in A$
- un bilancio b_i per ogni nodo $i \in N$ (i è detto sorgente se $b_i < 0$, i è detto pozzo se $b_i > 0$)

trovare un flusso sugli archi che rispetti i bilanci dei nodi e le capacità degli archi e che sia di costo totale minimo.

Esempio 1. Trovare un flusso di costo minimo sul seguente grafo (su ogni nodo è indicato il bilancio, su ogni arco sono indicati, in ordine, il costo e la capacità superiore).



Modello

Variabili decisionali: per ogni arco $(i,j) \in A$ definiamo x_{ij} = flusso sull'arco (i,j)

Modello di ottimizzazione:

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = b_k \qquad \forall\ k \in N \qquad \text{(vincoli di bilancio)} \\ 0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A \qquad \text{(vincoli di capacità)} \end{cases}$$

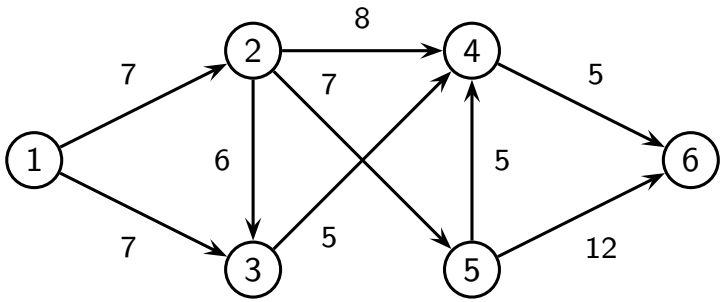
Casi particolari – cammini minimi

Il problema dell'albero dei cammini minimi di radice r è un particolare problema di flusso di costo minimo, dove le capacità superiori $u_{ij} = +\infty$ per ogni $(i, j) \in A$ ed i bilanci sono $b_r = -(n-1)$ e $b_i = 1$ per ogni $i \neq r$.

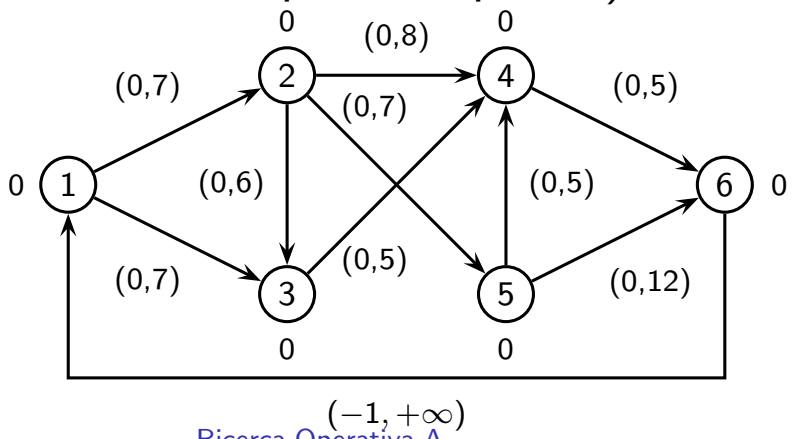
Casi particolari – flusso massimo

Il problema del flusso massimo da s a t su un grafo (N, A) equivale ad un problema di flusso di costo minimo sul grafo $(N, A \cup \{(t, s)\})$, dove $b_i = 0$ per ogni $i \in N$, $c_{ij} = 0$ per ogni $(i, j) \in A$, $c_{ts} = -1$ e $u_{ts} = +\infty$. Tale problema è chiamato problema di circolazione.

Esempio 2. Il problema del flusso massimo dal nodo 1 al nodo 6 sul seguente grafo (sugli archi sono indicate le capacità superiori)



equivale al problema di flusso di costo minimo sul seguente grafo (sui nodi sono indicati i bilanci mentre sugli archi i costi e le capacità superiori)



Esistenza di soluzioni ammissibili

Per determinare se esistono soluzioni ammissibili per il problema del flusso di costo minimo su un grafo (N, A), si può risolvere il problema del flusso massimo sul grafo (N', A'), dove
$N' = N \cup \{s, t\}$,
$A' = A \cup \{(s, i) : \ i \in N \text{ tale che } b_i < 0\} \cup \{(j, t) : \ j \in N \text{ tale che } b_i > 0\}$,
$u'_{si} = -b_i$, $u'_{jt} = b_j$, $u'_{ij} = u_{ij}$ per ogni $(i, j) \in A$.

Sia x il flusso di valore massimo sul grafo (N', A').

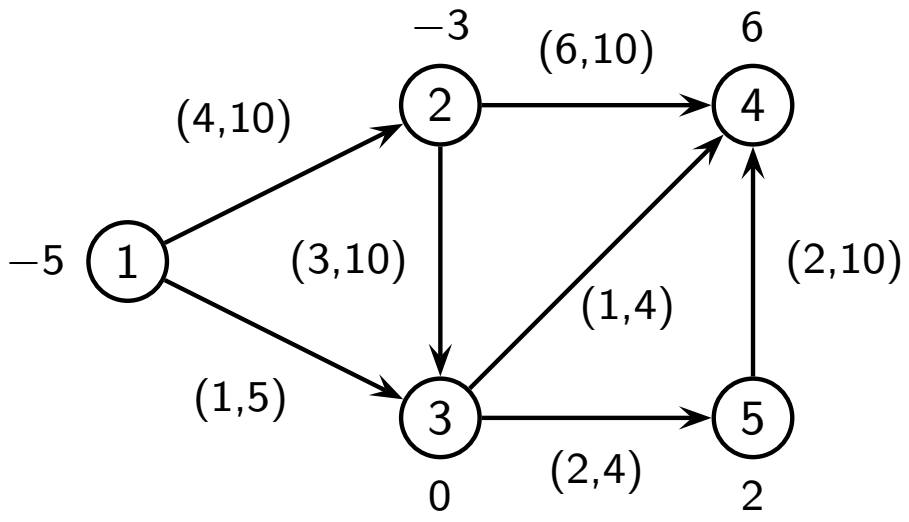
Se il valore di x è uguale a $\sum_{\substack{i \in N \\ b_i > 0}} b_i$ (cioè tutti gli archi di $A' \setminus A$ sono saturi),

allora x è ammissibile per il problema del flusso di costo minimo,

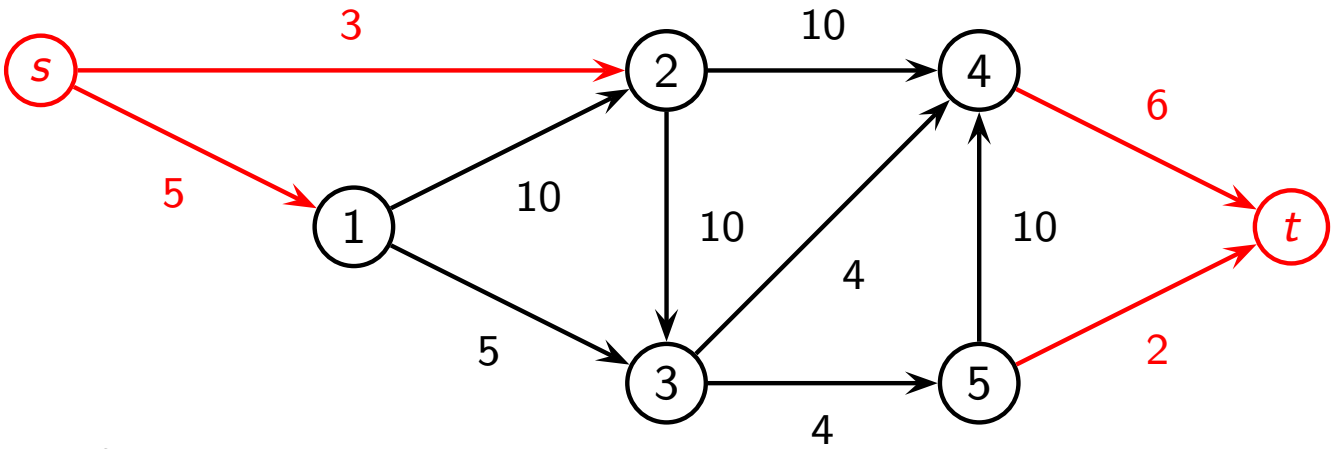
altrimenti il flusso di costo minimo non ammette soluzioni ammissibili.

Esistenza di soluzioni ammissibili

Esempio 3. Per trovare una soluzione ammissibile per il flusso di costo minimo sul grafo



si può risolvere il problema del flusso massimo sul seguente grafo (sugli archi sono indicate le capacità superiori):



Verificare che il flusso massimo da s a t ha valore uguale a 8.

Condizioni di ottimalità

Ad ogni flusso x associamo il grafo residuo $G(x) = (N, A')$ in cui l'insieme degli archi A' è così definito:

- se $(i, j) \in A$ con $x_{ij} < u_{ij}$, allora $(i, j) \in A'$ con capacità residua $r_{ij} = u_{ij} - x_{ij}$ e costo $c'_{ij} = c_{ij}$,
- se $(i, j) \in A$ con $x_{ij} > 0$, allora $(j, i) \in A'$ con capacità residua $r_{ji} = x_{ij}$ e costo $c'_{ji} = -c_{ij}$.

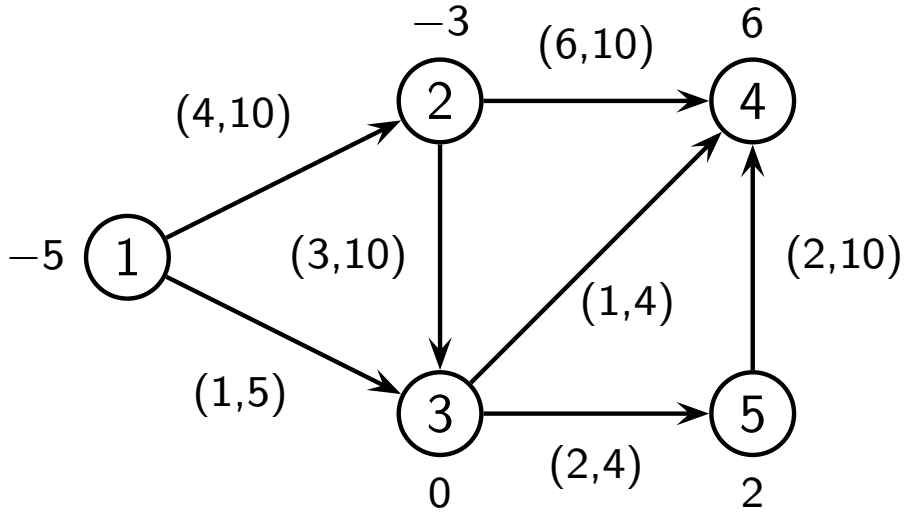
Teorema

Sia x un flusso ammissibile.
x è ottimo se e solo se in $G(x)$ non esistono cicli orientati di costo negativo.

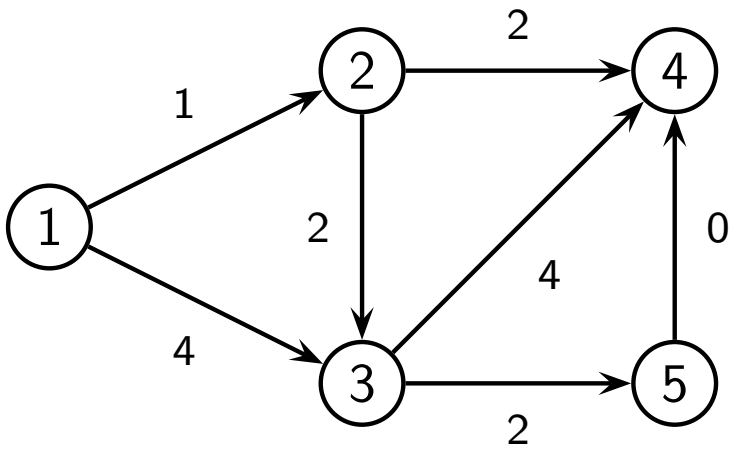
[Un ciclo orientato in $G(x)$ è chiamato ciclo aumentante rispetto a x).

Condizioni di ottimalità

Esempio 4. Si consideri il problema del flusso di costo minimo sul grafo

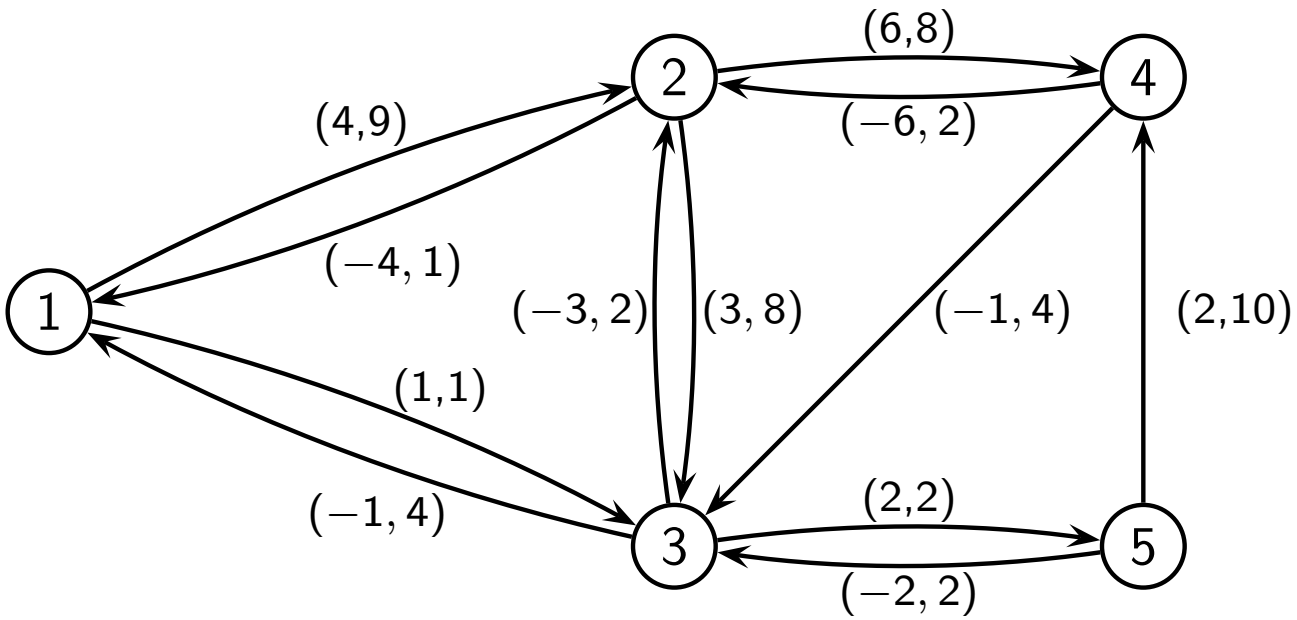


Dire se il seguente flusso ammissibile x è ottimo oppure no:



Condizioni di ottimalità

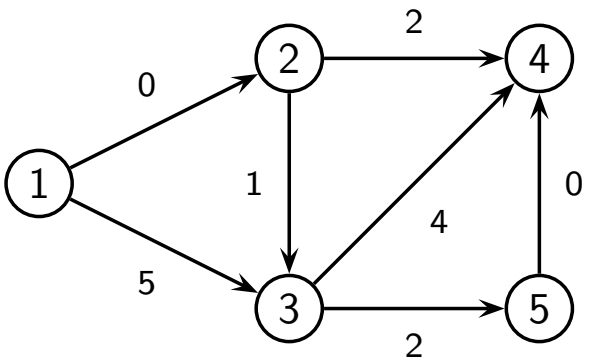
Esempio 4 (segue). Il grafo residuo associato al flusso x è il seguente (su ogni arco sono indicati il costo e la capacità residua):



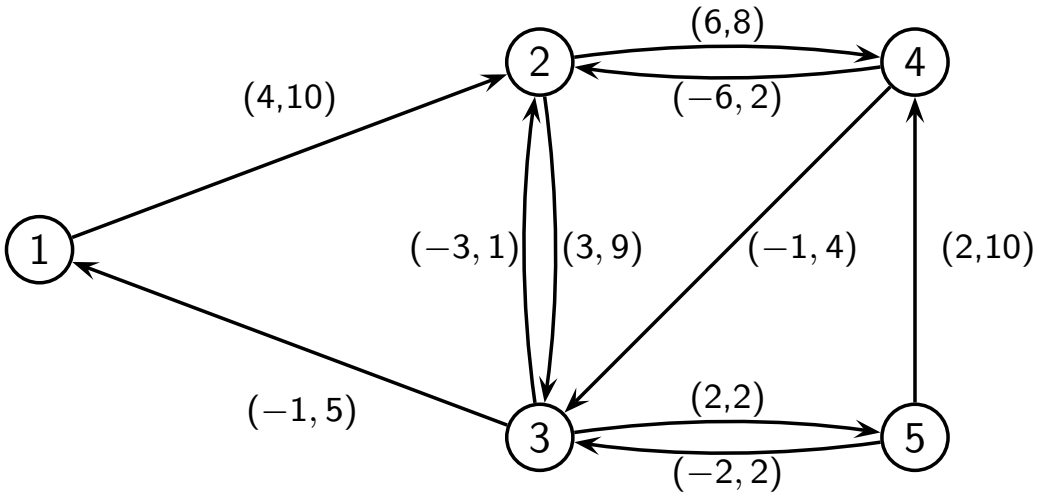
Poiché il ciclo 1-3-2-1 ha costo -6, il flusso x non è ottimo.

Condizioni di ottimalità

Esempio 4 (segue). Se modifichiamo il flusso x spedendo una unità di flusso lungo il ciclo 1-3-2-1, otteniamo il seguente flusso x':



Il grafo residuo $G(x')$ associato a x' è il seguente:



Poiché in $G(x')$ non esistono cicli orientati di costo negativo, il flusso x' è ottimo.

Pseudoflussi

Uno pseudoflusso x è un flusso che rispetta i vincoli di capacità degli archi ($0 \leq x_{ij} \leq u_{ij}$) ma non necessariamente i vincoli di bilancio dei nodi.

Se x è uno pseudoflusso, allora

$$e_x(i) = \sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} - b_i$$

è chiamato lo sbilanciamento del nodo i rispetto a x.
Indichiamo con
$E_x = \{i \in N : \ e_x(i) > 0\}$ l'insieme dei nodi con eccesso di flusso
$D_x = \{i \in N : \ e_x(i) < 0\}$ l'insieme dei nodi con difetto di flusso
$g(x) = \sum_{i\in E_x} e_x(i)$ lo sbilanciamento complessivo di x.
Uno pseudoflusso x è ammissibile (cioè rispetta anche i vincoli di bilancio) se e solo se $E_x = D_x = \emptyset$ se e solo se $g(x) = 0$.

Uno pseudoflusso x è detto minimale se ha costo minimo tra tutti gli pseudoflussi aventi lo stesso vettore di sbilanciamenti e_x.
Teorema. Uno pseudoflusso x è minimale se e solo se non esistono cicli aumentanti rispetto a x di costo negativo.

Algoritmo dei cammini minimi successivi

L'algoritmo dei cammini minimi successivi parte con uno pseudoflusso minimale.

Ad ogni iterazione l'algoritmo mantiene uno pseudoflusso x minimale e cerca nel grafo residuo $G(x)$ un cammino di costo minimo da un nodo di E_x (con eccesso di flusso) ad un nodo di D_x (con difetto di flusso), in modo da diminuire lo sbilanciamento complessivo al minor costo possibile.

Algoritmo dei cammini minimi successivi

Algoritmo

0. Per ogni $(i,j) \in A$ poni $x_{ij} = \begin{cases} 0 & \text{se } c_{ij} \geq 0, \\ u_{ij} & \text{se } c_{ij} < 0. \end{cases}$

1. Se $E_x = \emptyset$ allora stop (x è ottimo)

2. Aggiungi a $G(x)$ un nodo r e gli archi (r,i) per ogni $i \in E_x$ con costo $c_{ri} = 0$ e capacità residua $r_{ri} = +\infty$.
Trova in $G(x)$ un albero dei cammini minimi T di radice r.

3. Trova $t = \arg\min_{i \in D_x} \pi_i$ (nodo di D_x con potenziale minimo).
Se $\pi_t = +\infty$ allora stop (non esistono flussi ammissibili)

4. Indica con P il cammino da r a t contenuto nell'albero T
(P passa per un nodo $s \in E_x$).
Calcola $\delta = \min\{e_x(s), -e_x(t), \min\{r_{ij} : \ (i,j) \in P\}\}$ (max quantità da spedire lungo P).

5. Aggiorna x spedendo δ unità di flusso lungo il cammino P e torna al passo 1.

Algoritmo dei cammini minimi successivi

Teorema

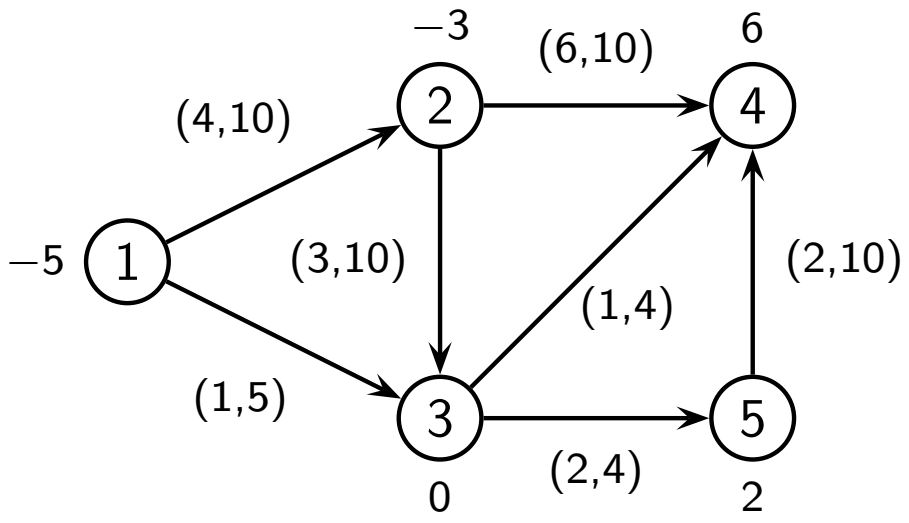
Se le capacità ed i bilanci sono numeri interi, allora l'algoritmo trova una soluzione ottima dopo un numero finito di iterazioni.

Dim. (sketch)

- Ad ogni iterazione x ha componenti intere e quindi anche lo sbilanciamento complessivo $g(x)$ è intero. Poiché $g(x)$ diminuisce ad ogni iterazione di almeno una unità, l'algoritmo si ferma dopo un numero finito di iterazioni.
- Alla prima iterazione x è uno pseudoflusso minimale perché in $G(x)$ non ci sono archi di costo negativo.
- Poiché ad ogni iterazione si spedisce flusso lungo un cammino aumentante di costo minimo da un nodo di E_x ad un nodo di D_x, il nuovo pseudoflusso ottenuto rimane minimale.
- All'ultima iterazione lo pseudoflusso minimale è ammissibile e quindi è un flusso di costo minimo.

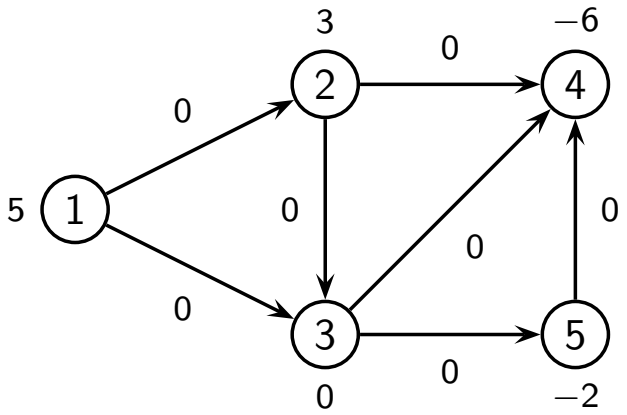
Algoritmo dei cammini minimi successivi

Esempio 5. Applicare l'algoritmo dei cammini minimi successivi per risolvere il problema del flusso di costo minimo sul grafo seguente:



Algoritmo dei cammini minimi successivi

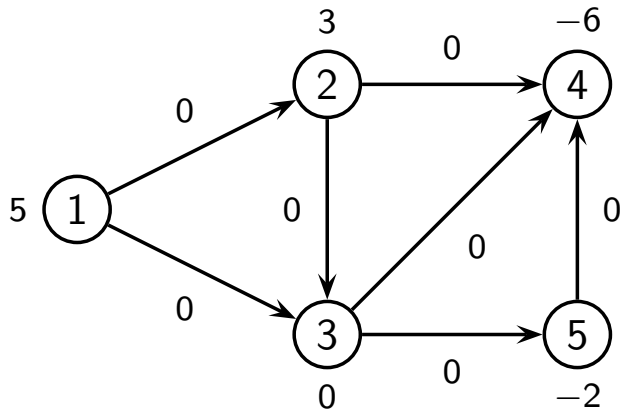
Iterazione 1. Lo pseudoflusso x iniziale è il seguente (su ogni nodo è indicato lo sbilanciamento):



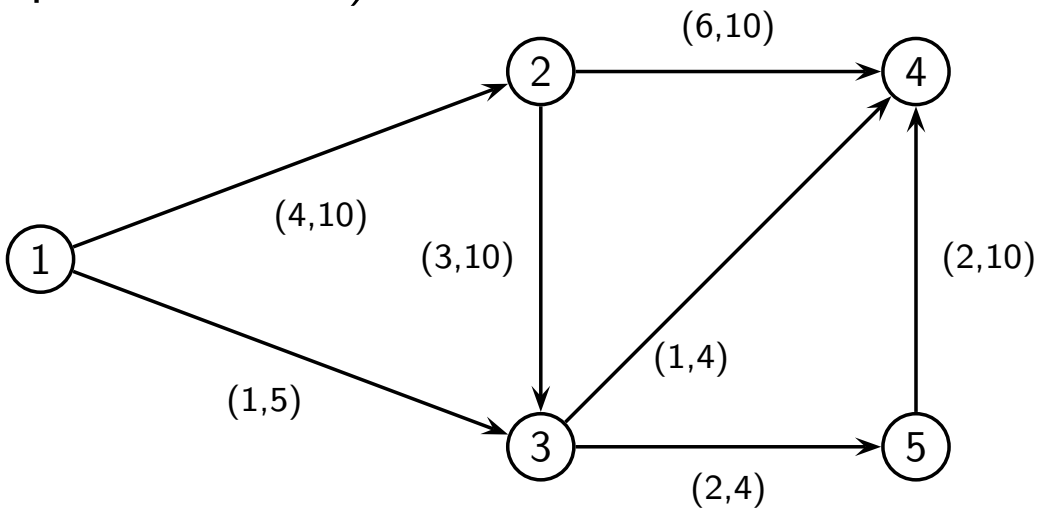
$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 8$.

Algoritmo dei cammini minimi successivi

Iterazione 1. Lo pseudoflusso x iniziale è il seguente (su ogni nodo è indicato lo sbilanciamento):

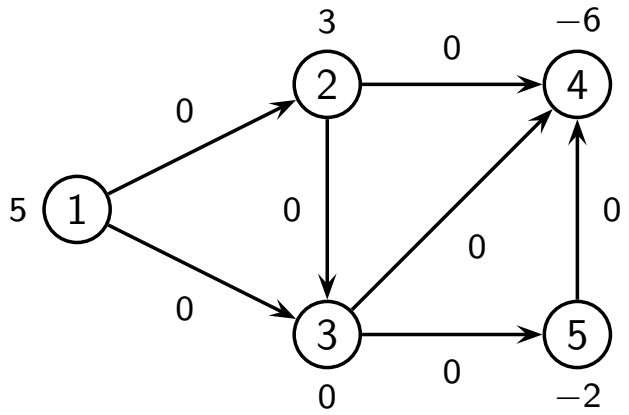


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 8$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

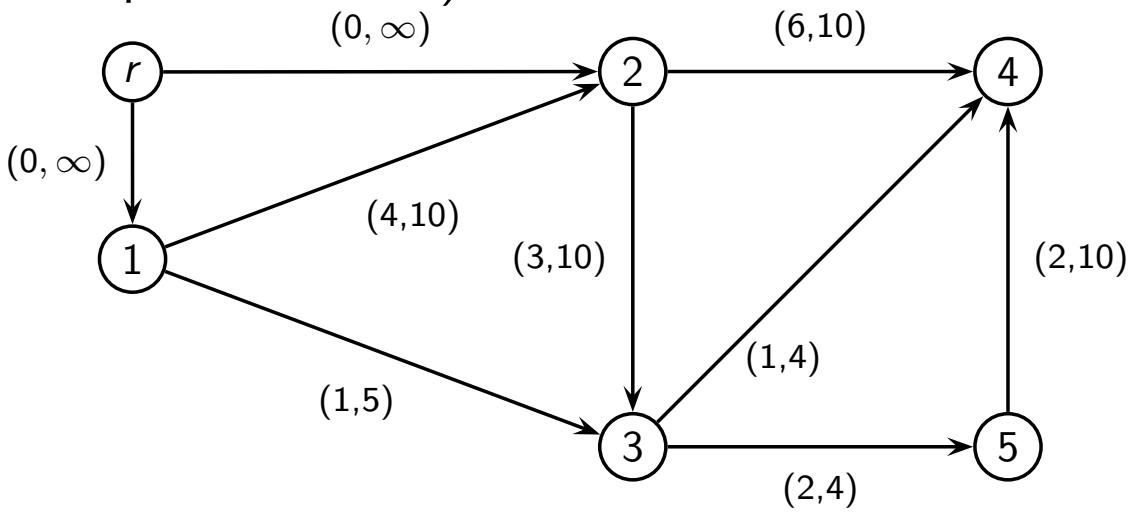


Algoritmo dei cammini minimi successivi

Iterazione 1. Lo pseudoflusso x iniziale è il seguente (su ogni nodo è indicato lo sbilanciamento):

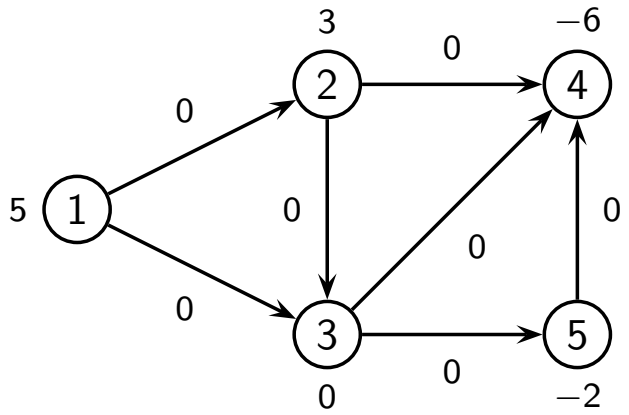


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 8$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

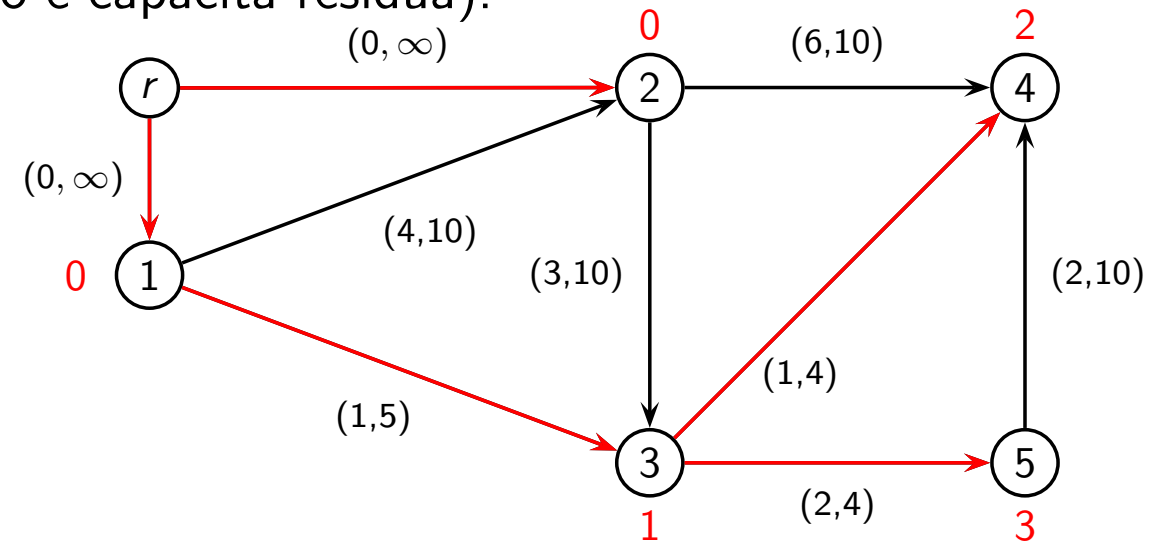


Algoritmo dei cammini minimi successivi

Iterazione 1. Lo pseudoflusso x iniziale è il seguente (su ogni nodo è indicato lo sbilanciamento):

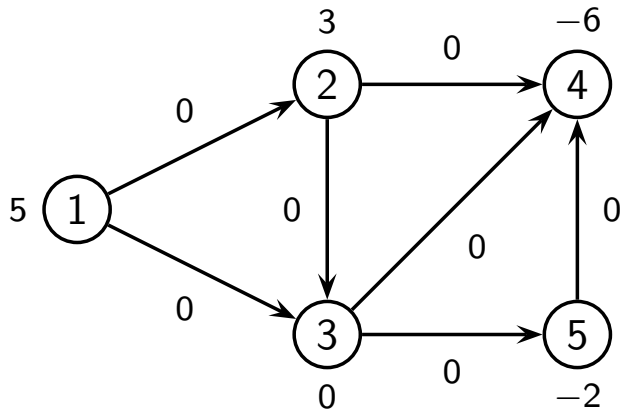


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 8$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

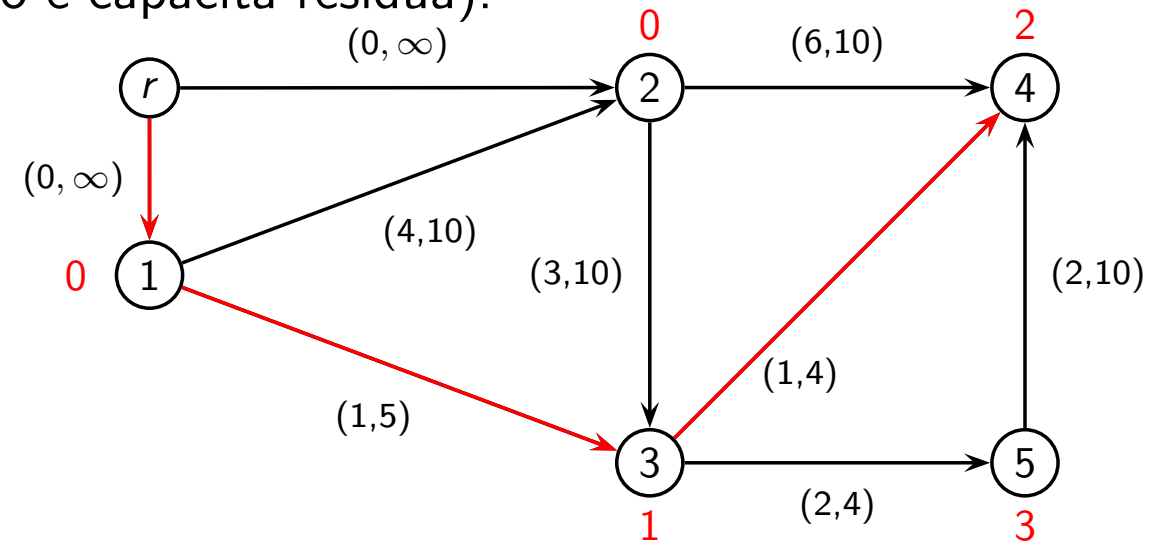


Algoritmo dei cammini minimi successivi

Iterazione 1. Lo pseudoflusso x iniziale è il seguente (su ogni nodo è indicato lo sbilanciamento):



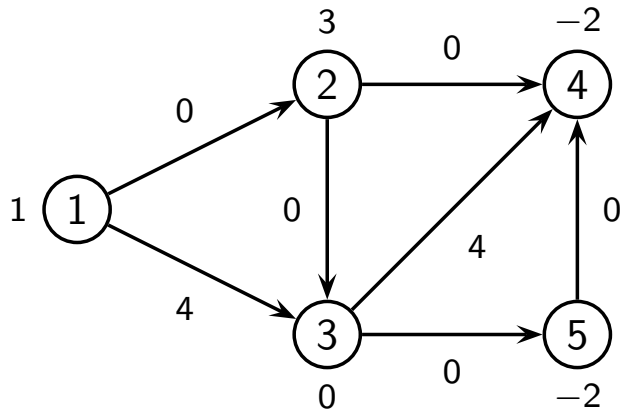
$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 8$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):



$t = 4$, P è il cammino r-1-3-4 passante per $s = 1$, spedisco $\delta = \min\{5, 6, 4\} = 4$.

Algoritmo dei cammini minimi successivi

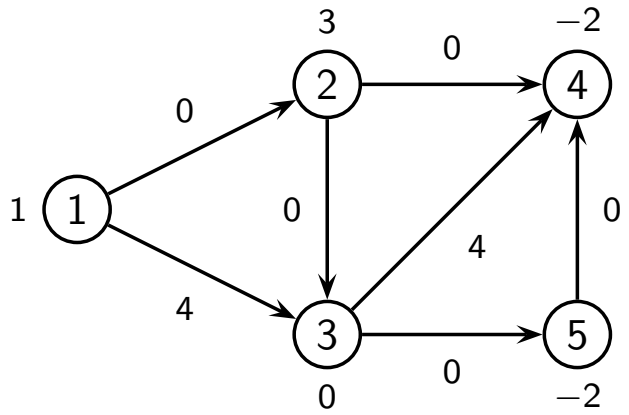
Iterazione 2. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



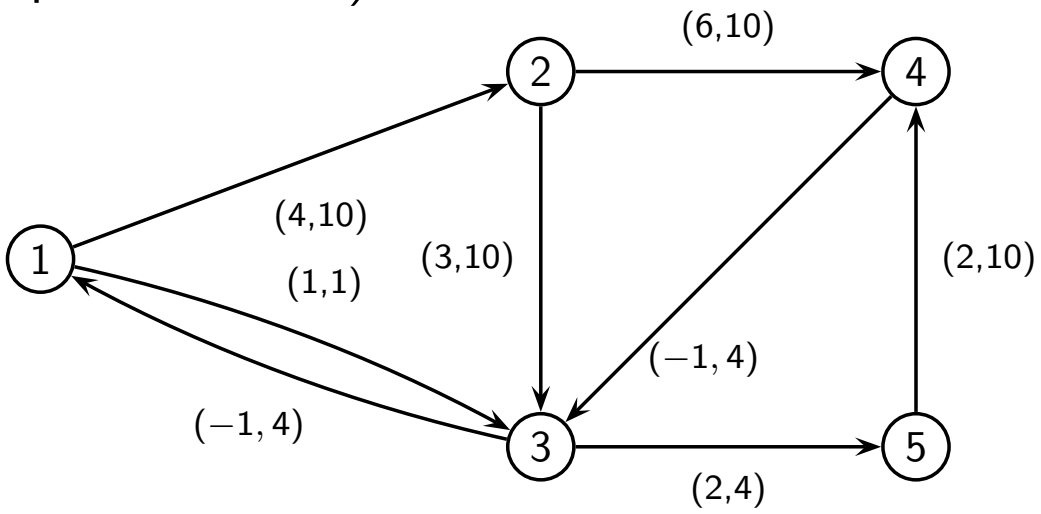
$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 4$.

Algoritmo dei cammini minimi successivi

Iterazione 2. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

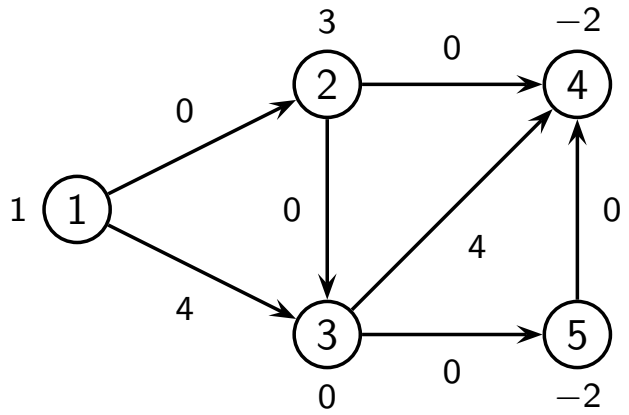


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 4$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

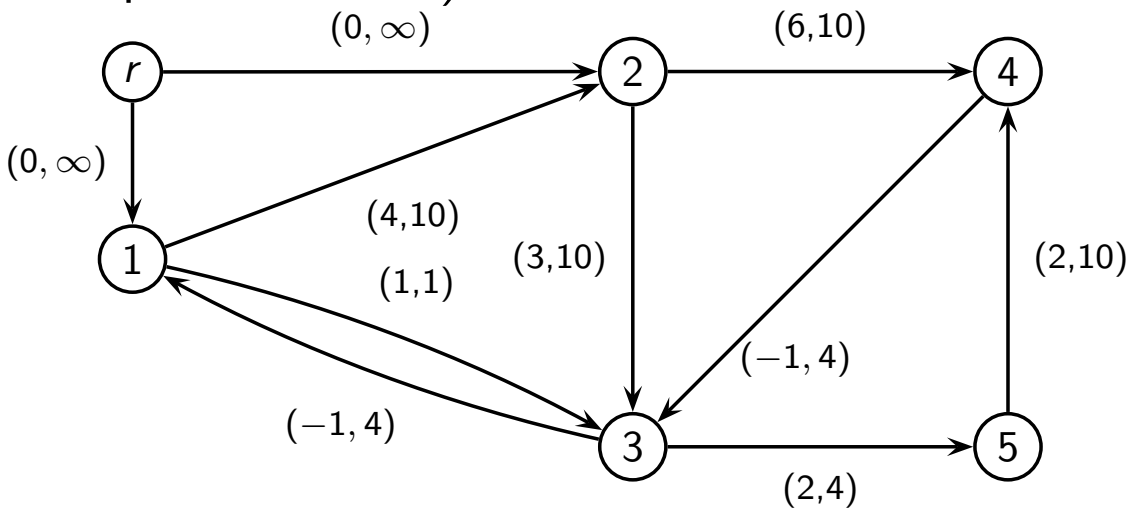


Algoritmo dei cammini minimi successivi

Iterazione 2. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

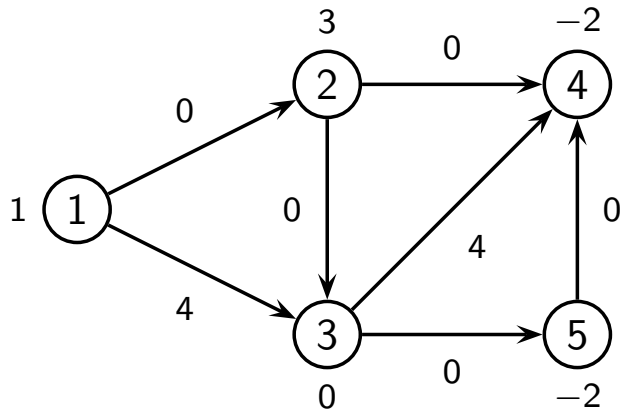


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 4$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

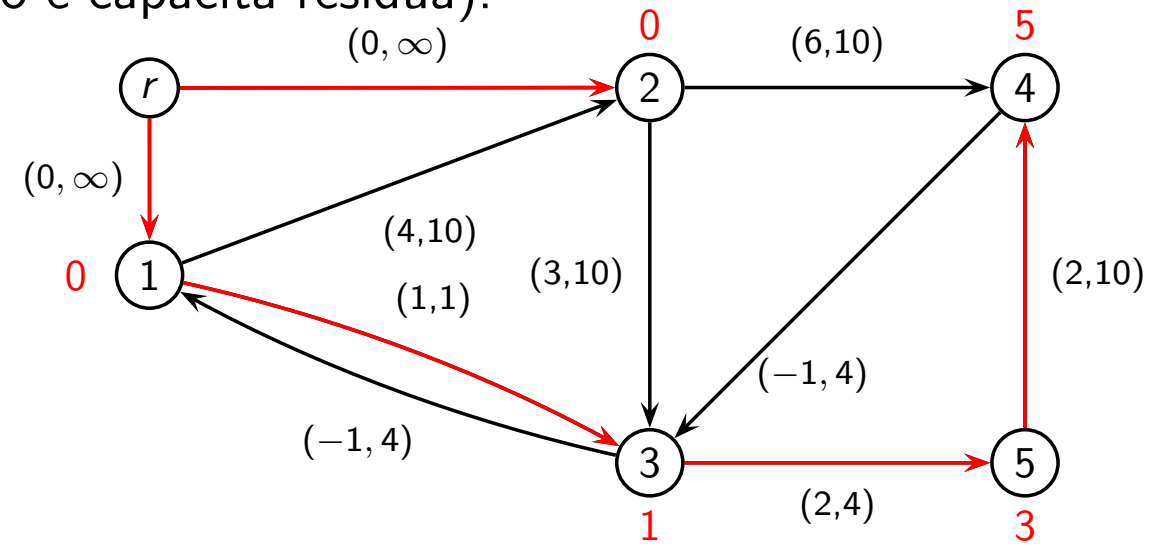


Algoritmo dei cammini minimi successivi

Iterazione 2. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

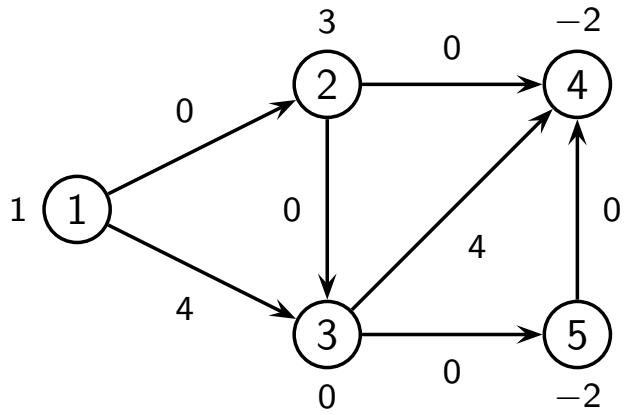


$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 4$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

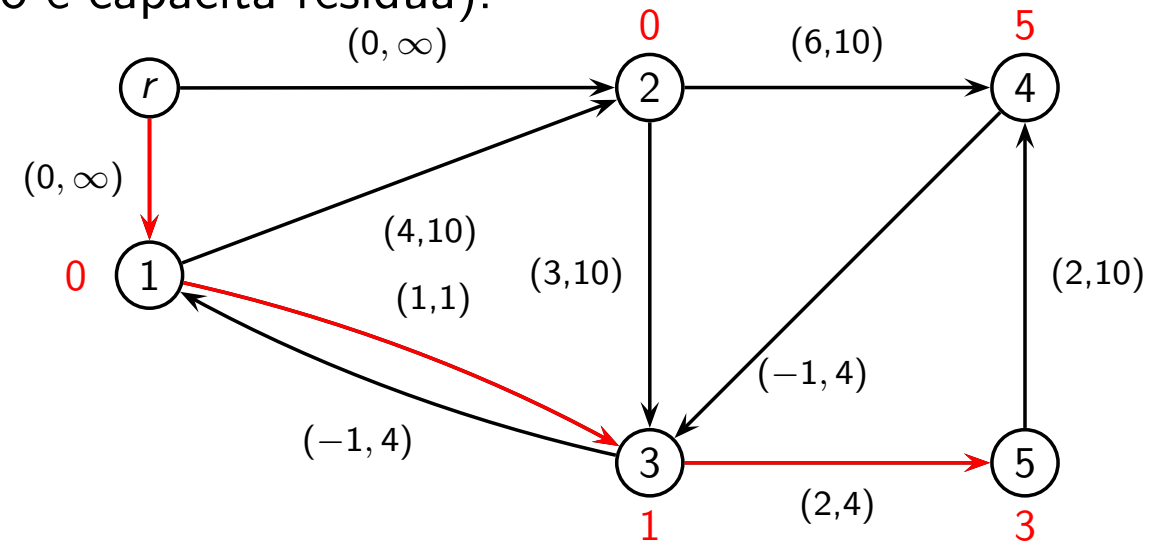


Algoritmo dei cammini minimi successivi

Iterazione 2. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



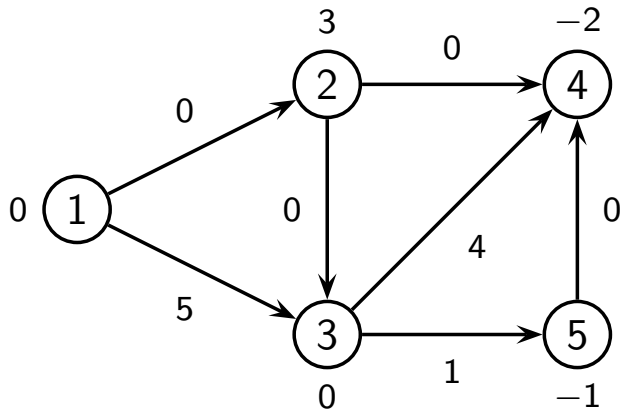
$E_x = \{1, 2\}$, $D_x = \{4, 5\}$, $g(x) = 4$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):



$t = 5$, P è il cammino r-1-3-5 passante per $s = 1$, spedisco $\delta = \min\{1, 2, 1\} = 1$.

Algoritmo dei cammini minimi successivi

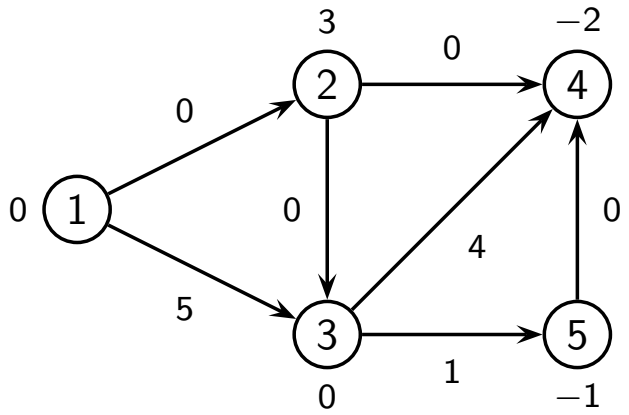
Iterazione 3. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



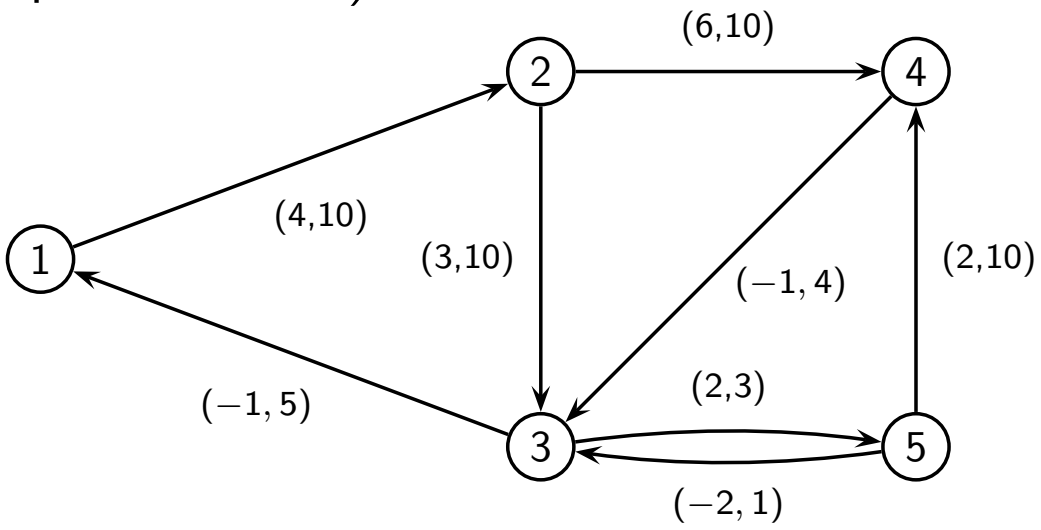
$E_x = \{2\}$, $D_x = \{4, 5\}$, $g(x) = 3$.

Algoritmo dei cammini minimi successivi

Iterazione 3. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

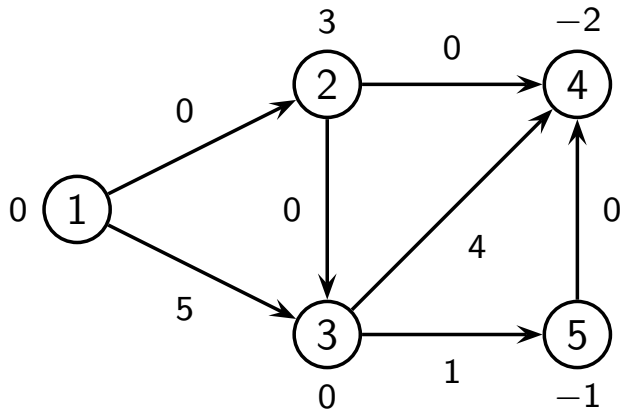


$E_x = \{2\}$, $D_x = \{4, 5\}$, $g(x) = 3$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

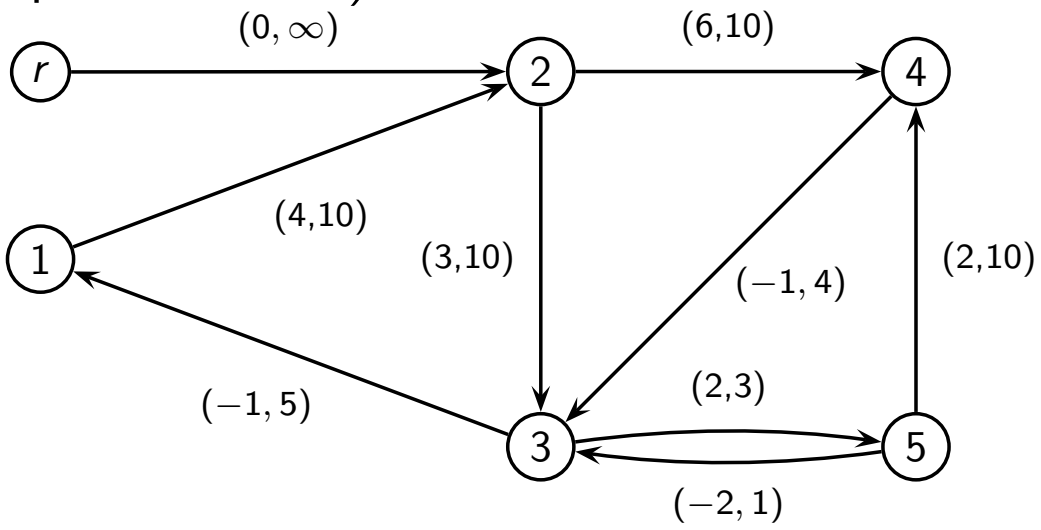


Algoritmo dei cammini minimi successivi

Iterazione 3. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

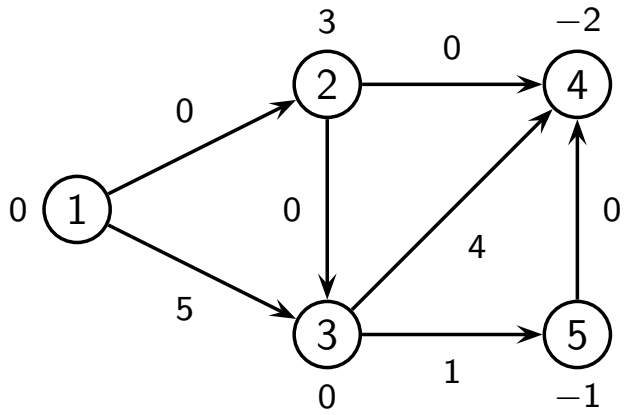


$E_x = \{2\}$, $D_x = \{4, 5\}$, $g(x) = 3$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

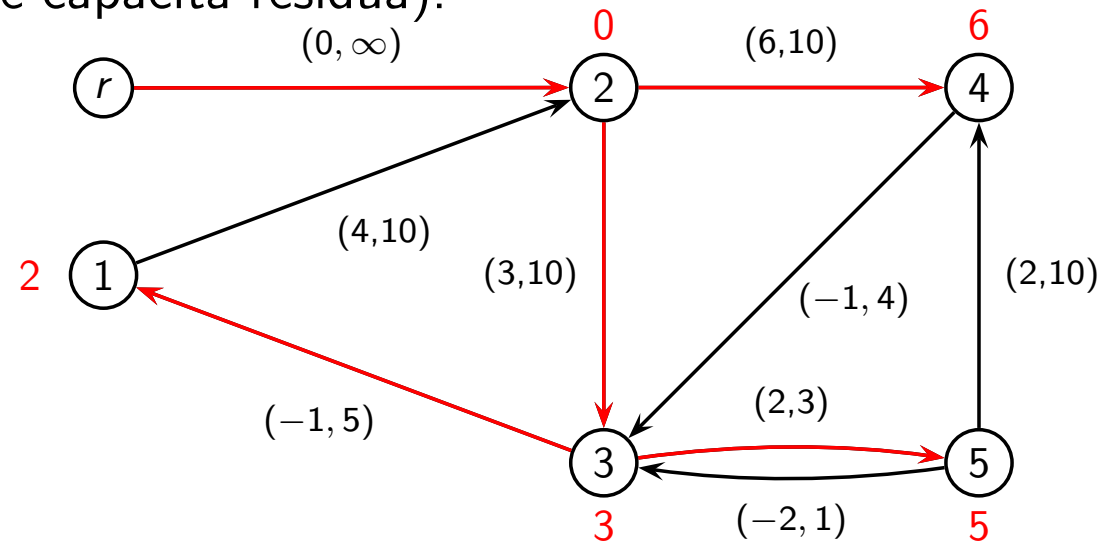


Algoritmo dei cammini minimi successivi

Iterazione 3. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

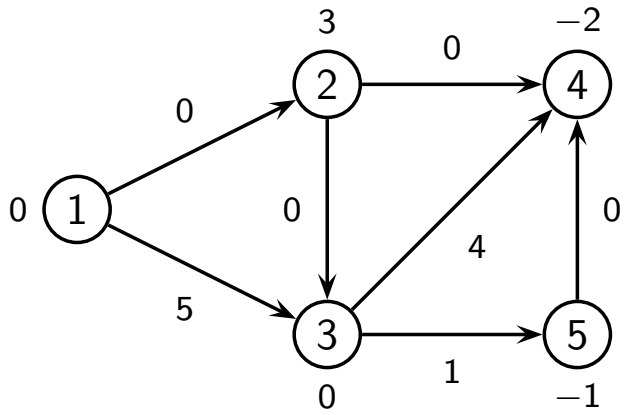


$E_x = \{2\}$, $D_x = \{4, 5\}$, $g(x) = 3$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

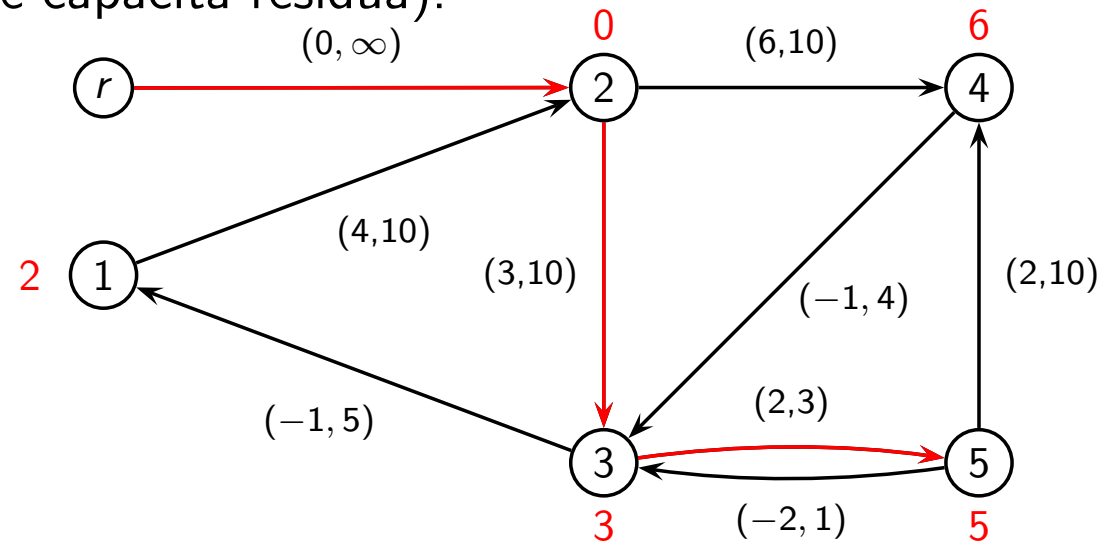


Algoritmo dei cammini minimi successivi

Iterazione 3. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



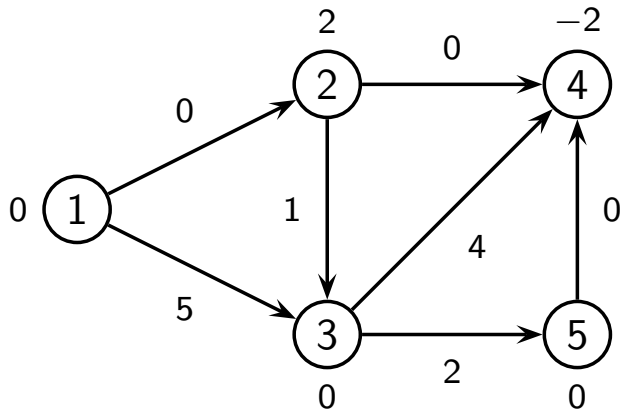
$E_x = \{2\}$, $D_x = \{4, 5\}$, $g(x) = 3$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):



$t = 5$, P è il cammino r-2-3-5 passante per $s = 2$, spedisco $\delta = \min\{3, 1, 3\} = 1$.

Algoritmo dei cammini minimi successivi

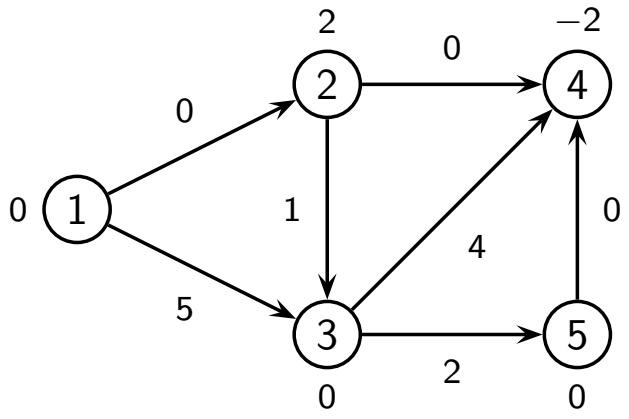
Iterazione 4. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



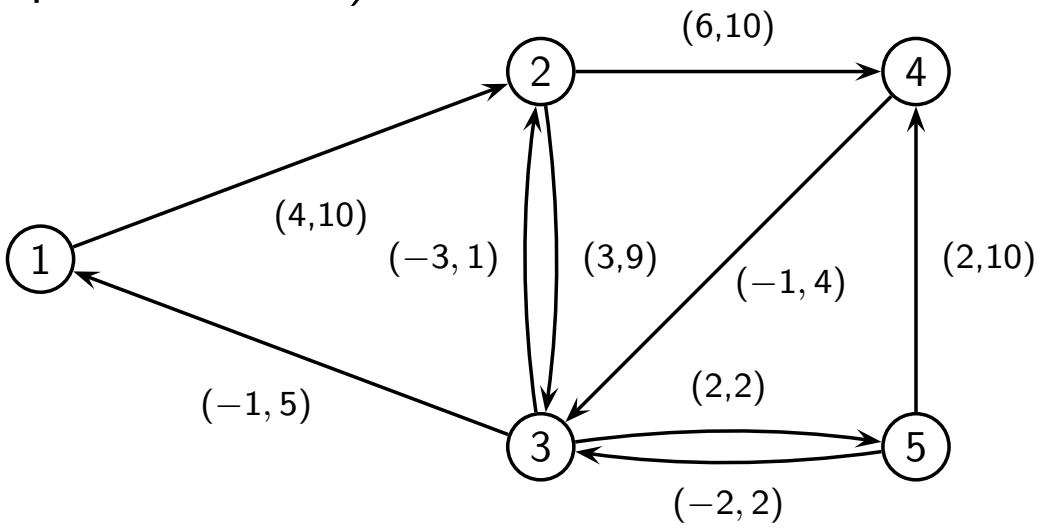
$E_x = \{2\}$, $D_x = \{4\}$, $g(x) = 2$.

Algoritmo dei cammini minimi successivi

Iterazione 4. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

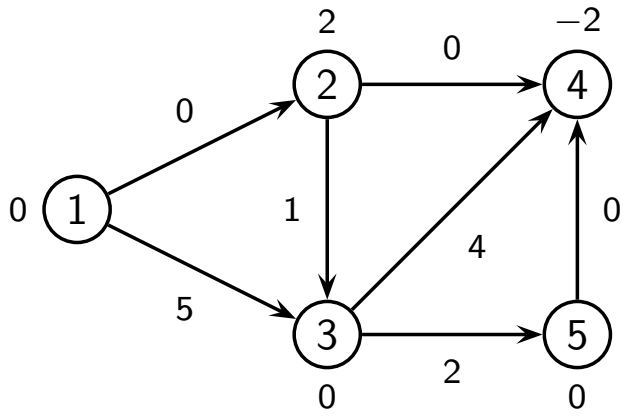


$E_x = \{2\}$, $D_x = \{4\}$, $g(x) = 2$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

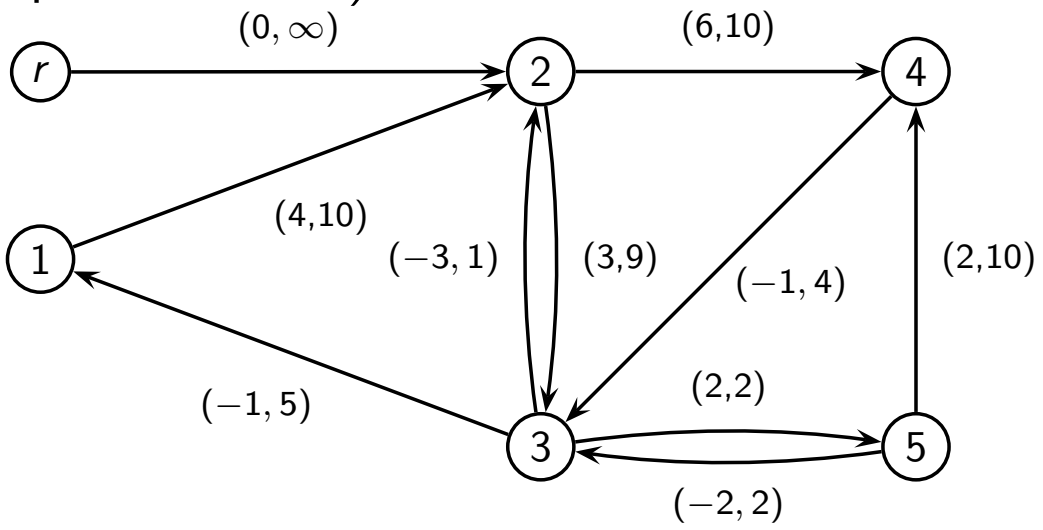


Algoritmo dei cammini minimi successivi

Iterazione 4. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

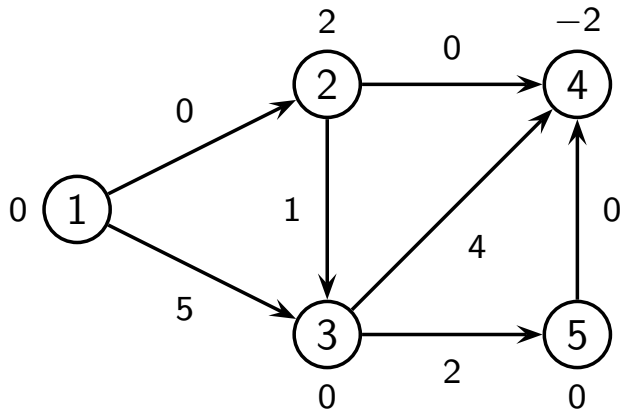


$E_x = \{2\}$, $D_x = \{4\}$, $g(x) = 2$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

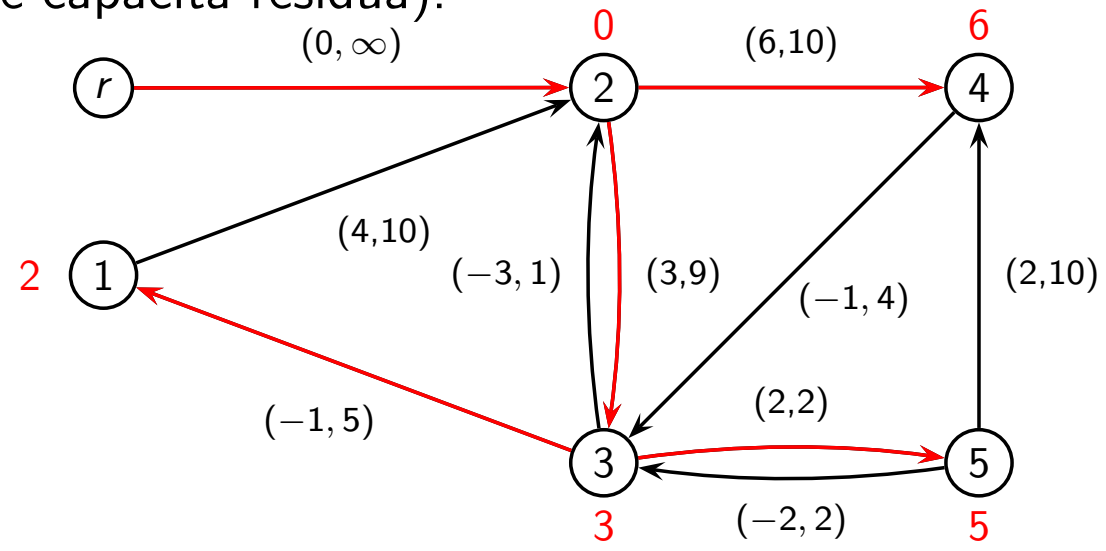


Algoritmo dei cammini minimi successivi

Iterazione 4. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):

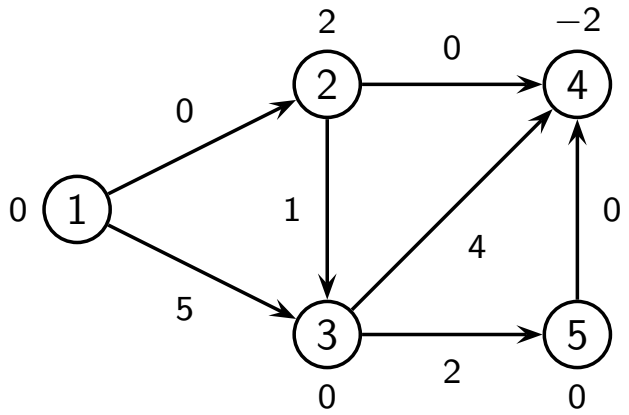


$E_x = \{2\}$, $D_x = \{4\}$, $g(x) = 2$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):

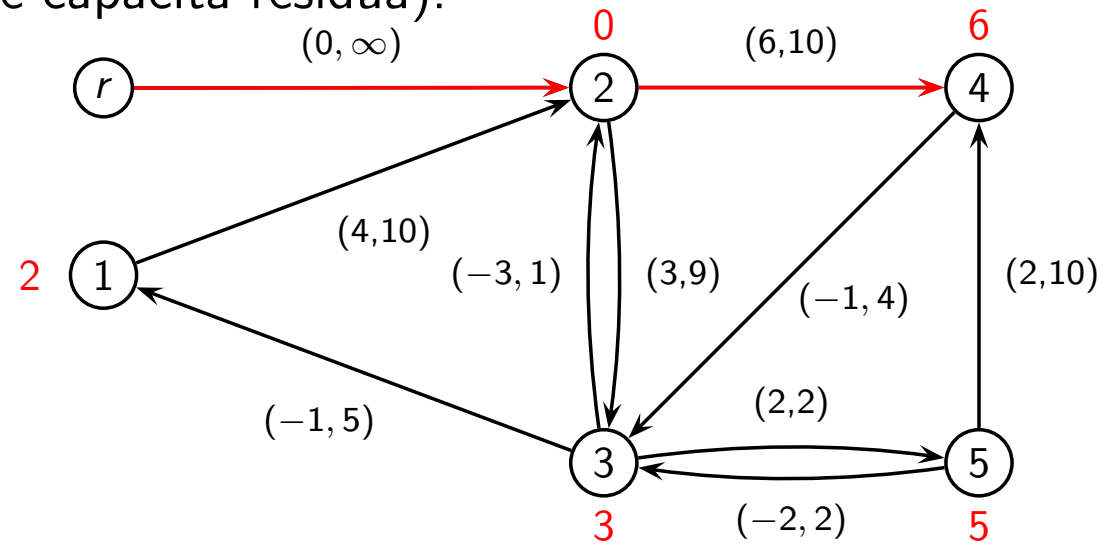


Algoritmo dei cammini minimi successivi

Iterazione 4. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



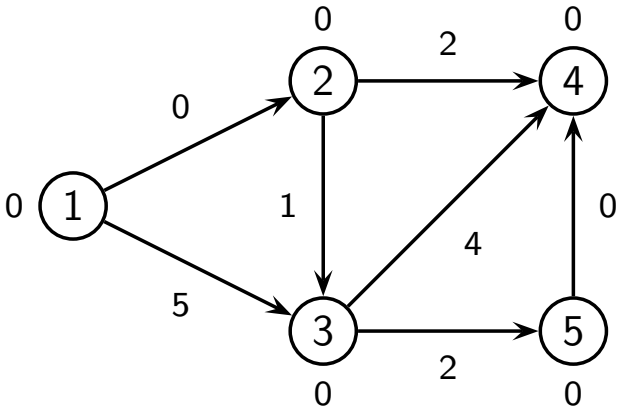
$E_x = \{2\}$, $D_x = \{4\}$, $g(x) = 2$. Il grafo residuo è il seguente (su ogni arco indichiamo costo e capacità residua):



$t = 4$, P è il cammino r-2-4 passante per $s = 2$, spedisco $\delta = \min\{2, 2, 10\} = 2$.

Algoritmo dei cammini minimi successivi

Iterazione 5. Lo pseudoflusso x è il seguente (su ogni nodo è indicato lo sbilanciamento):



$E_x = D_x = \emptyset$, $g(x) = 0$, stop, x è un flusso ottimo.

6 - Programmazione lineare

Mauro Passacantando

Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2020/21

Problema del contadino

Un coltivatore ha a disposizione 12 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg di semi di lattuga, 18 t di tuberi e 160 t di concime. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata per la coltivazione di lattuga è di 3000 €/ettaro e quella delle patate è di 5000 €/ettaro. L'assorbimento delle risorse per ogni tipo di coltivazione è di 7 kg di semi e 10 t di concime per ettaro di lattuga, 3 t di tuberi e 20 t di concime per ettaro di patate. Stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.

Problema del contadino - modello di PL

Variabili decisionali:
x_L = numero di ettari da coltivare a lattuga
x_P = numero di ettari da coltivare a patate

Modello di programmazione lineare:

$$\begin{cases} \max\ 3000x_L + 5000x_P \\ x_L + x_P \leq 12 \\ 7x_L \leq 70 \\ 3x_P \leq 18 \\ 10x_L + 20x_P \leq 160 \\ x_L \geq 0 \\ x_P \geq 0 \end{cases}$$

Forma matriciale

$$\begin{aligned} \max\ 3x_L + 5x_P & \\ x_L + x_P &\leq 12 \\ x_L &\leq 10 \\ x_P &\leq 6 \\ x_L + 2x_P &\leq 16 \\ -x_L &\leq 0 \\ -x_P &\leq 0 \end{aligned}$$

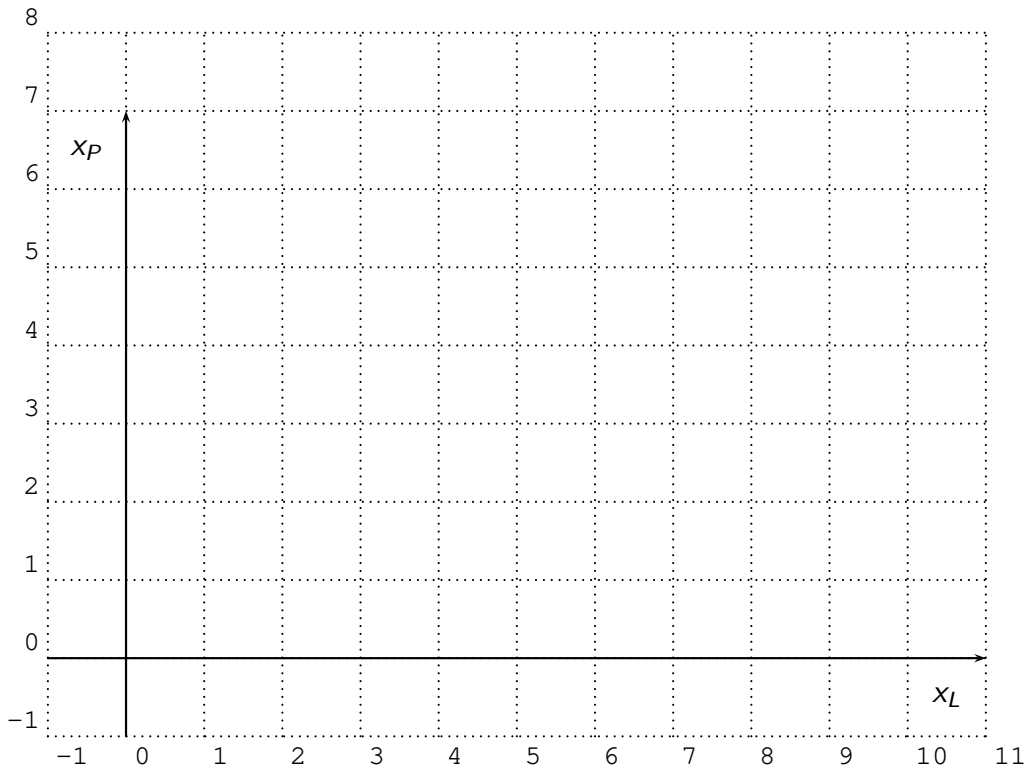
$$\begin{cases} \max\ c^T x \\ Ax \leq b \end{cases} \quad \text{dove } x = \begin{pmatrix} x_L \\ x_P \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 12 \\ 10 \\ 6 \\ 16 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$m =$ numero di vincoli
$n =$ numero di variabili
A matrice $m \times n$
b vettore con m componenti
c vettore con n componenti

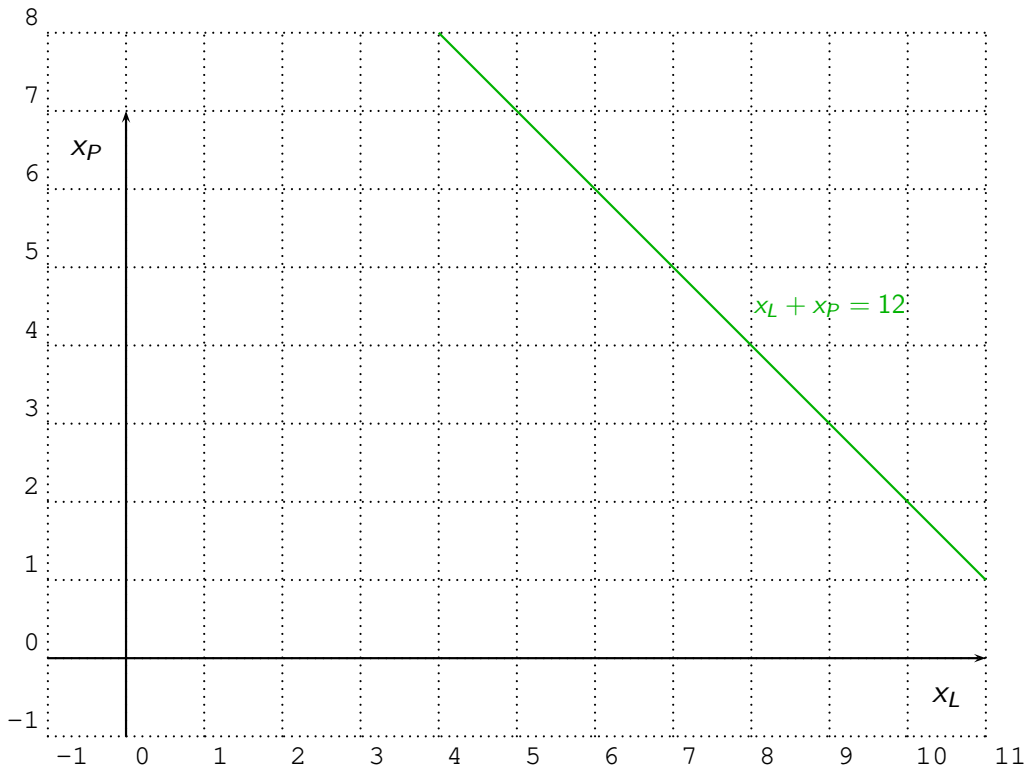
Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



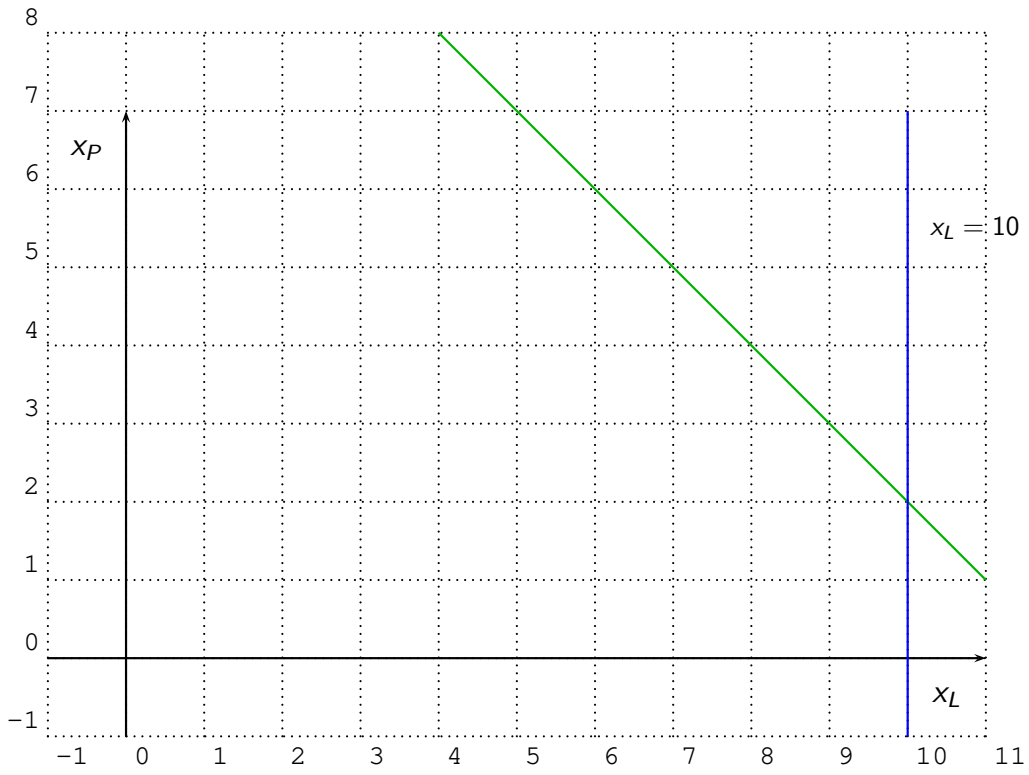
Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



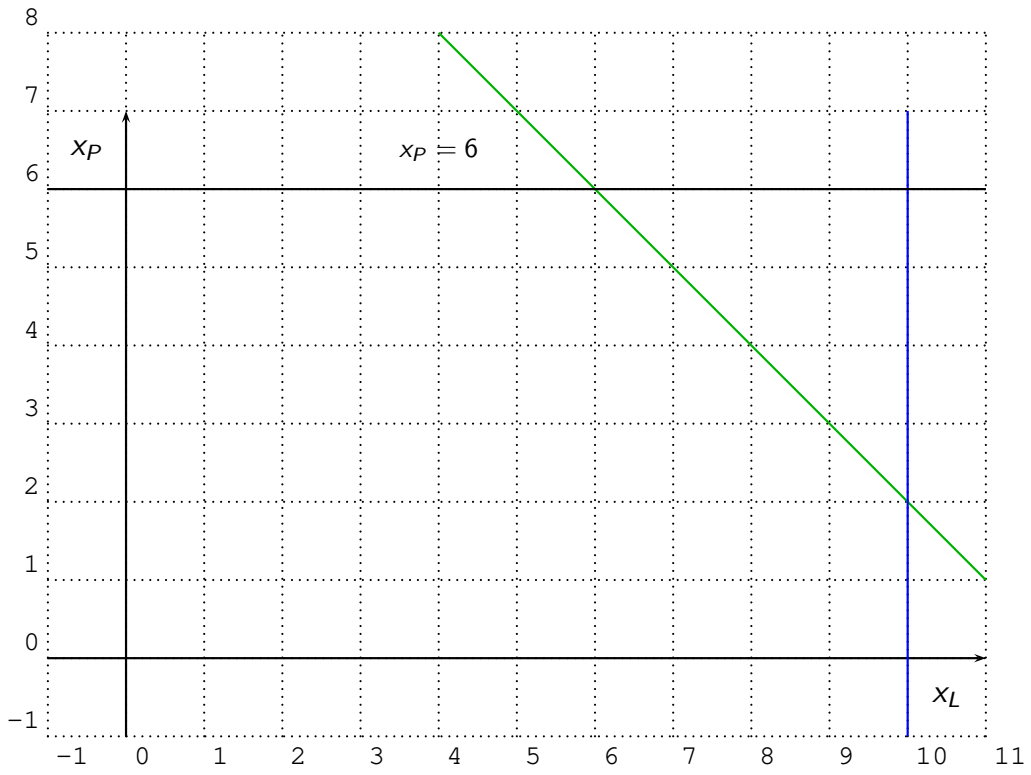
Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



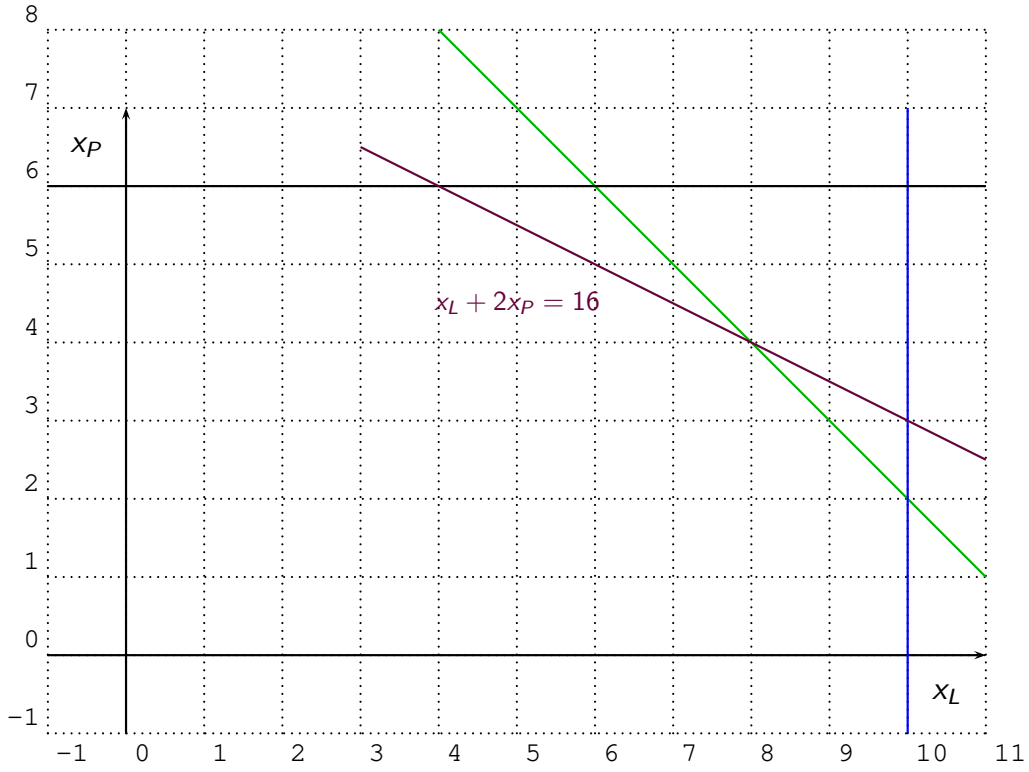
Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



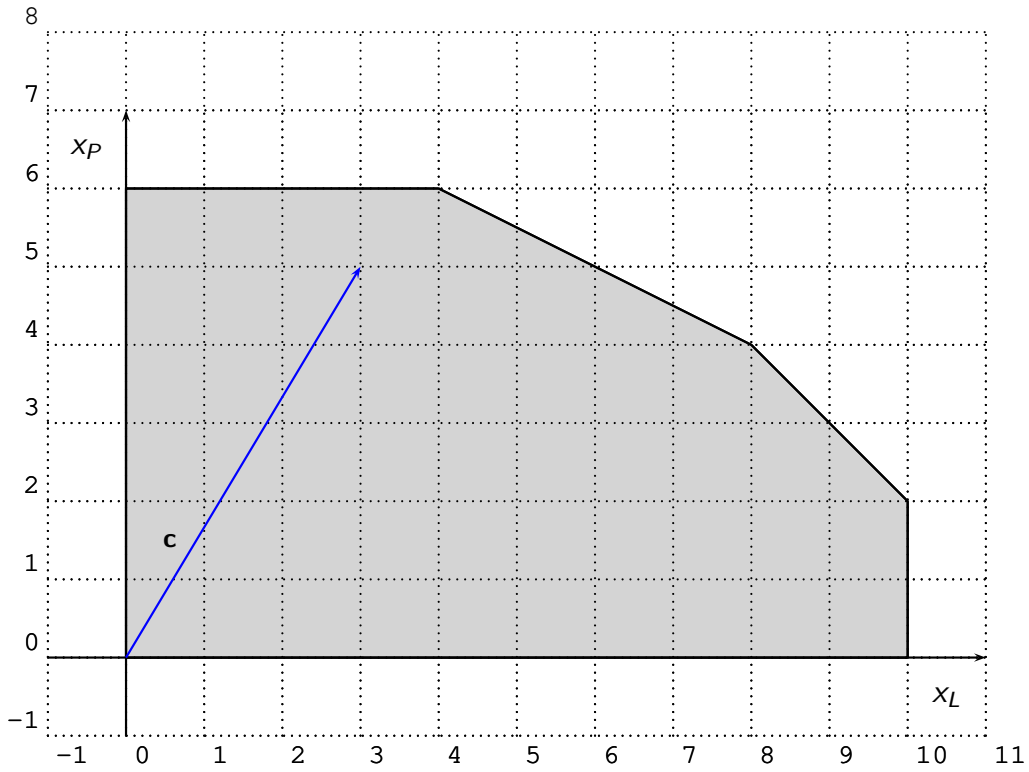
Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$

Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

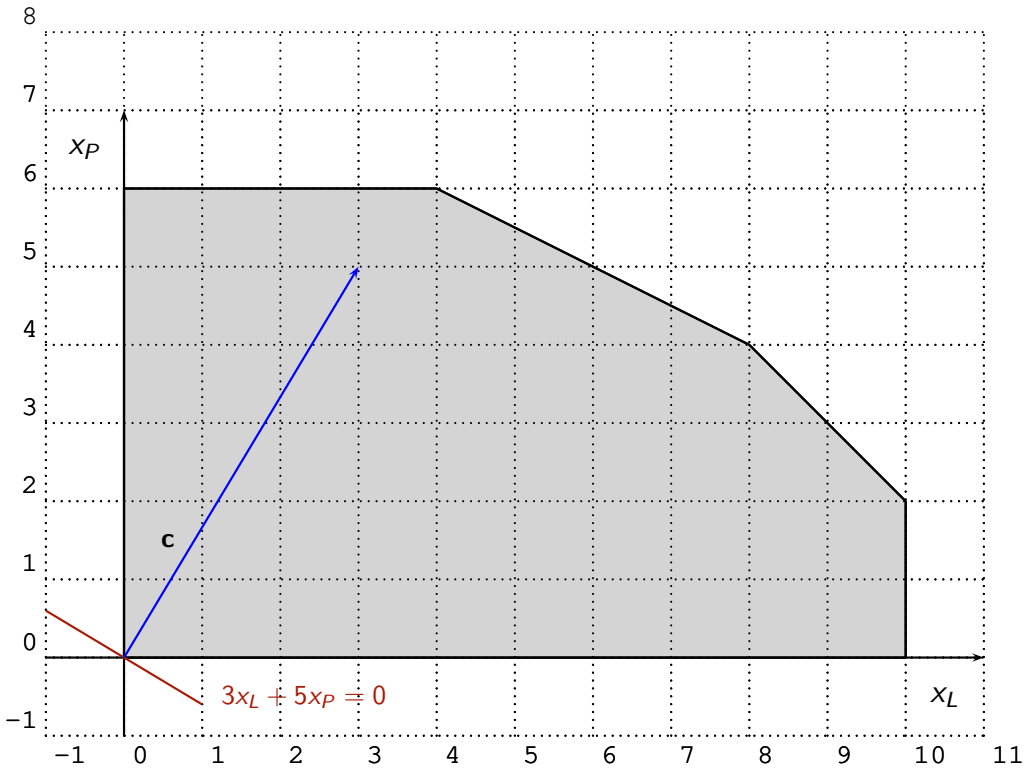
$$
\begin{aligned}
\max \ 3x_L + 5x_P & \\
x_L + x_P & \leq 12 \\
x_L & \leq 10 \\
x_P & \leq 6 \\
x_L + 2x_P & \leq 16 \\
-x_L & \leq 0 \\
-x_P & \leq 0
\end{aligned}
$$



Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

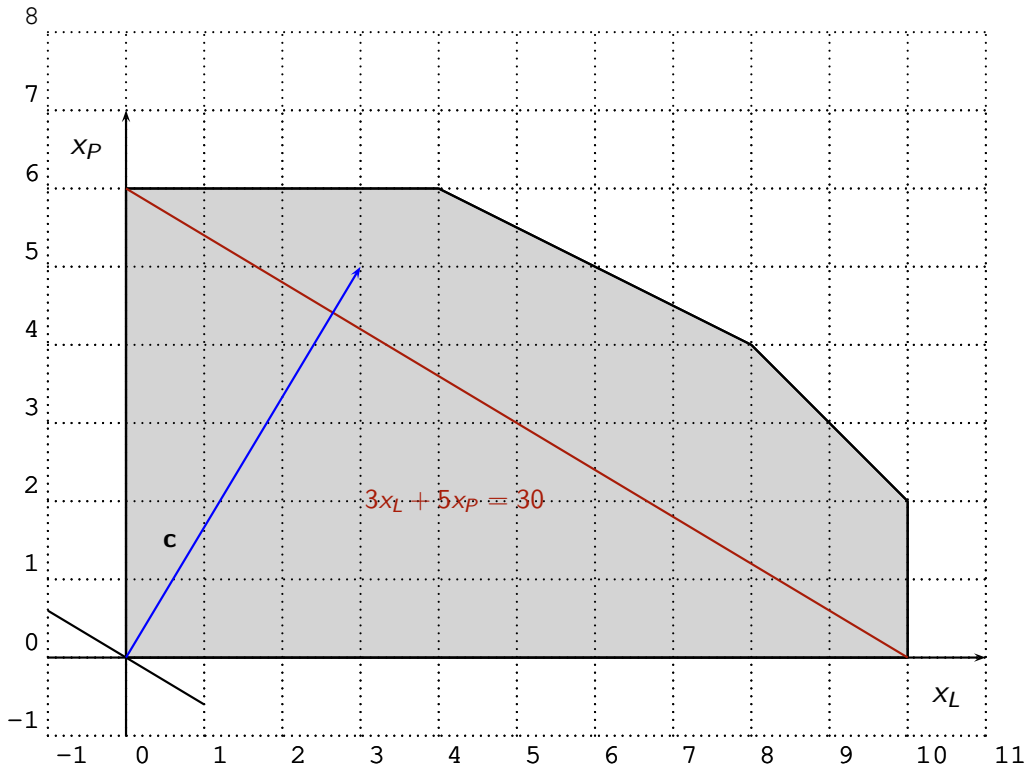
$$
\begin{array}{rl}
\max \ 3x_L + 5x_P & \\
x_L + x_P & \leq 12 \\
x_L & \leq 10 \\
x_P & \leq 6 \\
x_L + 2x_P & \leq 16 \\
-x_L & \leq 0 \\
-x_P & \leq 0
\end{array}
$$



Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

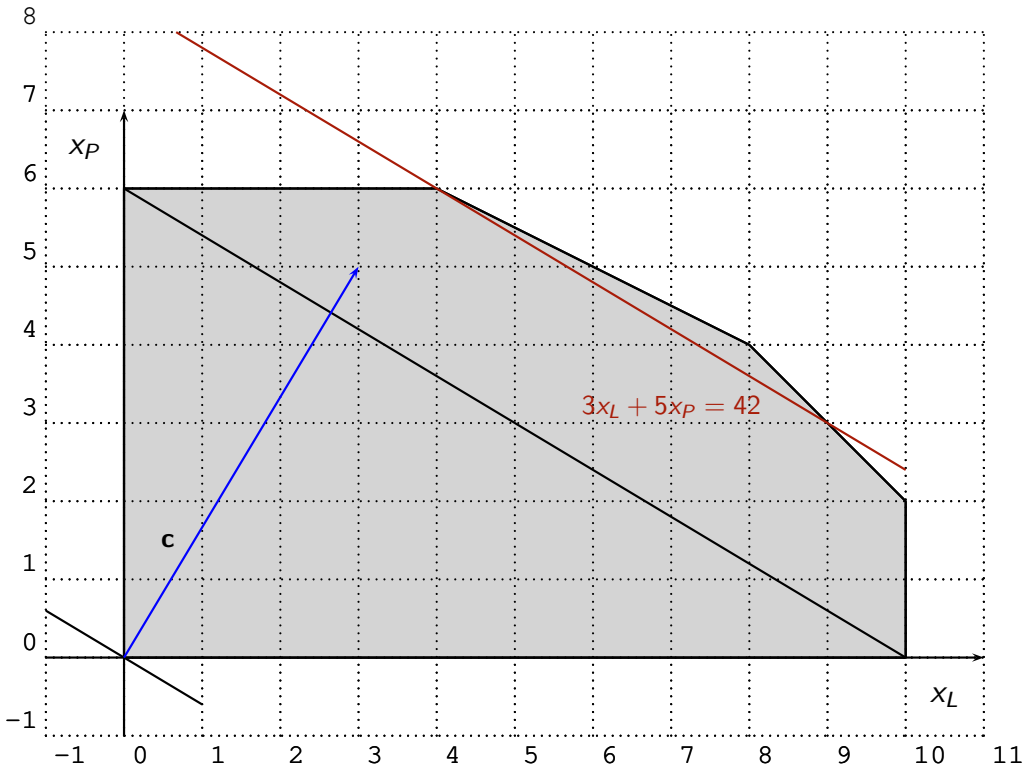
$$
\begin{aligned}
\max \ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

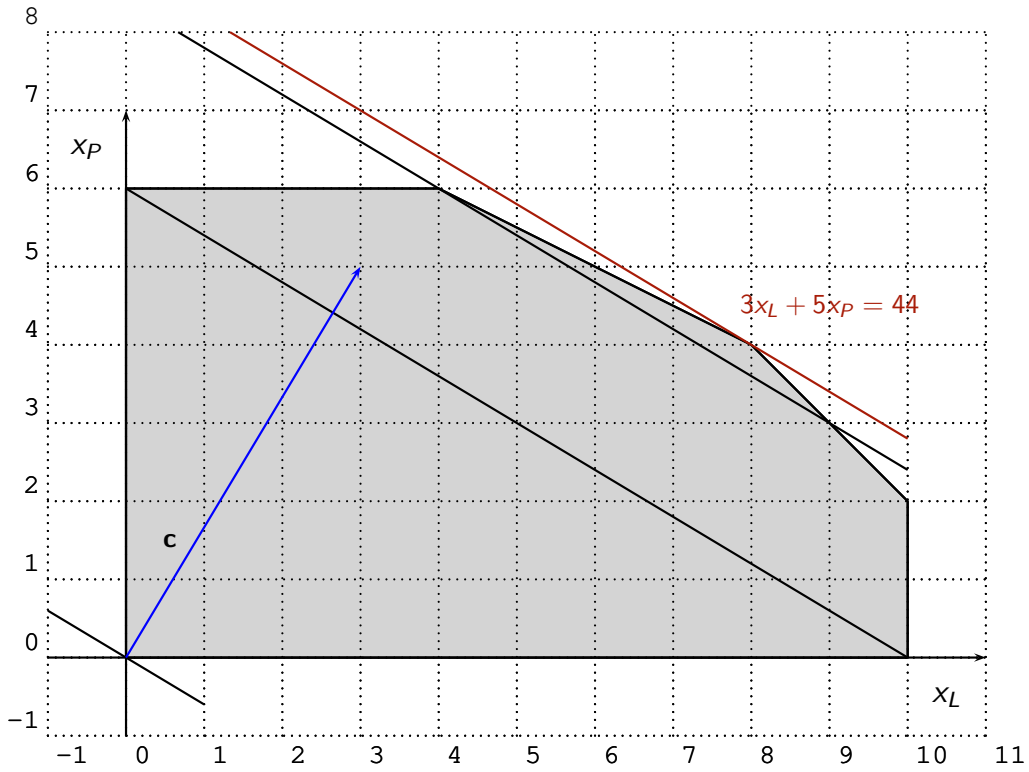
$$
\begin{aligned}
\max \ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

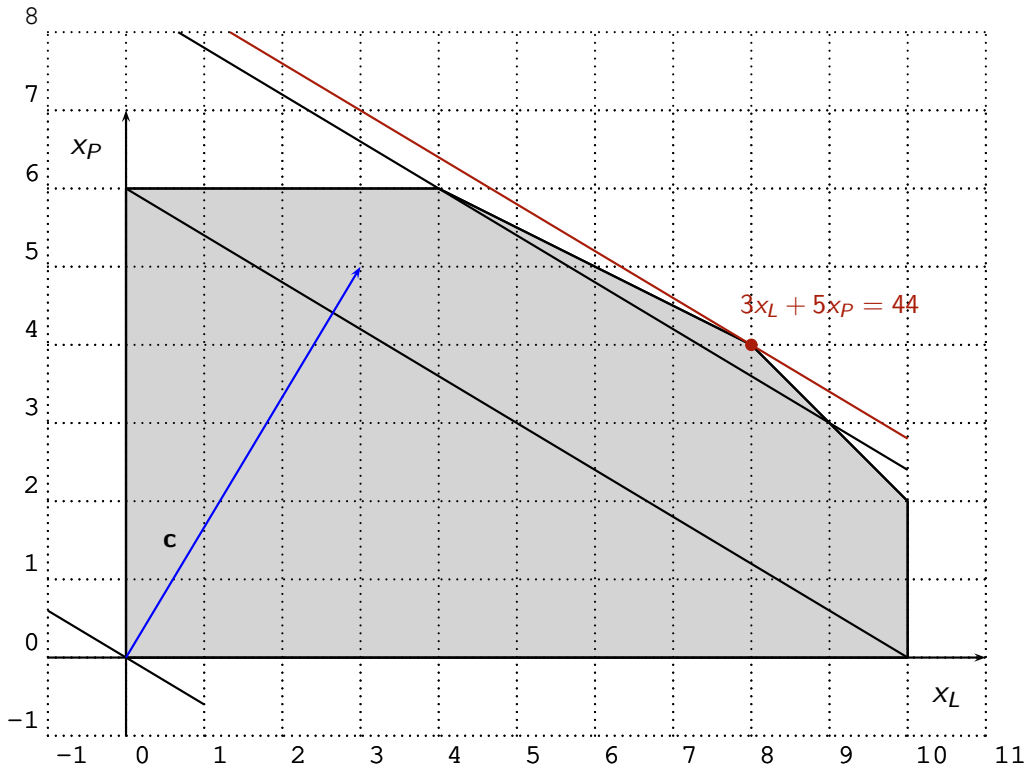
$$
\begin{aligned}
\max \ 3x_L + 5x_P & \\
x_L + x_P &\leq 12 \\
x_L &\leq 10 \\
x_P &\leq 6 \\
x_L + 2x_P &\leq 16 \\
-x_L &\leq 0 \\
-x_P &\leq 0
\end{aligned}
$$



Risoluzione grafica del problema

Gli insiemi di livello della funzione obiettivo sono $L(v) = \{x \in \mathbb{R}^n : \ c^T x = v\}$, dove $v \in \mathbb{R}$ è un valore fissato.

$$
\begin{array}{rl}
\max \ 3x_L + 5x_P & \\
x_L + x_P & \leq 12 \\
x_L & \leq 10 \\
x_P & \leq 6 \\
x_L + 2x_P & \leq 16 \\
-x_L & \leq 0 \\
-x_P & \leq 0
\end{array}
$$



Soluzione ottima: $x_L = 8, x_P = 4$ (8 ettari di lattuga, 4 ettari di patate)
Valore ottimo $= 44$ (ricavo massimo 44000 €)

Forma generale e forma canonica

Definizione

Un problema di Programmazione Lineare (PL) consiste nel trovare il massimo o il minimo di una funzione lineare di n variabili reali soggette a vincoli lineari di uguaglianza o di disuguaglianza, cioè

$$\begin{cases} \max(\text{o } \min)\ c^T x \\ A_1 x \leq b_1 \\ A_2 x \geq b_2 \\ A_3 x = b_3 \\ (x \in \mathbb{R}^n) \end{cases}$$

Forma generale e forma canonica

Definizione

Un problema di Programmazione Lineare (PL) consiste nel trovare il massimo o il minimo di una funzione lineare di n variabili reali soggette a vincoli lineari di uguaglianza o di disuguaglianza, cioè

$$\begin{cases} \max(\text{o } \min)\ c^T x \\ A_1 x \leq b_1 \\ A_2 x \geq b_2 \\ A_3 x = b_3 \\ (x \in \mathbb{R}^n) \end{cases}$$

Definizione

Un problema nella forma

$$\begin{cases} \max\ c^T x \\ Ax \leq b \end{cases}$$

è chiamato problema di PL in forma canonica.

Forma generale e forma canonica

Teorema

Ogni problema di PL può essere riscritto in modo equivalente in forma canonica.

Dim. $\min\ c^T x = -\max\,(-c^T x)$

$a^T x \geq b$ è equivalente a $-a^T x \leq -b$

$a^T x = b$ è equivalente a $\begin{cases} a^T x \leq b \\ -a^T x \leq -b \end{cases}$

Esercizio

Scrivere in forma canonica il seguente problema di PL:

$$\begin{cases} \min\ 2x_1 + 5x_2 \\ 6x_1 + 9x_2 = 17 \\ x_1 \geq 0 \\ x_1 + 3x_2 \geq 1 \end{cases}$$

Combinazioni convesse

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione convessa dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \in [0, 1]$, con $\sum_{i=1}^{m} \alpha_i = 1$, tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Combinazioni convesse

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione convessa dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \in [0, 1]$, con $\sum_{i=1}^{m} \alpha_i = 1$, tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(2, 2)$ è combinazione convessa di $(4, 0)$ e $(1, 3)$. Infatti:
$(2, 2) = \frac{1}{3}(4, 0) + \frac{2}{3}(1, 3)$.

Esempio. $(2, 2)$ è combinazione convessa di $(1, 1)$, $(3, 1)$ e $(2, 3)$. Infatti:
$(2, 2) = \frac{1}{4}(1, 1) + \frac{1}{4}(3, 1) + \frac{1}{2}(2, 3)$.

Combinazioni convesse

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione convessa dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \in [0, 1]$, con $\sum_{i=1}^{m} \alpha_i = 1$, tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(2, 2)$ è combinazione convessa di $(4, 0)$ e $(1, 3)$. Infatti:
$(2, 2) = \frac{1}{3}(4, 0) + \frac{2}{3}(1, 3)$.

Esempio. $(2, 2)$ è combinazione convessa di $(1, 1)$, $(3, 1)$ e $(2, 3)$. Infatti:
$(2, 2) = \frac{1}{4}(1, 1) + \frac{1}{4}(3, 1) + \frac{1}{2}(2, 3)$.

Definizione

L'involucro convesso di un insieme $K \subset \mathbb{R}^n$, denotato con $\text{conv}(K)$, è l'insieme di tutte le combinazioni convesse di punti di K.

Esercizio. Qual è l'involucro convesso dei punti $(1, 1)$, $(3, 1)$ e $(2, 3)$?

Combinazioni convesse

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione convessa dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \in [0,1]$, con $\sum_{i=1}^{m} \alpha_i = 1$, tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(2,2)$ è combinazione convessa di $(4,0)$ e $(1,3)$. Infatti:
$(2,2) = \frac{1}{3}(4,0) + \frac{2}{3}(1,3)$.

Esempio. $(2,2)$ è combinazione convessa di $(1,1)$, $(3,1)$ e $(2,3)$. Infatti:
$(2,2) = \frac{1}{4}(1,1) + \frac{1}{4}(3,1) + \frac{1}{2}(2,3)$.

Definizione

L'involucro convesso di un insieme $K \subset \mathbb{R}^n$, denotato con $\mathrm{conv}(K)$, è l'insieme di tutte le combinazioni convesse di punti di K.

Esercizio. Qual è l'involucro convesso dei punti $(1,1)$, $(3,1)$ e $(2,3)$?

Definizione

Un insieme $K \subseteq \mathbb{R}^n$ è detto convesso se per ogni $x, y \in K$ il vettore $\alpha x + (1-\alpha) y \in K$ per ogni $\alpha \in [0,1]$.

Combinazioni coniche

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione conica dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \geq 0$ tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Combinazioni coniche

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione conica dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \geq 0$ tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(4, 4)$ è combinazione conica di $(2, 3)$ e $(3, 1)$. Infatti:
$(4, 4) = \frac{8}{7}(2, 3) + \frac{4}{7}(3, 1)$.

Combinazioni coniche

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione conica dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \geq 0$ tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(4, 4)$ è combinazione conica di $(2, 3)$ e $(3, 1)$. Infatti:
$(4, 4) = \frac{8}{7}(2, 3) + \frac{4}{7}(3, 1)$.

Definizione

L'involucro conico di un insieme $K \subset \mathbb{R}^n$, denotato con $\text{cono}(K)$, è l'insieme di tutte le combinazioni coniche di punti di K.

Combinazioni coniche

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione conica dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \geq 0$ tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(4, 4)$ è combinazione conica di $(2, 3)$ e $(3, 1)$. Infatti:
$(4, 4) = \frac{8}{7}(2, 3) + \frac{4}{7}(3, 1)$.

Definizione

L'involucro conico di un insieme $K \subset \mathbb{R}^n$, denotato con $\text{cono}(K)$, è l'insieme di tutte le combinazioni coniche di punti di K.

Esercizio. Qual è l'involucro conico di $(2, 3)$ e $(3, 1)$?

Combinazioni coniche

Definizione

Un vettore $x \in \mathbb{R}^n$ è detto combinazione conica dei vettori $x^1, \ldots, x^m \in \mathbb{R}^n$ se esistono coefficienti $\alpha_1, \ldots, \alpha_m \geq 0$ tali che $x = \sum_{i=1}^{m} \alpha_i x^i$.

Esempio. $(4, 4)$ è combinazione conica di $(2, 3)$ e $(3, 1)$. Infatti:
$(4, 4) = \frac{8}{7}(2, 3) + \frac{4}{7}(3, 1)$.

Definizione

L'involucro conico di un insieme $K \subset \mathbb{R}^n$, denotato con $\text{cono}(K)$, è l'insieme di tutte le combinazioni coniche di punti di K.

Esercizio. Qual è l'involucro conico di $(2, 3)$ e $(3, 1)$?

Definizione

Un insieme $K \subseteq \mathbb{R}^n$ è detto cono se per ogni $x \in K$ il vettore $\alpha x \in K$ per ogni $\alpha \geq 0$.

Poliedri

L'insieme $\{x \in \mathbb{R}^n : \ a^T x \leq b\}$ è un semispazio chiuso di $\mathbb{R}^n$.

Definizione

Un poliedro in $\mathbb{R}^n$ è l'intersezione di un numero finito di semispazi chiusi, oppure, è l'insieme delle soluzioni di un sistema di disequazioni lineari $Ax \leq b$.
[La regione ammissibile di ogni problema di PL è un poliedro.]

Un poliedro P è detto limitato se esiste $M > 0$ tale che $\|x\| \leq M$ per ogni $x \in P$, ossia se è contenuto in una opportuna sfera centrata nell'origine.

Esempi.

$P_1 = \{x \in \mathbb{R}^2 : \ 1 \leq x_1 \leq 4, \quad 1 \leq x_2 \leq 3\}$ è un poliedro limitato.

$P_2 = \{x \in \mathbb{R}^2 : \ x_1 \geq 1, \quad x_2 \geq 1, \quad x_1 + x_2 \geq 3\}$ è un poliedro illimitato.

Direzioni di recessione

Definizione

Una direzione di recessione di un poliedro P è un vettore d tale che $x + \alpha d \in P$ per ogni $x \in P$ e $\alpha > 0$.
L'insieme di tutte le direzioni di recessione di un poliedro P è un cono convesso ed è denotato con $\text{rec}(P)$.

Esempio. $(1, 1)$ è una direzione di recessione del poliedro

$$P_2 = \{x \in \mathbb{R}^2 : \; x_1 \geq 1, \qquad x_2 \geq 1, \qquad x_1 + x_2 \geq 3\}.$$

Direzioni di recessione

Definizione

Una direzione di recessione di un poliedro P è un vettore d tale che $x + \alpha d \in P$ per ogni $x \in P$ e $\alpha > 0$.
L'insieme di tutte le direzioni di recessione di un poliedro P è un cono convesso ed è denotato con $\text{rec}(P)$.

Esempio. $(1, 1)$ è una direzione di recessione del poliedro

$$P_2 = \{x \in \mathbb{R}^2 : \ x_1 \geq 1, \qquad x_2 \geq 1, \qquad x_1 + x_2 \geq 3\}.$$

Teorema

Se un poliedro $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$, allora $\text{rec}(P) = \{x \in \mathbb{R}^n : \ Ax \leq 0\}$.

Direzioni di linealità

Definizione

Una direzione di linealità di un poliedro P è un vettore d tale che $x + \alpha d \in P$ per ogni $x \in P$ e $\alpha \in \mathbb{R}$.
L'insieme di tutte le direzioni di linealità di un poliedro P è un sottospazio vettoriale (detto spazio di linealità) ed è denotato con lineal(P).

Esempio. $(1, 0)$ è una direzione di linealità del poliedro

$$P_3 = \{x \in \mathbb{R}^2 : \ 1 \leq x_2 \leq 2\}.$$

Direzioni di linealità

Definizione

Una direzione di linealità di un poliedro P è un vettore d tale che $x + \alpha d \in P$ per ogni $x \in P$ e $\alpha \in \mathbb{R}$.
L'insieme di tutte le direzioni di linealità di un poliedro P è un sottospazio vettoriale (detto spazio di linealità) ed è denotato con lineal(P).

Esempio. $(1, 0)$ è una direzione di linealità del poliedro

$$P_3 = \{x \in \mathbb{R}^2 : \ 1 \leq x_2 \leq 2\}.$$

Teorema

Se un poliedro $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$, allora lineal$(P) = \{x \in \mathbb{R}^n : \ Ax = 0\}$.

Esercizio.
Consideriamo un poliedro $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$, dove A è una matrice $m \times n$. Dimostrare che lineal$(P) = \{0\}$ se e solo se rank$(A) = n$ (cioè le colonne di A sono linearmente indipendenti).

Vertici

Definizione

Un punto x di un poliedro P è chiamato vertice se non esistono due punti $y, z \in P$ diversi da x tali che x è combinazione convessa di y e z.

Esempi.

I vertici di $P_1 = \{x \in \mathbb{R}^2 : \ 1 \leq x_1 \leq 4, \ 1 \leq x_2 \leq 3\}$ sono $(1,1)$, $(1,3)$, $(4,1)$ e $(4,3)$.

I vertici di $P_2 = \{x \in \mathbb{R}^2 : \ x_1 \geq 1, \quad x_2 \geq 1, \quad x_1 + x_2 \geq 3\}$ sono $(1,2)$ e $(2,1)$.

Il poliedro $P_3 = \{x \in \mathbb{R}^2 : \ 1 \leq x_2 \leq 2\}$ non ha vertici.

Teorema

Un poliedro P ha almeno un vertice se e solo se $\text{lineal}(P) = \{0\}$.

Corollario

Un poliedro $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$, dove A è una matrice $m \times n$, ha almeno un vertice se e solo se $\text{rank}(A) = n$.

Teorema di decomposizione dei poliedri

Teorema

Se P è un poliedro, allora esistono:
- un insieme finito $\{v^1, \ldots, v^m\}$ di punti di P
- un insieme finito $\{d^1, \ldots, d^q\}$ di direzioni di recessione di P tali che

$$P = \text{conv}\{v^1, \ldots, v^m\} + \underbrace{\text{cono}\{d^1, \ldots, d^q\}}_{\text{rec}(P)}.$$

Teorema di decomposizione dei poliedri

Teorema

Se P è un poliedro, allora esistono:
- un insieme finito $\{v^1, \ldots, v^m\}$ di punti di P
- un insieme finito $\{d^1, \ldots, d^q\}$ di direzioni di recessione di P tali che

$$P = \text{conv}\{v^1, \ldots, v^m\} + \underbrace{\text{cono}\{d^1, \ldots, d^q\}}_{\text{rec}(P)}.$$

Corollario 1. Se $\text{lineal}(P) = \{0\}$, allora $v^1, \ldots, v^m$ sono i vertici di P.

Corollario 2. Se P è un poliedro limitato, allora $P = \text{conv}(V)$, dove V è l'insieme dei suoi vertici.

Teorema di decomposizione dei poliedri

Teorema

Se P è un poliedro, allora esistono:
- un insieme finito $\{v^1, \ldots, v^m\}$ di punti di P
- un insieme finito $\{d^1, \ldots, d^q\}$ di direzioni di recessione di P tali che

$$P = \text{conv}\{v^1, \ldots, v^m\} + \underbrace{\text{cono}\{d^1, \ldots, d^q\}}_{\text{rec}(P)}.$$

Corollario 1. Se $\text{lineal}(P) = \{0\}$, allora $v^1, \ldots, v^m$ sono i vertici di P.

Corollario 2. Se P è un poliedro limitato, allora $P = \text{conv}(V)$, dove V è l'insieme dei suoi vertici.

Esercizio. Scrivere la decomposizione dei seguenti poliedri:
$P_1 = \{x \in \mathbb{R}^2 : \ 1 \leq x_1 \leq 4, \quad 1 \leq x_2 \leq 3\}$
$P_2 = \{x \in \mathbb{R}^2 : \ x_1 \geq 1, \quad x_2 \geq 1, \quad x_1 + x_2 \geq 3\}$
$P_3 = \{x \in \mathbb{R}^2 : \ 1 \leq x_2 \leq 2\}$

Direzioni di crescita e di decrescita

Definizione

Consideriamo un problema di PL in forma canonica

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases} \qquad (\mathcal{P})$$

Un vettore d è detto direzione di crescita per la funzione obiettivo di $(\mathcal{P})$ se $c^T d > 0$.
[la funzione obiettivo cresce lungo le semirette di direzione d:
$c^T(x + \alpha d) = c^T x + \alpha\, c^T d \to +\infty$ per $\alpha \to +\infty$]

Un vettore d è detto direzione di decrescita per la funzione obiettivo di $(\mathcal{P})$ se $c^T d < 0$.

Direzioni di crescita e di decrescita

Definizione

Consideriamo un problema di PL in forma canonica

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases} \qquad (\mathcal{P})$$

Un vettore d è detto direzione di crescita per la funzione obiettivo di $(\mathcal{P})$ se $c^T d > 0$.
[la funzione obiettivo cresce lungo le semirette di direzione d:
$c^T(x + \alpha d) = c^T x + \alpha\, c^T d \to +\infty$ per $\alpha \to +\infty$]

Un vettore d è detto direzione di decrescita per la funzione obiettivo di $(\mathcal{P})$ se $c^T d < 0$.

Esempio. Dato il problema

$$\begin{cases} \max \ 2x_1 - 3x_2 \\ x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

il vettore $d = (2, 1)$ è una direzione di crescita perché $(2, -3)^T(2, 1) = 1 > 0$,
mentre $d = (1, 1)$ è una direzione di decrescita perché $(2, -3)^T(1, 1) = -1 < 0$.

Teorema fondamentale della PL

Consideriamo un problema di PL in forma canonica

$$\begin{cases} \max \ c^T x \\ x \in P = \{x \in \mathbb{R}^n : \ Ax \leq b\} \end{cases} \qquad (\mathcal{P})$$

Teorema fondamentale della PL

Supponiamo che la regione ammissibile P sia decomposta nel modo seguente: $P = \text{conv}\{v^1, \ldots, v^m\} + \text{cono}\{d^1, \ldots, d^q\}$.

- Il valore ottimo di $(\mathcal{P})$ è finito se e solo se $c^T d^j \leq 0$ per ogni $j = 1, \ldots, q$, cioè nessuna direzione di recessione di P è una direzione di crescita.
- Se il valore ottimo di $(\mathcal{P})$ è finito, allora esiste un indice $i \in \{1, \ldots, m\}$ tale che v^i è una soluzione ottima di $(\mathcal{P})$.

Corollario. Se il poliedro P è limitato, allora un suo vertice è una soluzione ottima di $(\mathcal{P})$.

Teorema fondamentale della PL

Esempio. Consideriamo il problema

$$\begin{cases} \max\ 2x_1 - 3x_2 \\ x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

Sappiamo che $P = \text{conv}\{(1,2),(2,1)\} + \text{cono}\{(1,0),(0,1)\}$.
Il valore ottimo è $+\infty$ poiché la direzione di recessione $d = (1,0)$ è una direzione di crescita, cioè $(2,-3)^T(1,0) = 2 > 0$.

Esempio. Consideriamo il problema

$$\begin{cases} \max\ -2x_1 - 3x_2 \\ x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

Il valore ottimo è finito perché nessuna delle due direzioni di recessione $(1,0)$ e $(0,1)$ è di crescita, cioè $(-2,-3)^T(1,0) = -2$ e $(-2,-3)^T(0,1) = -3$.
La soluzione ottima è il vertice $(2,1)$, infatti $(-2,-3)^T(1,2) = -8$ e $(-2,-3)^T(2,1) = -7$.

Quante soluzioni ottime ha un problema di PL?

Una, nessuna o ... infinite

Corollario. Abbiamo tre casi possibili:

1. Se esiste una direzione di recessione di P che è anche una direzione di crescita, allora il valore ottimo è $+\infty$ e non esiste nessuna soluzione ottima.
- Se nessuna direzione di recessione di P è una direzione di crescita, allora il valore ottimo v è finito ed esiste almeno una soluzione ottima. Abbiamo due sotto-casi:
  2. la soluzione ottima è unica.
  3. se esistono due soluzioni ottime x e x' diverse, allora anche $\alpha x + (1-\alpha)x'$ è ottima per ogni $\alpha \in (0,1)$. Infatti, $\alpha x + (1-\alpha)x'$ è ammissibile e
  $$c^T[\alpha x + (1-\alpha)x'] = \alpha c^T x + (1-\alpha)c^T x' = \alpha v + (1-\alpha)v = v$$
  Quindi esistono infinite soluzioni ottime.

Esempio. Nel problema

$$\begin{cases} \max \ -x_1 - x_2 \\ x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

le soluzioni ottime sono tutti i punti del segmento di estremi $(1,2)$ e $(2,1)$.

Problema duale

Consideriamo un problema di PL forma canonica

$$\begin{cases} \max \ c^T x \\ x \in P = \{x \in \mathbb{R}^n : \ Ax \leq b\} \end{cases} \qquad (\mathcal{P})$$

che d'ora in poi sarà chiamato problema primale.

Definizione

Il problema di PL definito come

$$\begin{cases} \min \ y^T b \\ y \in D = \{y \in \mathbb{R}^m : \ y^T A = c^T, \quad y \geq 0\} \end{cases} \qquad (\mathcal{D})$$

è chiamato problema duale di $(\mathcal{P})$.

	Primale	Duale
Obiettivo	max	min
Variabili	n	m
Vincoli	m	n

Problema duale

Esempio. Il problema duale di

$$\begin{cases} \max\ 4x_1 + 5x_2 \\ x_1 \leq 1 \\ x_2 \leq 2 \\ x_1 + x_2 \leq 3 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \; b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \; c = \begin{pmatrix} 4 \\ 5 \end{pmatrix} \qquad (\mathcal{P})$$

è il problema

$$\begin{cases} \min\ y_1 + 2y_2 + 3y_3 \\ y_1 + y_3 = 4 \\ y_2 + y_3 = 5 \\ y \geq 0 \end{cases} \qquad (\mathcal{D})$$

Proprietà del problema duale

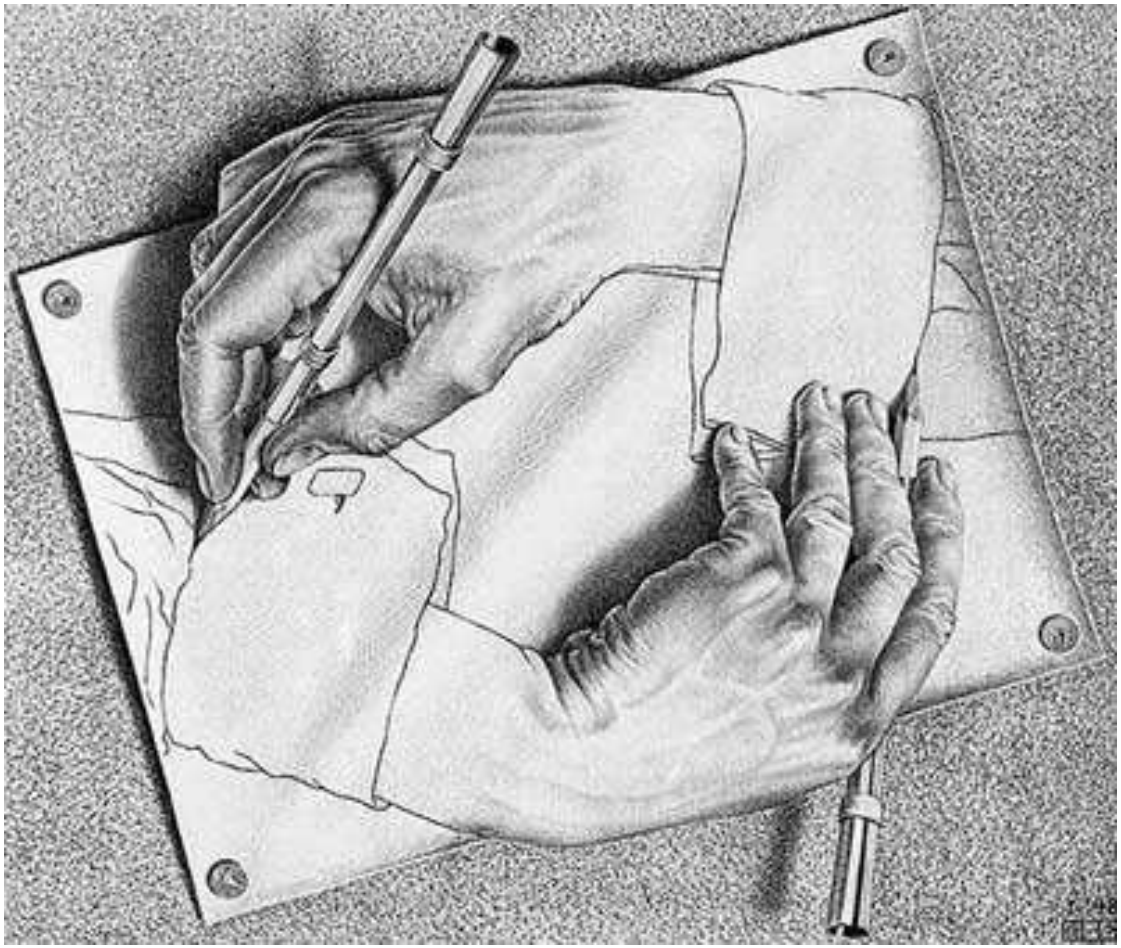
Perché è chiamato duale?

Proprietà del problema duale

Perché è chiamato duale?

Teorema

Il duale di $(\mathcal{D})$ è equivalente al problema $(\mathcal{P})$.



M.C. Escher, Drawing Hands, 1948.

Proprietà del problema duale

Teorema (Lemma di Farkas)

I sistemi

$$\begin{cases} c^T x > 0 \\ Ax \leq 0 \end{cases} \quad \text{e} \quad \begin{cases} y^T A = c^T \\ y \geq 0 \end{cases}$$

sono in alternativa, cioè uno ammette soluzioni se e solo se l'altro non ha soluzioni.

In termini di PL, il Lemma di Farkas equivale a dire che nel problema primale esiste una direzione di recessione che è anche di crescita se e solo se la regione ammissibile del problema duale è vuota.

Proprietà del problema duale

Esempio. Consideriamo il problema primale

$$\begin{cases} \max\ 2x_1 - 3x_2 \\ -x_1 \leq -1 \\ -x_2 \leq -1 \\ -x_1 - x_2 \leq -3 \end{cases}$$

Sappiamo che esiste una direzione di recessione che è di crescita. Verifichiamo ora che il suo problema duale ha la regione ammissibile vuota. Il duale è

$$\begin{cases} \min\ -y_1 - y_2 - 3y_3 \\ -y_1 - y_3 = 2 \\ -y_2 - y_3 = -3 \\ y \geq 0 \end{cases}$$

Il primo vincolo di uguaglianza implica che $y_3 = -y_1 - 2 \leq -2$ che non è compatibile con la condizione $y_3 \geq 0$, quindi non esistono soluzioni ammissibili del problema duale.

Proprietà del problema duale

Esempio. Consideriamo il problema primale

$$\begin{cases} \max\ -2x_1 - 3x_2 \\ -x_1 \leq -1 \\ -x_2 \leq -1 \\ -x_1 - x_2 \leq -3 \end{cases}$$

Sappiamo che non esistono direzioni di recessione che sono di crescita. Verifichiamo ora che il suo problema duale ha la regione ammissibile non vuota. Il duale è

$$\begin{cases} \min\ -y_1 - y_2 - 3y_3 \\ -y_1 - y_3 = -2 \\ -y_2 - y_3 = -3 \\ y \geq 0 \end{cases}$$

Dai due vincoli di uguaglianza si ottiene che $y_2 = y_1 + 1$ e $y_3 = 2 - y_1$. Esistono infinite soluzioni ammissibili del duale, ad esempio $y = (0, 1, 2)$.

Proprietà del problema duale

Consideriamo un problema primale

$$\begin{cases} \max \ c^T x \\ x \in P = \{x \in \mathbb{R}^n : \ Ax \leq b\} \end{cases} \qquad (\mathcal{P})$$

in cui la regione ammissibile $P \neq \emptyset$.

Teorema di dualità debole

Se $x \in P$ e $y \in D$, allora $c^T x \leq y^T b$. Quindi $v(\mathcal{P}) \leq v(\mathcal{D})$.

Proprietà del problema duale

Consideriamo un problema primale

$$\begin{cases} \max \ c^T x \\ x \in P = \{x \in \mathbb{R}^n : \ Ax \leq b\} \end{cases} \qquad (\mathcal{P})$$

in cui la regione ammissibile $P \neq \emptyset$.

Teorema di dualità debole

Se $x \in P$ e $y \in D$, allora $c^T x \leq y^T b$. Quindi $v(\mathcal{P}) \leq v(\mathcal{D})$.

Teorema di dualità forte

- Se il poliedro duale $D = \emptyset$, allora il valore ottimo del primale $v(\mathcal{P}) = +\infty$.
- Se il poliedro duale $D \neq \emptyset$, allora $v(\mathcal{P})$ è finito e $v(\mathcal{P}) = v(\mathcal{D})$.

Proprietà del duale

Esempio. Consideriamo il problema

$$\begin{cases} \min \ y_1 + y_2 + y_3 + y_4 \\ y_1 + y_2 - y_3 - y_4 = 2 \\ y_1 - y_2 + y_3 - y_4 = 1 \\ y \geq 0 \end{cases} \qquad (*)$$

Qual è il suo valore ottimo?

Proprietà del duale

Esempio. Consideriamo il problema

$$\begin{cases} \min \ y_1 + y_2 + y_3 + y_4 \\ y_1 + y_2 - y_3 - y_4 = 2 \\ y_1 - y_2 + y_3 - y_4 = 1 \\ y \geq 0 \end{cases} \qquad (*)$$

Qual è il suo valore ottimo? Questo problema è il duale di

$$\begin{cases} \max \ 2x_1 + x_2 \\ x_1 + x_2 \leq 1 \\ x_1 - x_2 \leq 1 \\ -x_1 + x_2 \leq 1 \\ -x_1 - x_2 \leq 1 \end{cases}$$

che è facile da risolvere graficamente: la soluzione ottima è $(1, 0)$ ed il valore ottimo è 2.
Il teorema di dualità forte garantisce che anche il valore ottimo di (*) è 2.

Condizioni di ottimalità

Come riconoscere una soluzione ottima?

Condizioni di ottimalità

Come riconoscere una soluzione ottima?

Teorema (degli scarti complementari)

Supponiamo che $\bar{x}$ sia una soluzione ammissibile del primale $(\mathcal{P})$.
Allora $\bar{x}$ è ottima se e solo se esiste una soluzione $\bar{y}$ del sistema

$$\begin{cases} \bar{y}^T A = c^T & \text{(ammissibilità} \\ \bar{y} \geq 0 & \text{duale)} \\ \bar{y}^T(b - A\bar{x}) = 0 & (\bar{x} \text{ e } \bar{y} \text{ sono in scarti complementari)} \end{cases}$$

Qualunque soluzione $\bar{y}$ di questo sistema è una soluzione ottima del duale $(\mathcal{D})$.

Il sistema sopra è equivalente al seguente sistema:

$$\begin{cases} \bar{y}^T A = c^T \\ \bar{y} \geq 0 \\ \bar{y}_i(b_i - A_i\bar{x}) = 0 \quad \forall\ i = 1, \ldots, m \end{cases}$$

Scarti complementari

Esempio. Dire se $\bar{x} = (1, 1)$ è ottima per il problema

$$\begin{cases} \max\ 3x_1 + 4x_2 \\ 2x_1 + x_2 \leq 3 \\ x_1 + 2x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases}$$

$\bar{x}$ è ottima se e solo se esiste una soluzione del sistema

$$\begin{cases} 2y_1 + y_2 - y_3 = 3 \\ y_1 + 2y_2 - y_4 = 4 \\ y \geq 0 \\ y^T(0, 0, 1, 1) = 0 \end{cases}$$

che equivale a

$$\begin{cases} y_3 = y_4 = 0 \\ 2y_1 + y_2 = 3 \\ y_1 + 2y_2 = 4 \\ y_1, y_2 \geq 0 \end{cases}$$

Poiché $\bar{y} = (2/3, 5/3, 0, 0)$ è una soluzione del sistema, $\bar{x}$ è ottima.

Scarti complementari

Esempio. Dire se $\bar{x} = (0, 0)$ è ottima per il problema

$$\begin{cases} \max\ 3x_1 + 4x_2 \\ 2x_1 + x_2 \leq 3 \\ x_1 + 2x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases}$$

$\bar{x}$ è ottima se e solo se esiste una soluzione del sistema

$$\begin{cases} 2y_1 + y_2 - y_3 = 3 \\ y_1 + 2y_2 - y_4 = 4 \\ y \geq 0 \\ y^T(3, 3, 0, 0) = 0 \end{cases}$$

che equivale a

$$\begin{cases} y_1 = y_2 = 0 \\ y_3 = -3 \\ y_4 = -4 \\ y_3, y_4 \geq 0 \end{cases}$$

che è impossibile, quindi $\bar{x}$ non è ottima.

Scarti complementari

Esercizio. Consideriamo il problema

$$\begin{cases} \max \ \alpha x_1 + 4x_2 \\ 2x_1 + x_2 \leq 3 \\ x_1 + 2x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases}$$

Per quali valori di α il vettore $\bar{x} = (1, 1)$ è ottimo?

Esercizio. Trovare tutte le soluzioni ottime del problema

$$\begin{cases} \max \ 4x_1 + 7x_2 \\ x_1 \leq 2 \\ -x_1 \leq -1 \\ x_2 \leq 4 \\ -x_2 \leq -3 \end{cases}$$

e del suo duale utilizzando il teorema degli scarti complementari.

Caratterizzazione algebrica dei vertici

Sappiamo che se il poliedro $P \neq \emptyset$ e limitato, allora un vertice di P è ottimo per il primale.

Ma i vertici di un poliedro sono definiti in modo geometrico $\rightarrow$ abbiamo bisogno di proprietà algebriche dei vertici.

Caratterizzazione algebrica dei vertici

Sappiamo che se il poliedro $P \neq \emptyset$ e limitato, allora un vertice di P è ottimo per il primale.

Ma i vertici di un poliedro sono definiti in modo geometrico $\rightarrow$ abbiamo bisogno di proprietà algebriche dei vertici.

Consideriamo un problema primale

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

dove A è una matrice $m \times n$ con $\text{rank}(A) = n$ (cioè $\text{lineal}(P) = \{0\}$).
[Questa ipotesi non è restrittiva perché ogni problema di PL è equivalente ad uno avente le variabili $x \geq 0$, ponendo eventualmente $x = x' - x''$ con $x', x'' \geq 0$.]

Caratterizzazione algebrica dei vertici

Definizione

Una base è un insieme B di n indici di riga tali che la sottomatrice A_B (formata dalle righe A_i con $i \in B$) sia invertibile, cioè $\det(A_B) \neq 0$. Indichiamo con N l'insieme degli indici non in base:

$$A = \begin{pmatrix} A_B \\ A_N \end{pmatrix} \qquad b = \begin{pmatrix} b_B \\ b_N \end{pmatrix}$$

Data una base B, il vettore $\bar{x} = A_B^{-1} b_B$ è chiamato soluzione di base primale.

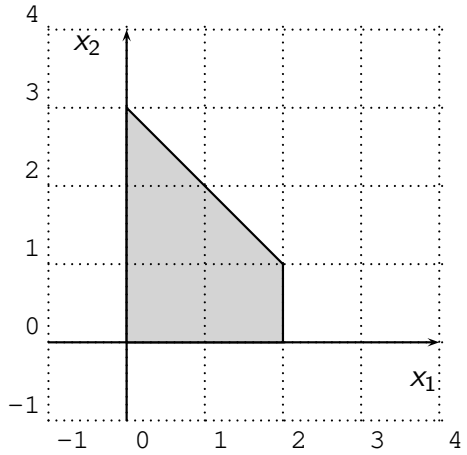
$\bar{x}$ è ammissibile se $A_N \bar{x} \leq b_N$ (tutti i vincoli non di base sono soddisfatti)
$\bar{x}$ non è ammissibile se esiste $i \in N$ tale che $A_i \bar{x} > b_i$ (almeno un vincolo non di base è violato)

$\bar{x}$ è degenere se esiste $i \in N$ tale che $A_i \bar{x} = b_i$ (almeno un vincolo non di base è attivo in $\bar{x}$)
$\bar{x}$ è non degenere se $A_i \bar{x} \neq b_i$ per ogni $i \in N$ (nessun vincolo non di base è attivo in $\bar{x}$)

Caratterizzazione algebrica dei vertici

Esempio. Consideriamo

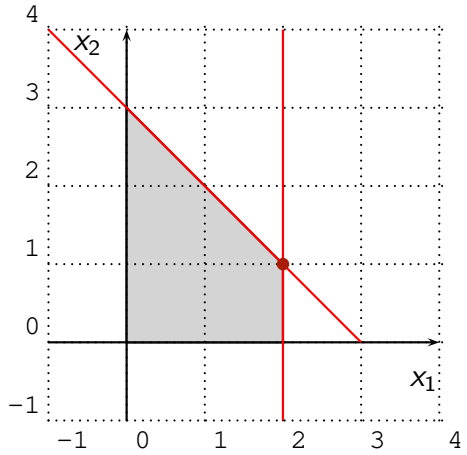
$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$



Caratterizzazione algebrica dei vertici

Esempio. Consideriamo

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$



$B = \{1, 2\}$ è una base perché $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ è invertibile: $\det(A_B) = 1$.

La relativa soluzione di base primale è $\bar{x} = A_B^{-1} b_B = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

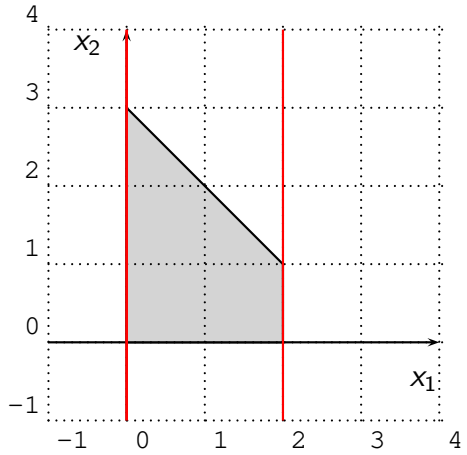
$\bar{x}$ è ammissibile perché $A_N \bar{x} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix} = b_N$

e non degenere perché $A_i \bar{x} \neq b_i$ per ogni $i \in N$.

Caratterizzazione algebrica dei vertici

Esempio. Consideriamo

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$



$B = \{1, 2\}$ è una base perché $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ è invertibile: $\det(A_B) = 1$.

La relativa soluzione di base primale è $\bar{x} = A_B^{-1} b_B = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

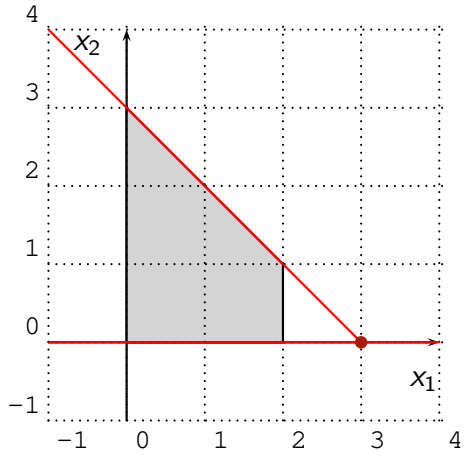
$\bar{x}$ è ammissibile perché $A_N \bar{x} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix} = b_N$
e non degenere perché $A_i \bar{x} \neq b_i$ per ogni $i \in N$.

$B = \{1, 3\}$ non è una base perché $A_B = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}$ non è invertibile: $\det(A_B) = 0$.

Caratterizzazione algebrica dei vertici

Esempio. Consideriamo

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$



$B = \{1, 2\}$ è una base perché $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ è invertibile: $\det(A_B) = 1$.

La relativa soluzione di base primale è $\bar{x} = A_B^{-1} b_B = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

$\bar{x}$ è ammissibile perché $A_N \bar{x} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix} = b_N$
e non degenere perché $A_i \bar{x} \neq b_i$ per ogni $i \in N$.

$B = \{1, 3\}$ non è una base perché $A_B = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}$ non è invertibile: $\det(A_B) = 0$.

$B = \{2, 4\}$ è una base e la relativa soluzione di base primale non è ammissibile:
$A_B = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$, $\bar{x} = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$, $A_N \bar{x} = \begin{pmatrix} 3 \\ -3 \end{pmatrix} \not\leq \begin{pmatrix} 2 \\ 0 \end{pmatrix} = b_N$ e non degenere.

Caratterizzazione algebrica dei vertici

Perché le soluzioni di base sono importanti?

Teorema

Sia $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$.
$\bar{x}$ è un vertice di P se e solo se $\bar{x}$ è una soluzione di base primale ammissibile.

Caratterizzazione algebrica dei vertici

Perché le soluzioni di base sono importanti?

Teorema

Sia $P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$.
$\bar{x}$ è un vertice di P se e solo se $\bar{x}$ è una soluzione di base primale ammissibile.

Come riconoscere un vertice ottimo?

Definizione

Data una base B, il vettore $\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix}$ dove $\bar{y}_B^T = c^T A_B^{-1}, \quad \bar{y}_N = 0$

è chiamato soluzione di base duale.

$\bar{y}$ è ammissibile se $\bar{y}_B \geq 0$
$\bar{y}$ non è ammissibile se esiste $i \in B$ tale che $\bar{y}_i < 0$

$\bar{y}$ è degenere se esiste $i \in B$ tale che $\bar{y}_i = 0$
$\bar{y}$ è non degenere se $\bar{y}_i \neq 0$ per ogni $i \in B$

Caratterizzazione algebrica dei vertici

Teorema (condizione sufficiente di ottimalità)

Sia $\bar{x}$ una soluzione di base primale ammissibile relativa alla base B.
Se la soluzione di base duale $\bar{y}$ relativa alla stessa base B è ammissibile, allora $\bar{x}$ è ottima per il problema primale (e $\bar{y}$ è ottima per il duale).

Dim. Due soluzioni di base $\bar{x}$ e $\bar{y}$ relative alla stessa base sono sempre in scarti complementari:

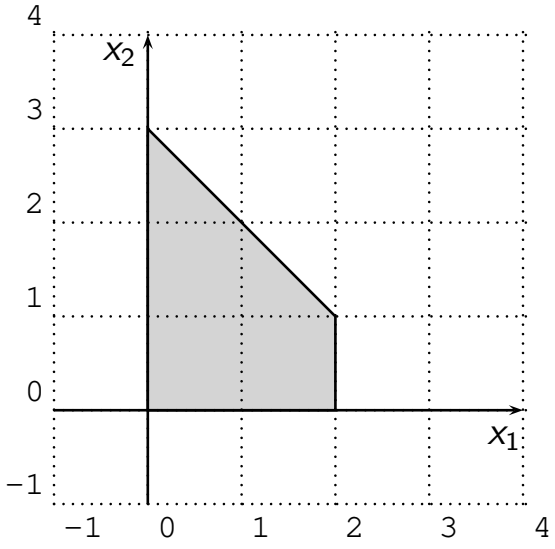
$$\bar{y}^T(b - A\bar{x}) = (\bar{y}_B^T, \bar{y}_N^T)\begin{pmatrix} b_B - A_B\bar{x} \\ b_N - A_N\bar{x} \end{pmatrix} = (\bar{y}_B^T, 0)\begin{pmatrix} 0 \\ b_N - A_N\bar{x} \end{pmatrix} = 0.$$

Se $\bar{x}$ e $\bar{y}$ sono anche ammissibili (rispettivamente per il primale e duale), allora sono anche ottime.

Caratterizzazione algebrica dei vertici

Esempio. Consideriamo il problema

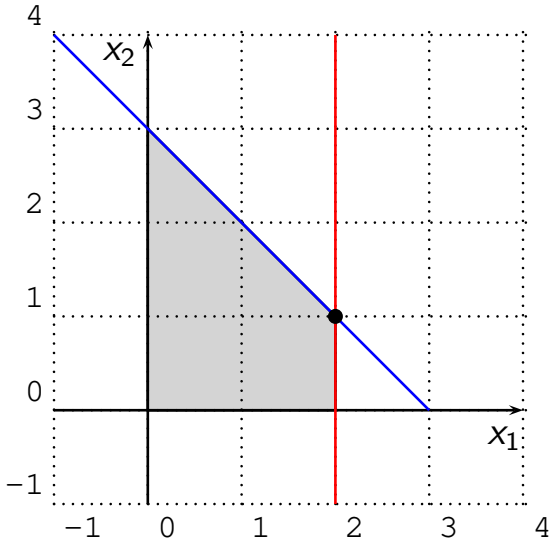
$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$



Caratterizzazione algebrica dei vertici

Esempio. Consideriamo il problema

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

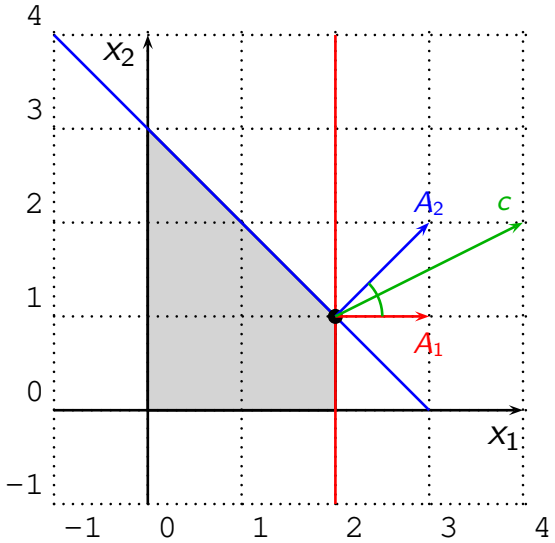


$\bar{x} = (2, 1)$ è una soluzione di base primale relativa alla base $B = \{1, 2\}$.

Caratterizzazione algebrica dei vertici

Esempio. Consideriamo il problema

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$



$\bar{x} = (2, 1)$ è una soluzione di base primale relativa alla base $B = \{1, 2\}$.
La soluzione di base duale relativa a B è

$$\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix} \qquad \text{dove} \qquad \bar{y}_B^T = c^T A_B^{-1} = (2, 1) \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = (1, 1), \qquad \bar{y}_N = 0.$$

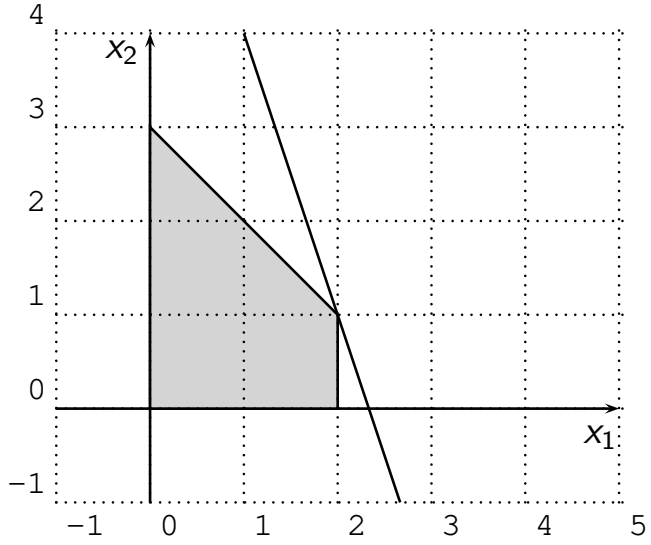
$\bar{y}$ è ammissibile perché $\bar{y}_B \geq 0$, ossia $c \in cono(A_1, A_2)$, quindi $\bar{x}$ è ottima.

Caratterizzazione algebrica dei vertici

In generale, la condizione di ottimalità basata sull'ammissibilità della soluzione di base duale è sufficiente ma non necessaria.

Esempio. Consideriamo il problema

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \\ 3x_1 + x_2 \leq 7 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \\ 3 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 7 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

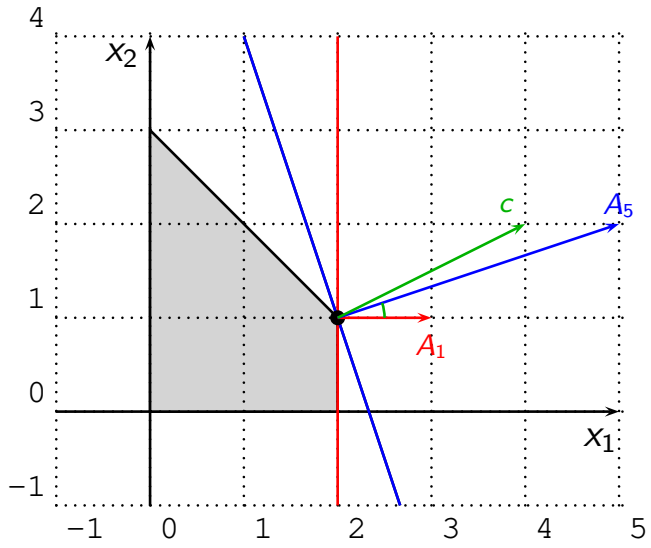


Caratterizzazione algebrica dei vertici

In generale, la condizione di ottimalità basata sull'ammissibilità della soluzione di base duale è sufficiente ma non necessaria.

Esempio. Consideriamo il problema

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \\ 3x_1 + x_2 \leq 7 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \\ 3 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 7 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$



$\bar{x} = (2, 1)$ è ottima ed è una soluzione di base primale relativa alla base $B = \{1, 5\}$.
La soluzione di base duale relativa a B è

$$\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix} \quad \text{dove} \quad \bar{y}_B^T = c^T A_B^{-1} = (2, 1) \begin{pmatrix} 1 & 0 \\ -3 & 1 \end{pmatrix} = (-1, 1), \quad \bar{y}_N = 0.$$

$\bar{y}$ non è ammissibile perché $\bar{y}_1 < 0$, ossia $c \notin cono(A_1, A_5)$, ma $\bar{x}$ è ottima.

Algoritmo del simplesso primale

Consideriamo un problema primale

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

in cui $\text{rank}(A) = n$ e quindi esistono vertici (soluzioni di base ammissibili) del poliedro primale.

L'algoritmo del simplesso primale parte da un vertice del poliedro primale.

Se il vertice del poliedro duale corrispondente alla stessa base è ammissibile, allora il vertice primale è ottimo e l'algoritmo si ferma.
Altrimenti l'algoritmo trova una direzione di spostamento che è una direzione di crescita per il primale.

Se tale direzione è di recessione per il poliedro primale, allora il valore ottimo del primale è $+\infty$ e l'algoritmo si ferma.
Altrimenti l'algoritmo trova il passo di spostamento lungo la direzione trovata e una nuova base, cambiando un solo indice rispetto alla vecchia base, in modo che la nuova soluzione di base primale rimanga ammissibile (il nuovo vertice primale è adiacente al vertice precedente).
E così via ...

Algoritmo del simplesso primale

1. Trova una base B t.c. la soluzione di base primale $\bar{x} = A_B^{-1}\, b_B$ sia ammissibile
2. Calcola la soluzione di base duale $\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix}$, dove $\bar{y}_B^T = c^T A_B^{-1}$, $\bar{y}_N = 0$
3. **Se** $\bar{y}_B \geq 0$ **allora** STOP [$\bar{x}$ è ottima per il primale, $\bar{y}$ è ottima per il duale]
   altrimenti trova l'indice uscente
   $$h = \min\{i \in B : \ \bar{y}_i < 0\} \qquad \text{(regola anticiclo di Bland)},$$
   poni $W = -A_B^{-1}$ e denota W^h la h–esima colonna di W
   (W^h è la direzione di spostamento)
4. **Se** $A_i\, W^h \leq 0$ per ogni $i \in N$
   allora STOP [valore ottimo primale $= +\infty$, problema duale vuoto]
   altrimenti calcola
   $$\vartheta := \min\left\{ \frac{b_i - A_i\, \bar{x}}{A_i\, W^h} : \ i \in N,\ A_i\, W^h > 0 \right\} \quad \text{(passo di spostamento)},$$
   trova l'indice entrante
   $$k = \min\left\{ i \in N : \ A_i\, W^h > 0,\ \frac{b_i - A_i\, \bar{x}}{A_i\, W^h} = \vartheta \right\} \quad \text{(regola anticiclo di Bland)},$$
   aggiorna la base $B = B \setminus \{h\} \cup \{k\}$, calcola $\bar{x} = A_B^{-1}\, b_B$ e torna al passo 2

Algoritmo del simplesso primale

$$(\mathcal{P}) \begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

Trova base B tale che $\bar{x} = A_B^{-1} b_B$ sia ammissibile

Calcola soluzione base duale: $\bar{y}_B^T = c^T A_B^{-1}, \quad \bar{y}_N = 0$

$\bar{y}_B \geq 0?$

si: $\bar{x}$ ottima di $(\mathcal{P})$; $\bar{y}$ ottima di $(\mathcal{D})$

no: Trova l'indice **uscente**: $h = \min\{i \in B : \ \bar{y}_i < 0\}$,
Poni $W = -A_B^{-1}$ (W^h è la h–esima colonna di W)

$A_N W^h \leq 0?$

si: $v(\mathcal{P}) = +\infty$; Duale vuoto

no: Trova $\vartheta = \min \left\{ \dfrac{b_i - A_i \bar{x}}{A_i W^h} : \ i \in N, \ A_i W^h > 0 \right\}$
Trova l'indice **entrante**:
$k = \min \left\{ i \in N : \ A_i W^h > 0, \ \dfrac{b_i - A_i \bar{x}}{A_i W^h} = \vartheta \right\}$
Aggiorna la base: $B = B \setminus \{h\} \cup \{k\}$
Calcola $\bar{x} = A_B^{-1} b_B$

Algoritmo del simplesso primale

Teorema

L'algoritmo del simplesso primale termina dopo un numero finito di iterazioni.

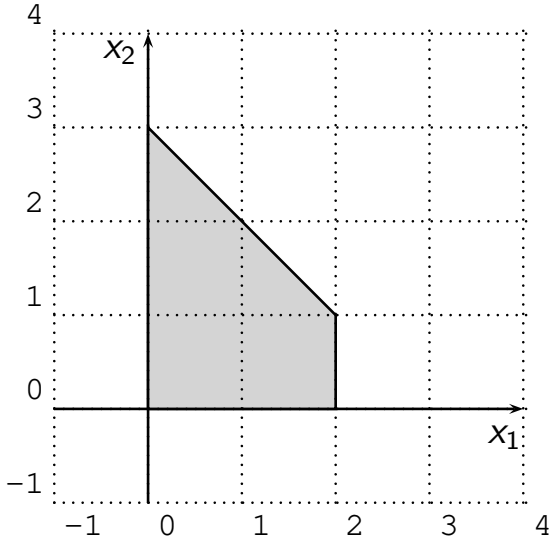
- Se il valore ottimo del problema è $+\infty$, l'algoritmo trova una direzione di recessione che è anche di crescita.
- Se il valore ottimo del problema è finito, l'algoritmo trova un vertice ottimo.

Algoritmo del simplesso primale

Esempio. Risolviamo il problema

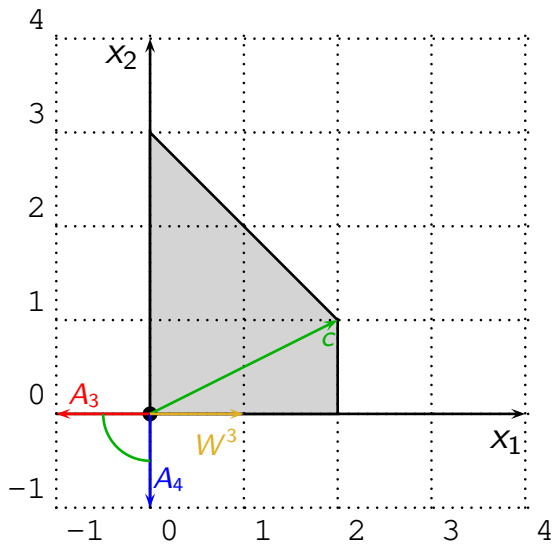
$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.



Algoritmo del simplesso primale

Esempio. Risolviamo il problema



$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \le 2 \\ x_1 + x_2 \le 3 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.

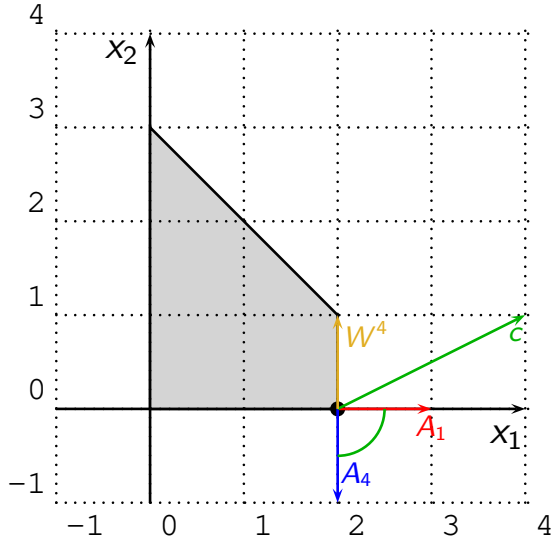
Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ è ammissibile.

$\bar{y}_B^T = (2, 1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, -1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = 1$, $A_2 W^3 = 1$,

$\vartheta = \min\{2/1, 3/1\} = 2$, $k = 1$.

Algoritmo del simplesso primale

Esempio. Risolviamo il problema



$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \le 2 \\ x_1 + x_2 \le 3 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.

Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ è ammissibile.

$\bar{y}_B^T = (2, 1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, -1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = 1$, $A_2 W^3 = 1$,

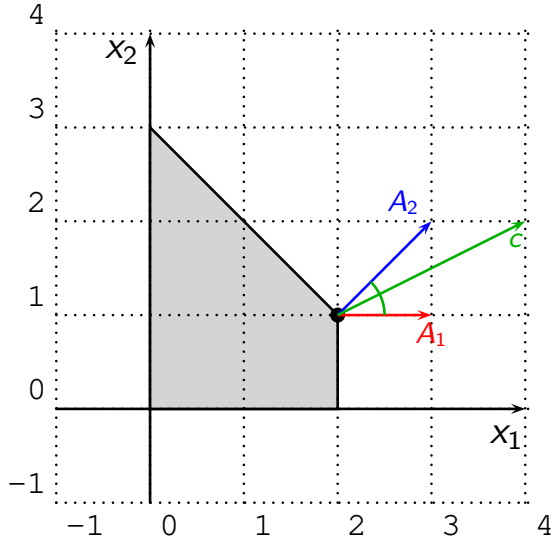
$\vartheta = \min\{2/1, 3/1\} = 2$, $k = 1$.

Iterazione 2. $B = \{1, 4\}$, $A_B = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$,

$\bar{y}_B^T = (2, 1) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = (2, -1)$, $h = 4$, $W^4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. $A_2 W^4 = 1$, $A_3 W^4 = 0$, $k = 2$.

Algoritmo del simplesso primale

Esempio. Risolviamo il problema



$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \le 2 \\ x_1 + x_2 \le 3 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases} \quad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.

Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ è ammissibile.

$\bar{y}_B^T = (2, 1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, -1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = 1$, $A_2 W^3 = 1$,

$\vartheta = \min\{2/1, 3/1\} = 2$, $k = 1$.

Iterazione 2. $B = \{1, 4\}$, $A_B = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$,

$\bar{y}_B^T = (2, 1) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = (2, -1)$, $h = 4$, $W^4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. $A_2 W^4 = 1$, $A_3 W^4 = 0$, $k = 2$.

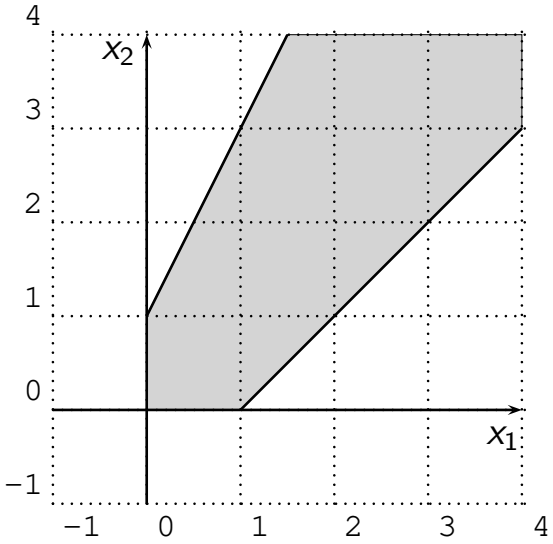
Iterazione 3. $B = \{1, 2\}$, $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, $\bar{x} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\bar{y}_B^T = (1, 1) \ge 0$ stop, $\bar{x}$ è ottimo.

Algoritmo del simplesso primale

Esempio. Risolviamo il problema

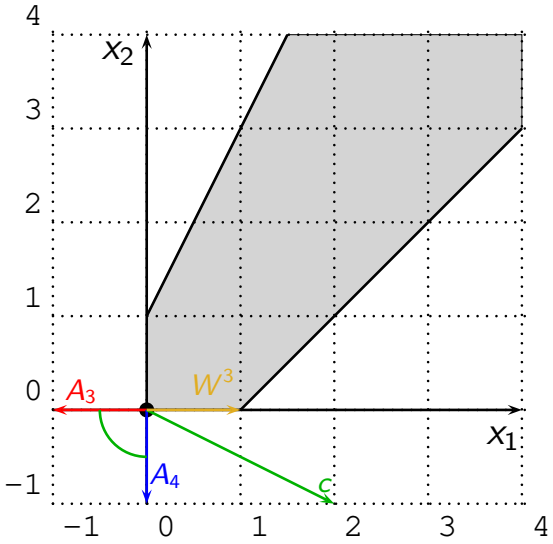
$$\begin{cases} \max\ 2x_1 - x_2 \\ -2x_1 + x_2 \leq 1 \\ x_1 - x_2 \leq 1 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \quad A = \begin{pmatrix} -2 & 1 \\ 1 & -1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.



Algoritmo del simplesso primale

Esempio. Risolviamo il problema



$$\begin{cases} \max\ 2x_1 - x_2 \\ -2x_1 + x_2 \le 1 \\ x_1 - x_2 \le 1 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases} \quad A = \begin{pmatrix} -2 & 1 \\ 1 & -1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

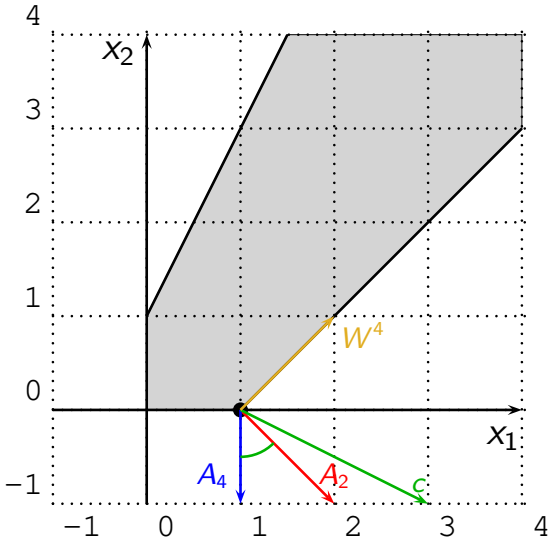
partendo dalla base $B = \{3, 4\}$.

Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ è ammissibile.

$\bar{y}_B^T = (2, -1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, 1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = -2$, $A_2 W^3 = 1$, $k = 2$.

Algoritmo del simplesso primale

Esempio. Risolviamo il problema



$$\begin{cases} \max\ 2x_1 - x_2 \\ -2x_1 + x_2 \le 1 \\ x_1 - x_2 \le 1 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases} \quad A = \begin{pmatrix} -2 & 1 \\ 1 & -1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

partendo dalla base $B = \{3, 4\}$.

Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ è ammissibile.

$\bar{y}_B^T = (2, -1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, 1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = -2$, $A_2 W^3 = 1$, $k = 2$.

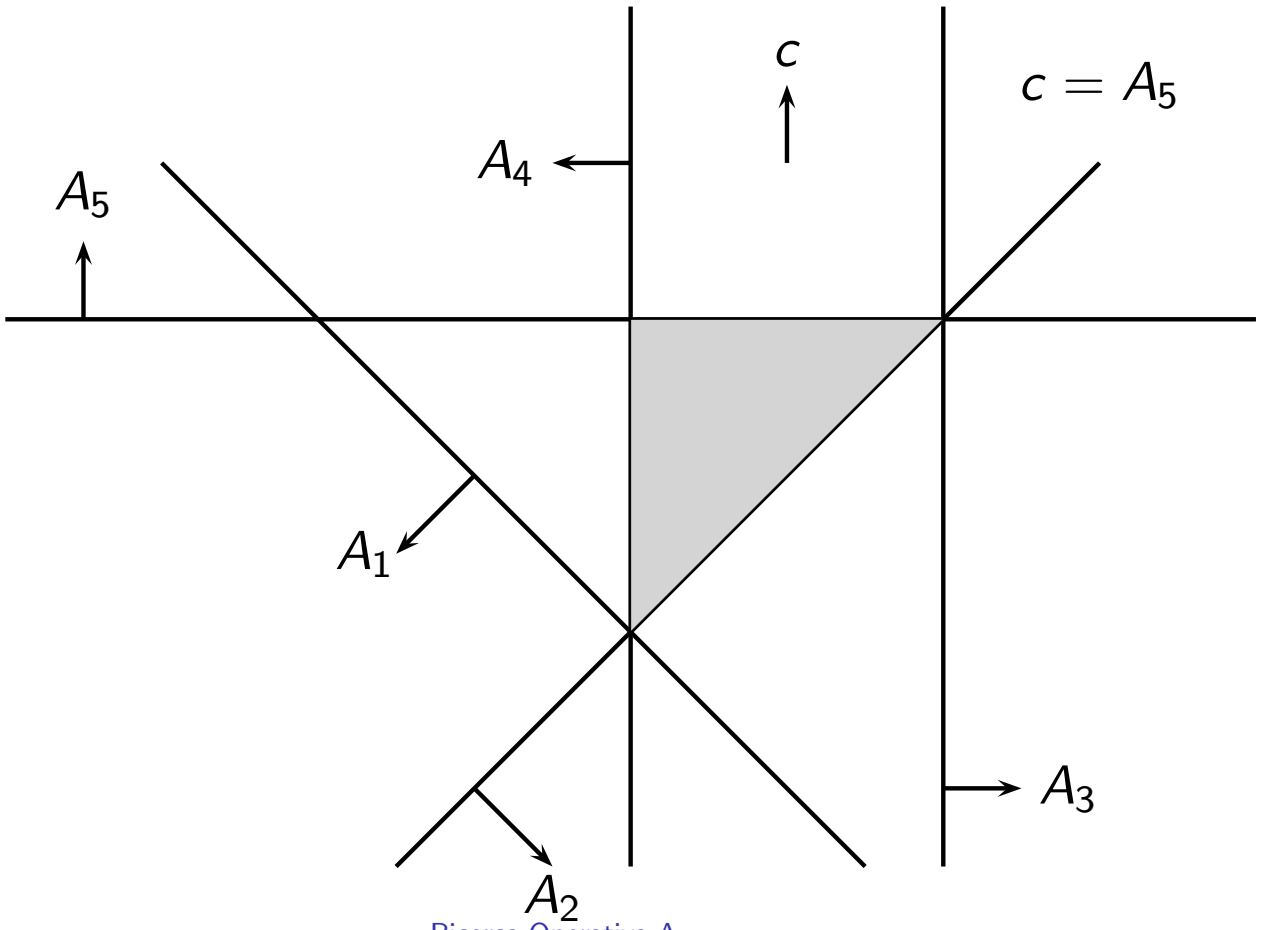
Iterazione 2. $B = \{2, 4\}$, $A_B = \begin{pmatrix} 1 & -1 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$,

$\bar{y}_B^T = (2, -1) \begin{pmatrix} 1 & -1 \\ 0 & -1 \end{pmatrix} = (2, -1)$, $h = 4$, $W^4 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. $A_1 W^4 = -1$, $A_3 W^4 = -1$ stop, il valore ottimo è $+\infty$.

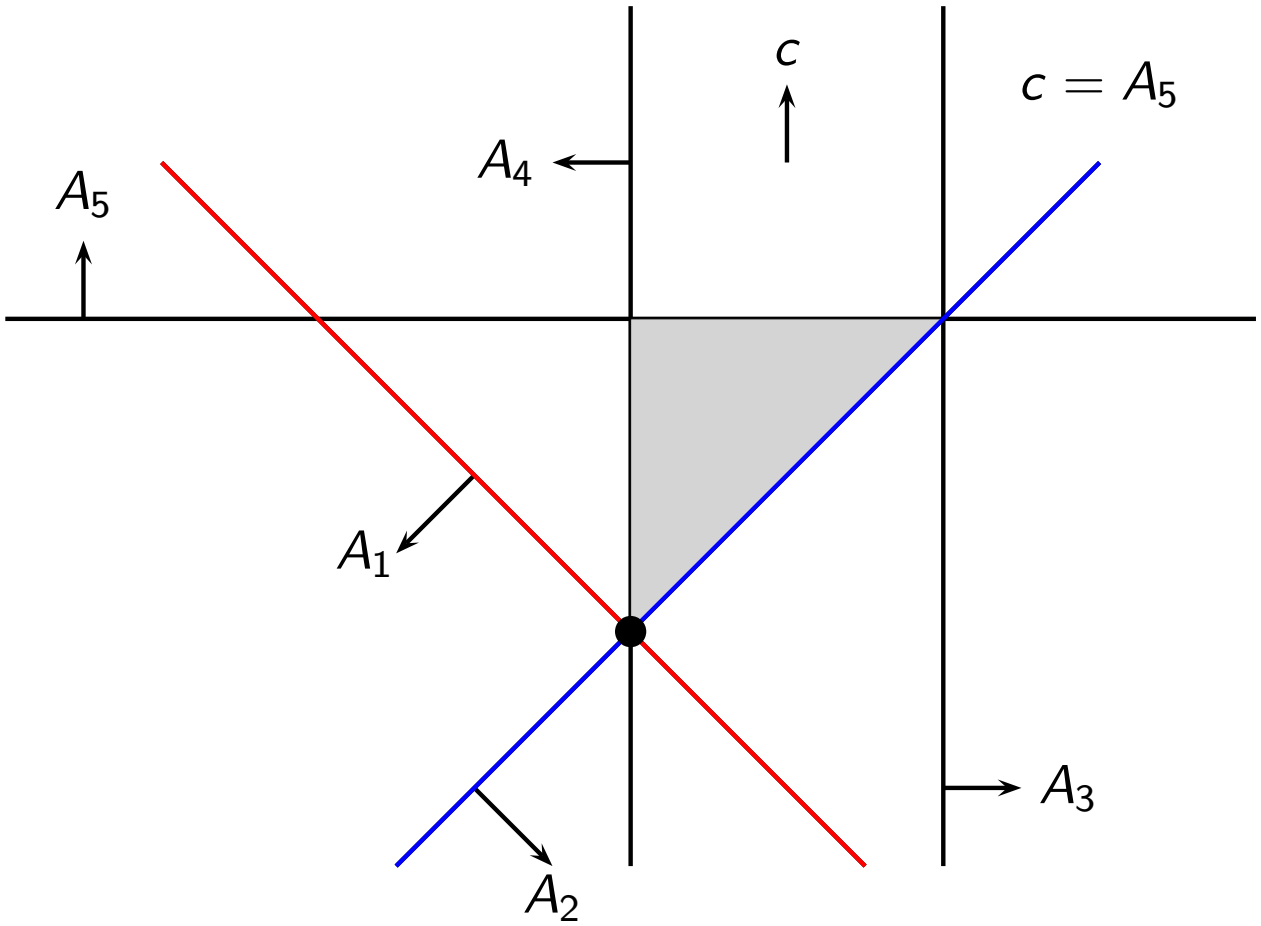
Esercizio. Verificare che il problema duale ha la regione ammissibile vuota.

Algoritmo del simplesso primale

Esercizio. Risolvere geometricamente per mezzo dell'algoritmo del simplesso primale il problema di PL in figura, partendo dalla base $B = \{1, 2\}$. Per ogni iterazione, trovare la base, la soluzione di base primale, il segno delle componenti della soluzione di base duale, l'indice uscente, la direzione di spostamento, l'indice entrante.

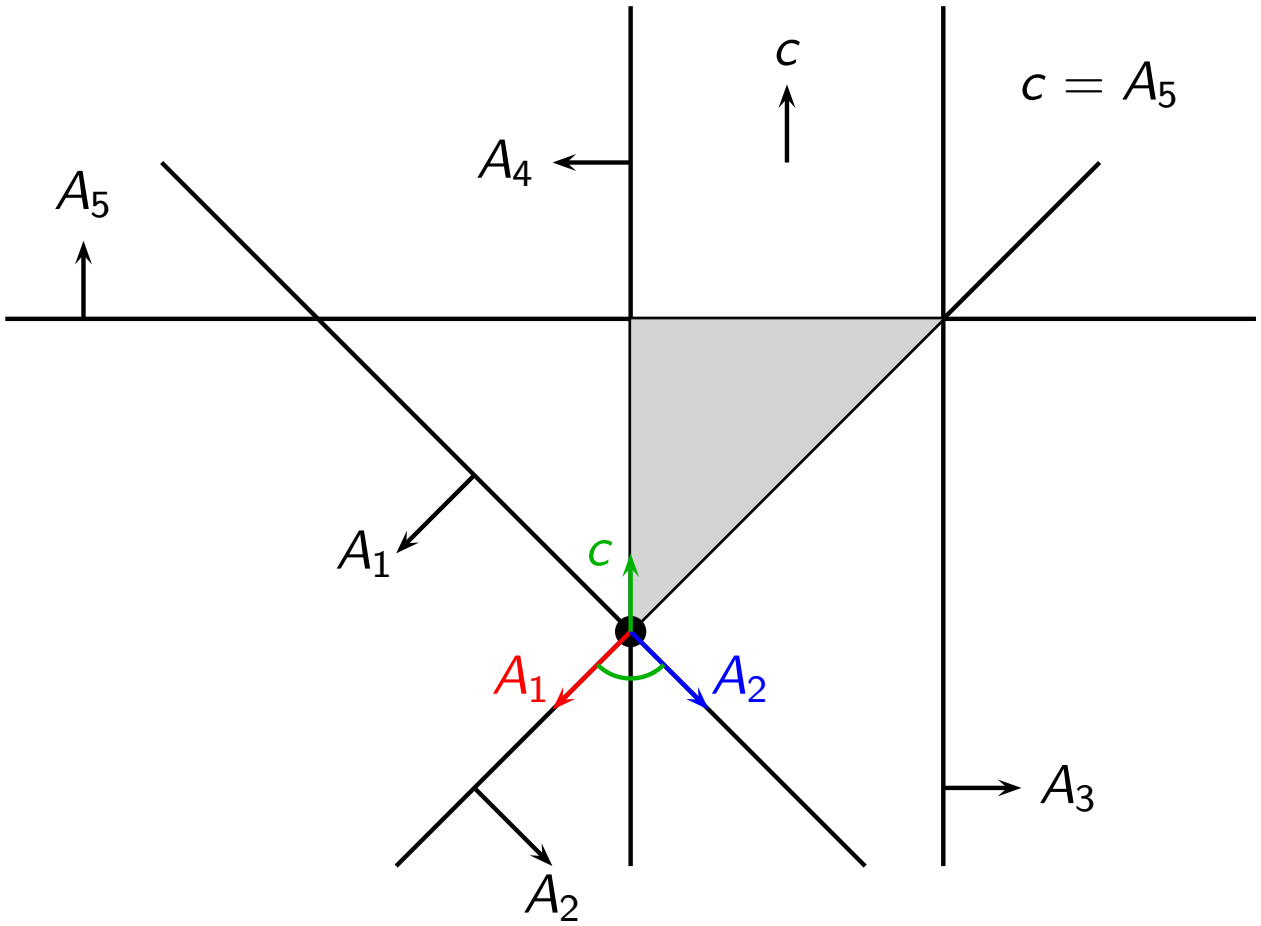


Algoritmo del simplesso primale



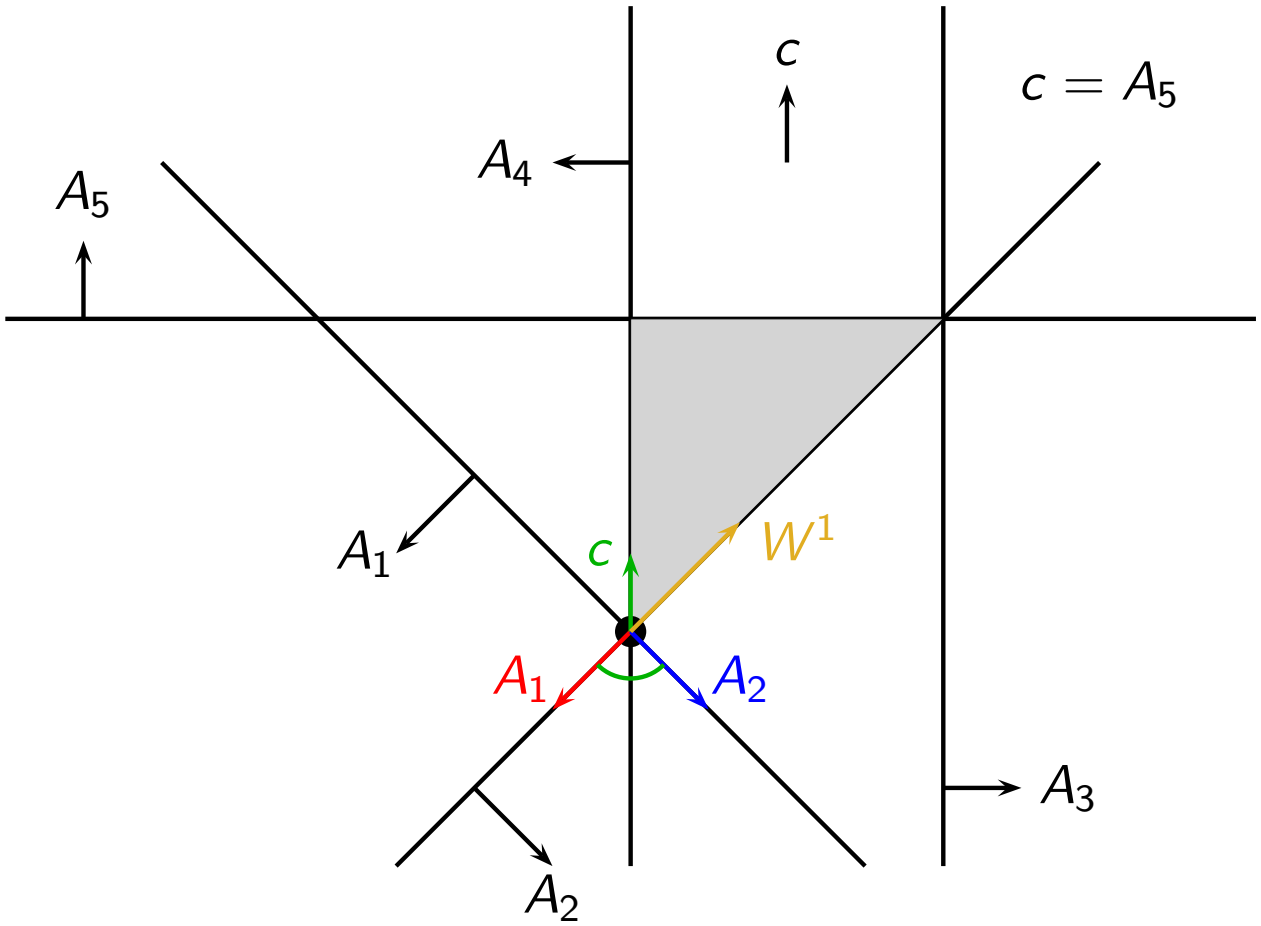
Iterazione 1. $B = \{1, 2\}$, $\bar{x}$ indicata in figura,

Algoritmo del simplesso primale



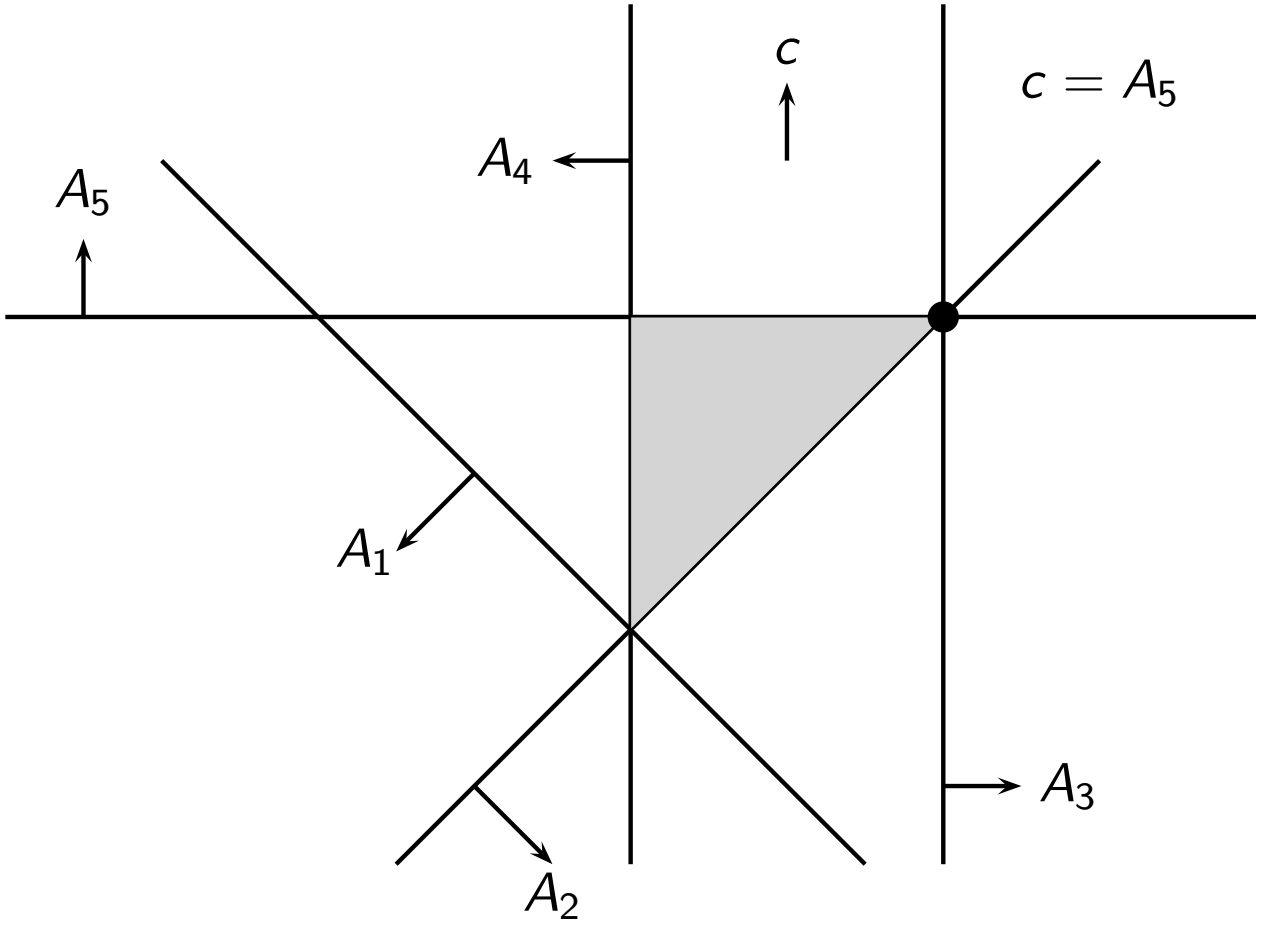
Iterazione 1. $B = \{1, 2\}$, $\bar{x}$ indicata in figura, $c \in \operatorname{int} \operatorname{cono}(-A_1, -A_2)$, quindi $\bar{y}_1 < 0$, $\bar{y}_2 < 0$.

Algoritmo del simplesso primale



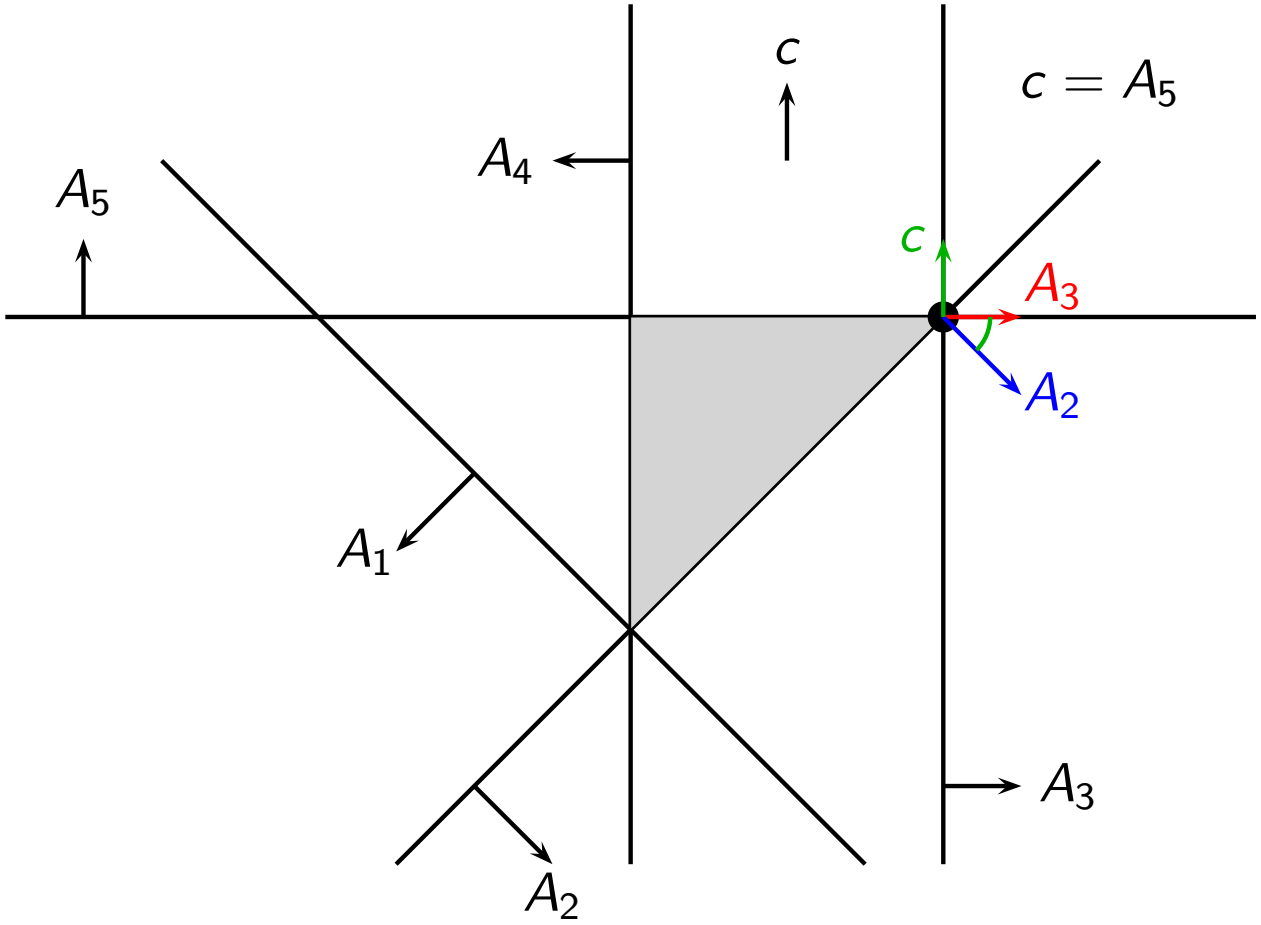
Iterazione 1. $B = \{1, 2\}$, $\bar{x}$ indicata in figura, $c \in \operatorname{int} \operatorname{cono}(-A_1, -A_2)$, quindi $\bar{y}_1 < 0$, $\bar{y}_2 < 0$. Indice uscente $h = 1$, W^1 indicata in figura, indice entrante $k = \min\{3, 5\} = 3$.

Algoritmo del simplesso primale



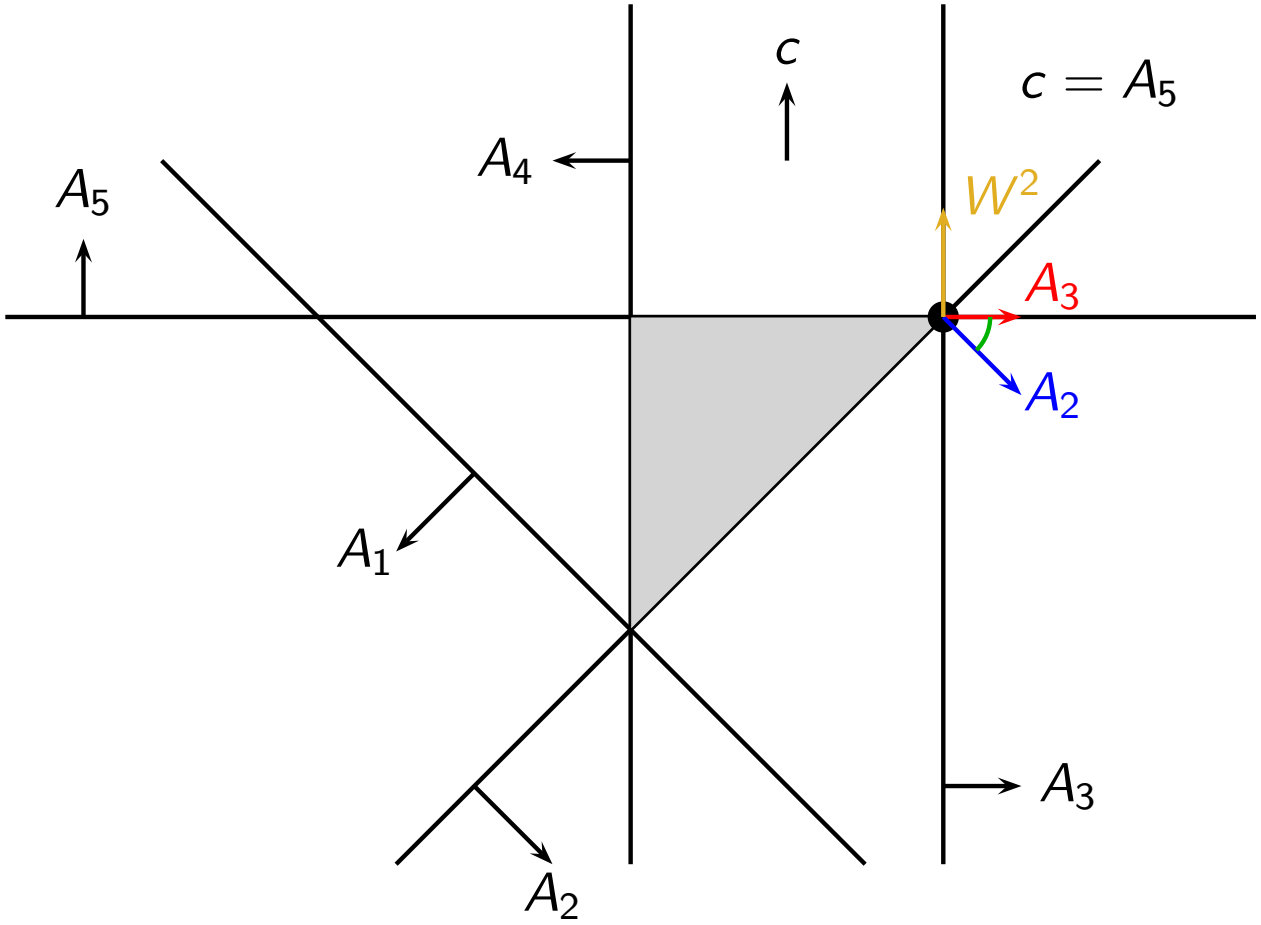
Iterazione 2. $B = \{2, 3\}$, $\bar{x}$ indicata in figura,

Algoritmo del simplesso primale



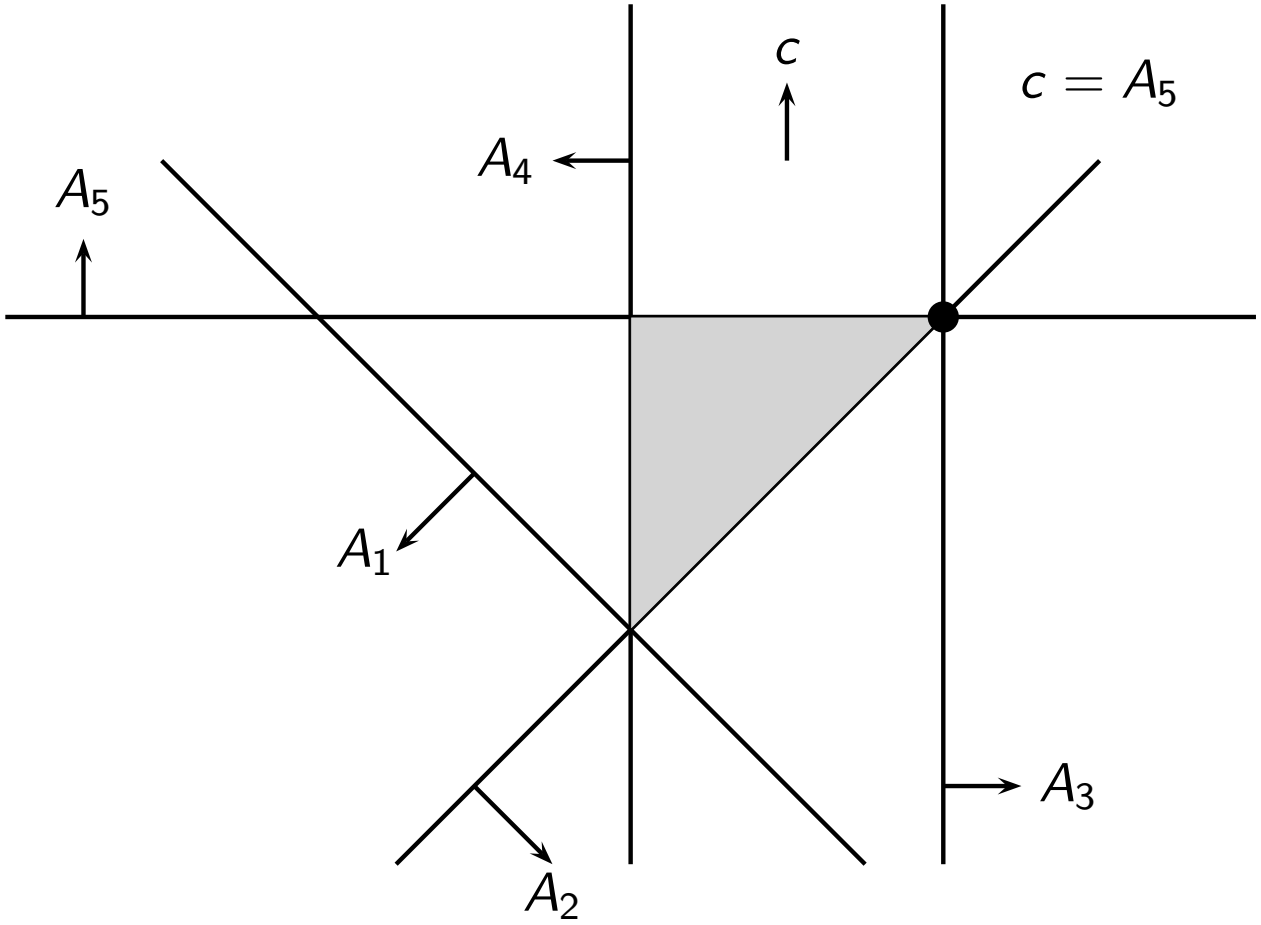
Iterazione 2. $B = \{2, 3\}$, $\bar{x}$ indicata in figura, $c \in \operatorname{int} \operatorname{cono}(-A_2, A_3)$, quindi $\bar{y}_2 < 0$, $\bar{y}_3 > 0$.

Algoritmo del simplesso primale



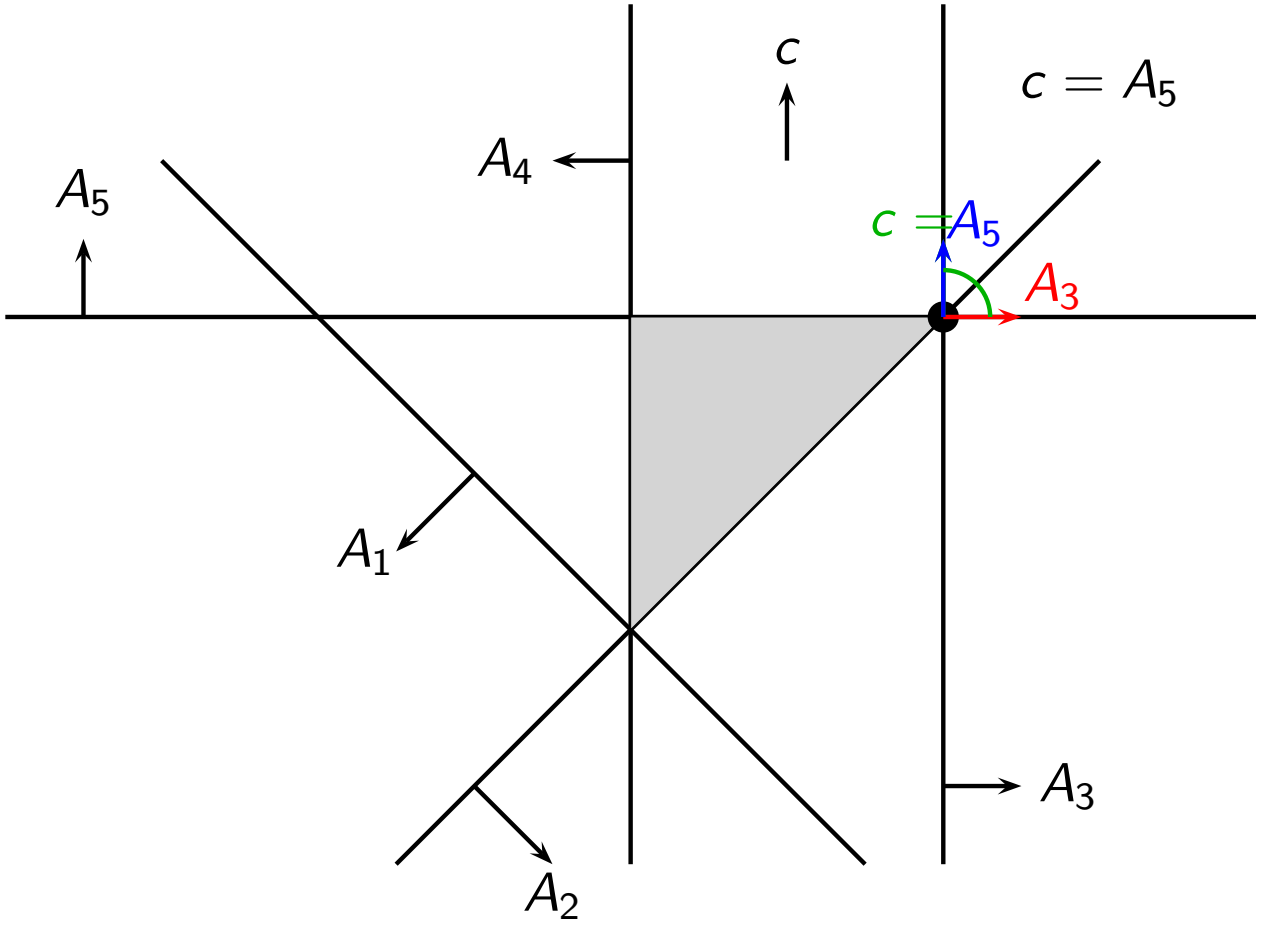
Iterazione 2. $B = \{2, 3\}$, $\bar{x}$ indicata in figura, $c \in \operatorname{int}\operatorname{cono}(-A_2, A_3)$, quindi $\bar{y}_2 < 0$, $\bar{y}_3 > 0$. Indice uscente $h = 2$, W^2 indicata in figura, indice entrante $k = 5$.

Algoritmo del simplesso primale



Iterazione 3. $B = \{3, 5\}$, $\bar{x}$ indicata in figura,

Algoritmo del simplesso primale



Iterazione 3. $B = \{3, 5\}$, $\bar{x}$ indicata in figura, $c = A_5$, quindi $\bar{y}_3 = 0$, $\bar{y}_5 = 1$, stop $\bar{x}$ è ottima.

Algoritmo del simplesso primale - vertice iniziale

Come si trova una soluzione di base ammissibile per $(\mathcal{P})$?

Supponiamo che B sia una base e che $\bar{x} = A_B^{-1} b_B$ non sia ammissibile.
Definiamo gli insiemi

$$U = \{i \in N : \ A_i \bar{x} \leq b_i\}, \qquad V = \{i \in N : \ A_i \bar{x} > b_i\},$$

e costruiamo il problema ausiliario primale:

$$\begin{cases} \max\limits_{(x,\varepsilon)} \ -\sum\limits_{i \in V} \varepsilon_i & \\ A_i x \leq b_i & \text{per } i \in B \cup U \\ A_i x - \varepsilon_i \leq b_i & \text{per } i \in V \\ -\varepsilon_i \leq 0 & \text{per } i \in V \end{cases} \qquad (\mathcal{P}_{aux})$$

Il vettore $(\bar{x}, \bar{\varepsilon})$, con

$$\bar{\varepsilon} = A_V \bar{x} - b_V \geq 0,$$

è una soluzione di base ammissibile per $(\mathcal{P}_{aux})$ relativa alla base $B \cup V$, con matrice di base uguale a

$$\left(\begin{array}{c|c} A_B & 0 \\ \hline A_V & -I \end{array} \right).$$

Algoritmo del simplesso primale - vertice iniziale

A partire da tale soluzione di base ammissibile per $(\mathcal{P}_{aux})$, applichiamo l'algoritmo del simplesso primale per risolvere il problema ausiliario.

Teorema

1. Se il valore ottimo di $(\mathcal{P}_{aux})$ è < 0, allora $(\mathcal{P})$ non ha soluzioni ammissibili.
2. Se il valore ottimo di $(\mathcal{P}_{aux})$ è $= 0$, allora a partire da una soluzione di base ottima di $(\mathcal{P}_{aux})$ si può costruire una soluzione di base ammissibile per $(\mathcal{P})$.

Algoritmo del simplesso duale

Consideriamo un problema primale

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

in cui $\text{rank}(A) = n$ e quindi esiste almeno un vertice del poliedro primale.

L'algoritmo del simplesso duale parte da un vertice del poliedro duale.

Se il vertice del poliedro primale corrispondente alla stessa base è ammissibile, allora il vertice primale è ottimo e l'algoritmo si ferma. Altrimenti l'algoritmo trova una direzione di spostamento che è una direzione di decrescita per il duale.

Se tale direzione è di recessione per il poliedro duale, allora la regione ammissibile del primale è vuota e l'algoritmo si ferma.
Altrimenti l'algoritmo trova il passo di spostamento lungo la direzione trovata e una nuova base, cambiando un solo indice rispetto alla vecchia base, in modo che la nuova soluzione di base duale rimanga ammissibile (il nuovo vertice duale è adiacente al vertice precedente).
E così via ...

Algoritmo del simplesso duale

1. Trova una base B tale che la soluzione di base duale $\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix}$, dove $\bar{y}_B^T = c^T A_B^{-1}$ e $\bar{y}_N = 0$, sia ammissibile
2. Calcola la soluzione di base primale $\bar{x} = A_B^{-1} b_B$
3. **Se** $A_N \bar{x} \leq b_N$ **allora** STOP [$\bar{x}$ è ottima per il primale, $\bar{y}$ è ottima per il duale]
   altrimenti trova l'indice entrante
   $$k = \min\{i \in N : \ A_i \bar{x} > b_i\} \qquad \text{(regola anticiclo di Bland)},$$
   poni $\eta_B := A_k A_B^{-1}$, la direzione di spostamento d è $d_i = \begin{cases} -\eta_i & \text{se } i \in B \\ 1 & \text{se } i = k \\ 0 & \text{se } i \in N \setminus \{k\} \end{cases}$
4. **Se** $\eta_B \leq 0$ **allora** STOP [il primale è vuoto, valore ottimo duale $= -\infty$]
   altrimenti
   calcola $\vartheta = \min \left\{ \dfrac{\bar{y}_i}{\eta_i} : \quad i \in B, \ \eta_i > 0 \right\}$ (passo di spostamento)
   trova l'indice uscente
   $$h = \min \left\{ i \in B : \ \eta_i > 0, \ \frac{\bar{y}_i}{\eta_i} = \vartheta \right\} \text{ (regola anticiclo di Bland)},$$
   aggiorna la base $B = B \setminus \{h\} \cup \{k\}$, calcola $\bar{y}_B^T = c^T A_B^{-1}$ e torna al passo 2

Algoritmo del simplesso duale

$$(\mathcal{P}) \begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

Trova base B tale che $\bar{y}_B^T = c^T A_B^{-1} \geq 0$

Calcola soluzione base primale: $\bar{x} = A_B^{-1} b_B$

$A_N \bar{x} \leq b_N$?

si: $\bar{x}$ ottima di $(\mathcal{P})$, $\bar{y}$ ottima di $(\mathcal{D})$

no: Trova l'indice **entrante**: $k = \min\{i \in N : A_i \bar{x} > b_i\}$

Poni $\eta_B := A_k A_B^{-1}$

$\eta_B \leq 0$?

si: Primale vuoto, $v(\mathcal{D}) = -\infty$

no: Trova il passo $\vartheta = \min \left\{ \frac{\bar{y}_i}{\eta_i} : \quad i \in B, \ \eta_i > 0 \right\}$

Trova l'indice **uscente**:

$h = \min \left\{ i \in B : \ \eta_i > 0, \ \frac{\bar{y}_i}{\eta_i} = \vartheta \right\}$

Aggiorna la base: $B = B \setminus \{h\} \cup \{k\}$

Calcola $\bar{y}_B^T = c^T A_B^{-1}$

Algoritmo del simplesso duale

Teorema

L'algoritmo del simplesso duale termina dopo un numero finito di iterazioni.

- Se la regione ammissibile del primale è vuota, l'algoritmo trova una direzione di recessione per il duale che è anche di decrescita.
- Se la regione ammissibile del primale non è vuota, l'algoritmo trova un vertice ottimo.

Algoritmo del simplesso duale

Esempio. Risolviamo il problema

$$\begin{cases} \max\ x_2 \\ x_2 \leq 4 \\ -x_1 + 2x_2 \leq 10 \\ -x_1 \leq 1 \\ -2x_1 + x_2 \leq 4 \\ -x_1 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 0 & 1 \\ -1 & 2 \\ -1 & 0 \\ -2 & 1 \\ -1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 4 \\ 10 \\ 1 \\ 4 \\ 0 \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

con l'algoritmo del simplesso duale partendo dalla base $B = \{1, 2\}$.

Iterazione 1. $A_B = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix}$, $A_B^{-1} = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix}$, $\bar{y}_B^T = (0, 1) \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix} = (1, 0)$ quindi $\bar{y}$ è ammissibile. $\bar{x} = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 10 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix}$.

$$A_N \bar{x} = \begin{pmatrix} -1 & 0 \\ -2 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} -2 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \\ 2 \end{pmatrix} \not\leq \begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix} = b_N,$$

$k = \min\{3, 4, 5\} = 3$, $\eta_B = A_3 A_B^{-1} = (-1, 0) \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix} = (-2, 1)$, $h = 2$.

Algoritmo del simplesso duale

Esempio. Risolviamo il problema

$$\begin{cases} \max\ x_2 \\ x_2 \leq 4 \\ -x_1 + 2x_2 \leq 10 \\ -x_1 \leq 1 \\ -2x_1 + x_2 \leq 4 \\ -x_1 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 0 & 1 \\ -1 & 2 \\ -1 & 0 \\ -2 & 1 \\ -1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 4 \\ 10 \\ 1 \\ 4 \\ 0 \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

con l'algoritmo del simplesso duale partendo dalla base $B = \{1, 2\}$.

Iterazione 2. $B = \{1, 3\}$, $A_B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, $A_B^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$,

$\bar{y}_B^T = (0, 1) \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = (1, 0)$, $\bar{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 4 \end{pmatrix}$.

$$A_N \bar{x} = \begin{pmatrix} -1 & 2 \\ -2 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 4 \end{pmatrix} = \begin{pmatrix} 9 \\ 6 \\ 1 \end{pmatrix} \not\leq \begin{pmatrix} 10 \\ 4 \\ 0 \end{pmatrix},$$

$k = \min\{4, 5\} = 4$, $\eta_B = A_4 A_B^{-1} = (-2, 1) \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = (1, 2)$, $\vartheta = \min\{1, 0\} = 0$,

$h = 3$.

Algoritmo del simplesso duale

Esempio. Risolviamo il problema

$$\begin{cases} \max\ x_2 \\ x_2 \leq 4 \\ -x_1 + 2x_2 \leq 10 \\ -x_1 \leq 1 \\ -2x_1 + x_2 \leq 4 \\ -x_1 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 0 & 1 \\ -1 & 2 \\ -1 & 0 \\ -2 & 1 \\ -1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 4 \\ 10 \\ 1 \\ 4 \\ 0 \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

con l'algoritmo del simplesso duale partendo dalla base $B = \{1, 2\}$.

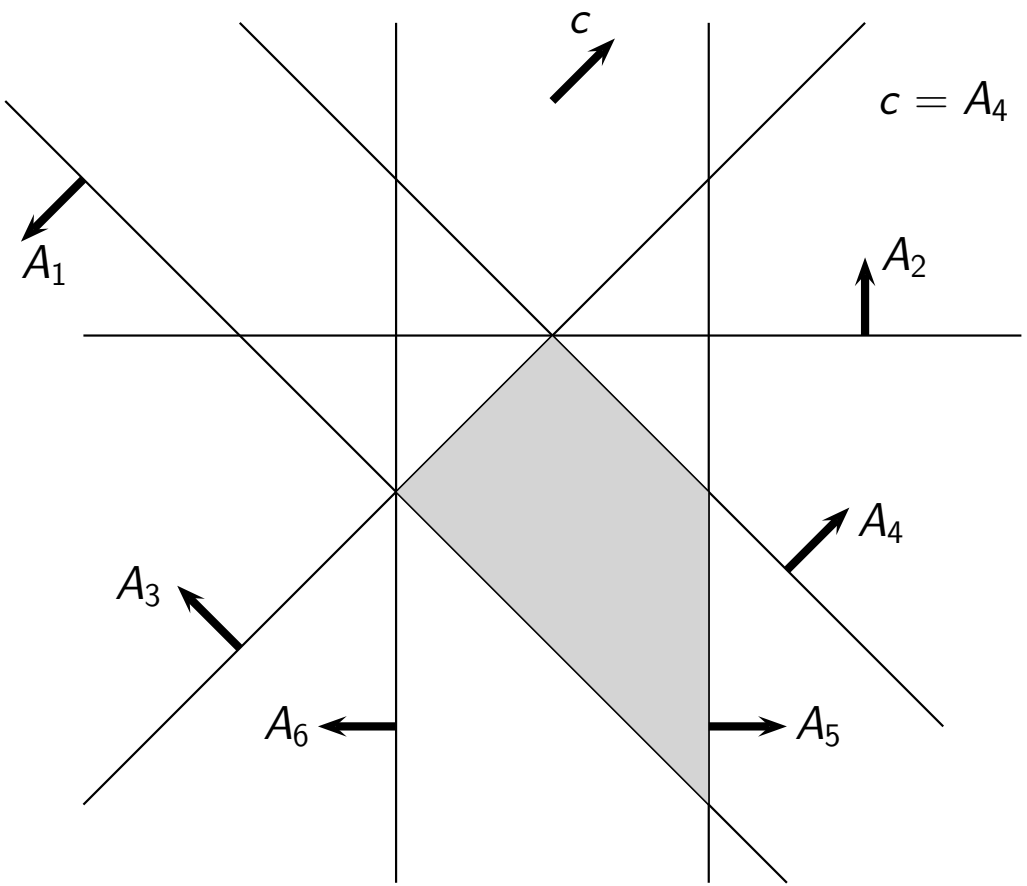
Iterazione 3. $B = \{1, 4\}$, $A_B = \begin{pmatrix} 0 & 1 \\ -2 & 1 \end{pmatrix}$, $A_B^{-1} = \begin{pmatrix} 1/2 & -1/2 \\ 1 & 0 \end{pmatrix}$,
$\bar{y}_B^T = (0, 1) \begin{pmatrix} 1/2 & -1/2 \\ 1 & 0 \end{pmatrix} = (1, 0)$, $\bar{x} = \begin{pmatrix} 1/2 & -1/2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$.

$$A_N \bar{x} = \begin{pmatrix} -1 & 2 \\ -1 & 0 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 4 \end{pmatrix} = \begin{pmatrix} 8 \\ 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 10 \\ 1 \\ 0 \end{pmatrix},$$

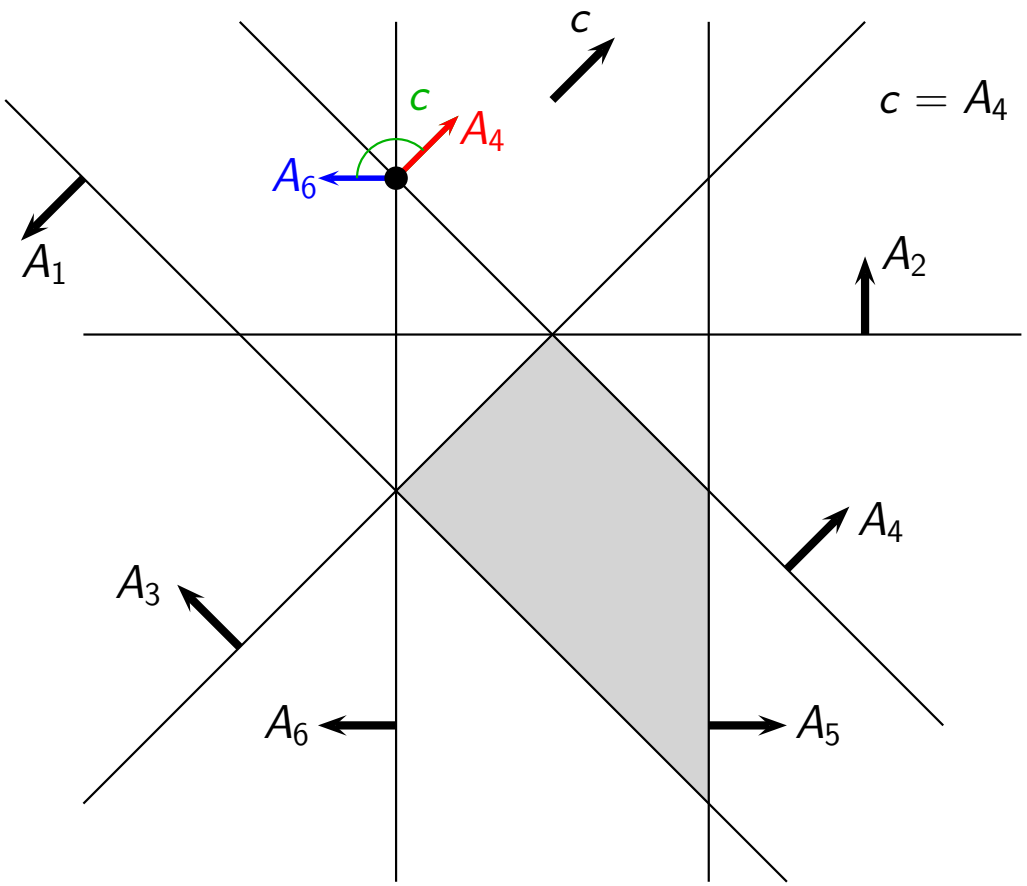
stop. $\bar{x} = (0, 4)$ è una soluzione ottima del primale e $\bar{y} = (1, 0, 0, 0, 0)$ una soluzione ottima del duale.

Algoritmo del simplesso duale

Esercizio. Risolvere geometricamente per mezzo dell'algoritmo del simplesso duale il problema di PL in figura, partendo dalla base $B = \{4, 6\}$. Per ogni iterazione, trovare la base, la soluzione di base primale, l'indice entrante, i segni delle componenti dei vettori $\bar{y}_B$ e η_B, l'indice uscente.

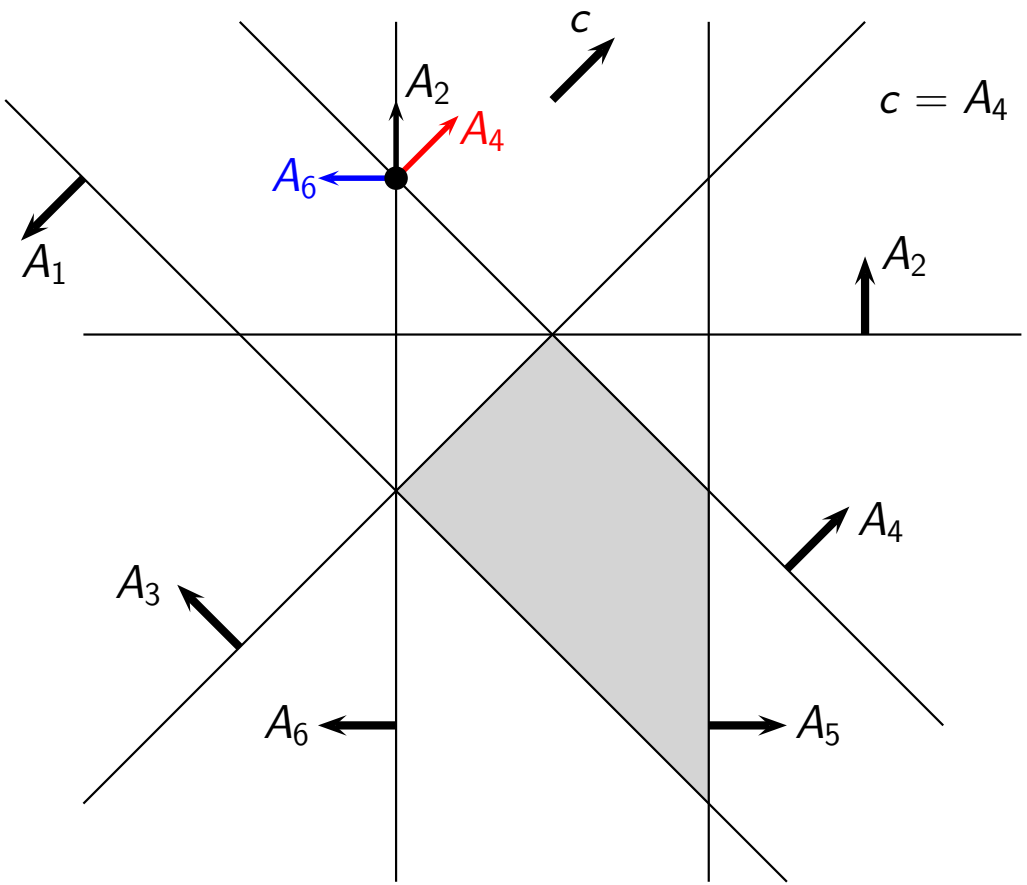


Algoritmo del simplesso duale



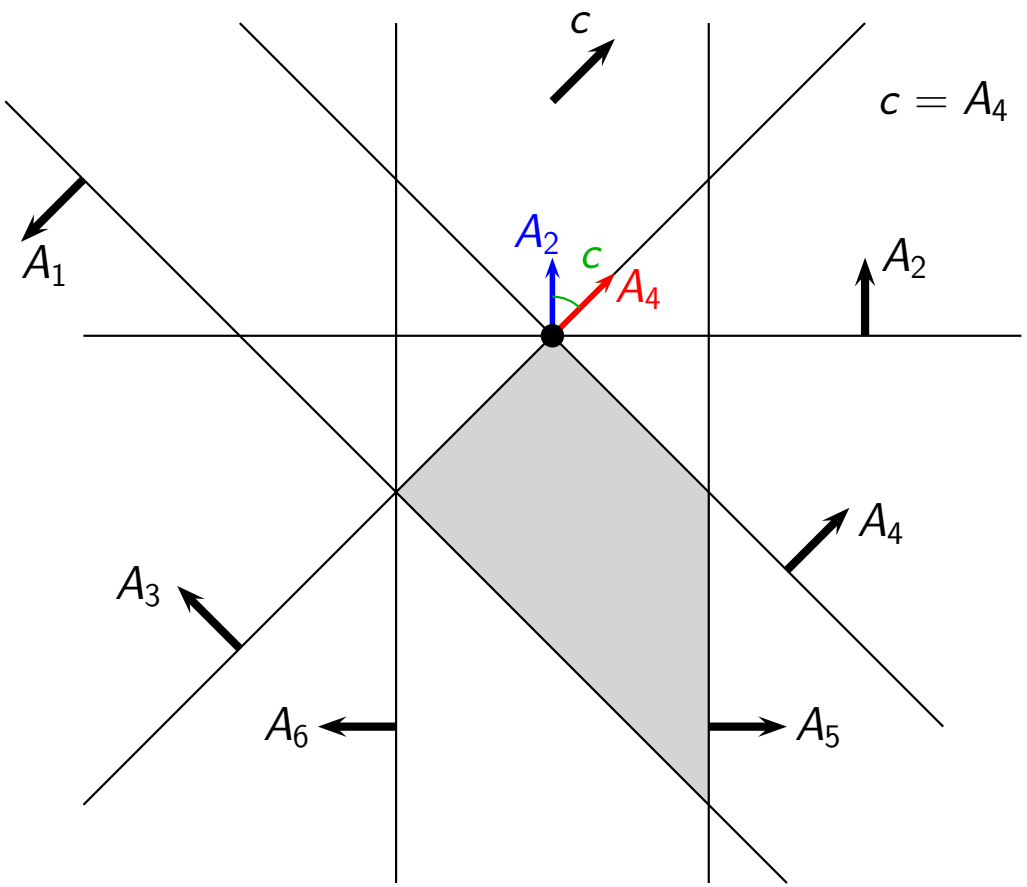
Iterazione 1. $B = \{4, 6\}$, $c = A_4$, quindi $\bar{y}_4 = 1$ e $\bar{y}_6 = 0$. $\bar{x}$ indicata in figura viola i vincoli 2 e 3, $k = \min\{2, 3\} = 2$.

Algoritmo del simplesso duale



Iterazione 1. $B = \{4, 6\}$, $c = A_4$, quindi $\bar{y}_4 = 1$ e $\bar{y}_6 = 0$. $\bar{x}$ indicata in figura viola i vincoli 2 e 3, $k = \min\{2, 3\} = 2$. $\eta_B = A_2 A_B^{-1}$, cioè $A_2 = \eta_B A_B = \eta_4 A_4 + \eta_6 A_6$. $A_2 \in \operatorname{int}\operatorname{cono}(A_4, A_6)$, quindi $\eta_4 > 0$, $\eta_6 > 0$. Poiché $0 = y_6/\eta_6 < y_4/\eta_4$, si ottiene $h = 6$.

Algoritmo del simplesso duale



Iterazione 2. $B = \{2, 4\}$, $c = A_4$, quindi $\bar{y}_2 = 0$ e $\bar{y}_4 = 1$. $\bar{x}$ indicata in figura è ammissibile e quindi è ottima.

Algoritmo del simplesso duale - vertice iniziale

Come si trova una soluzione di base ammissibile per $(\mathcal{D})$?

Possiamo supporre che $c \geq 0$ (cambiando eventualmente segno alle colonne di A relative alle componenti negative di c). Definiamo il problema ausiliario duale:

$$\begin{cases} \min\limits_{(y,\varepsilon)} \sum\limits_{i=1}^{n} \varepsilon_i \\ y^T A + \varepsilon^T = c^T \\ y \geq 0 \\ \varepsilon \geq 0 \end{cases} \qquad (\mathcal{D}_{aux})$$

Gli indici relativi alle variabili ausiliarie $\varepsilon_1, \ldots, \varepsilon_n$ formano una base (la matrice di base è I). La corrispondente soluzione di base è $\bar{y} = 0$, $\bar{\varepsilon} = c \geq 0$ e quindi è ammissibile per $(\mathcal{D}_{aux})$.
A partire da tale base, applichiamo il simplesso duale per risolvere $(\mathcal{D}_{aux})$.

Teorema

1. Se il valore ottimo di $(\mathcal{D}_{aux})$ è > 0, allora $(\mathcal{D})$ non ha soluzioni ammissibili.
2. Se il valore ottimo di $(\mathcal{D}_{aux})$ è $= 0$, allora a partire da una soluzione di base ottima di $(\mathcal{D}_{aux})$ si può costruire una soluzione di base ammissibile per $(\mathcal{D})$.

7 - Programmazione Lineare Intera

Mauro Passacantando

Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2020/21

Relazioni tra PLI e PL

Consideriamo un problema di Programmazione Lineare Intera nella forma

$$\begin{cases} \max \ c^{\mathsf{T}} x \\ A x \leq b \\ x \in \mathbb{Z}^n \end{cases} \tag{P}$$

dove i dati A, b, c sono a componenti intere e la regione ammissibile Ω è limitata.

Teorema

Il problema (P) è NP-hard.

Relazioni tra PLI e PL

Consideriamo un problema di Programmazione Lineare Intera nella forma

$$\begin{cases} \max \ c^{\mathsf{T}} x \\ A x \leq b \\ x \in \mathbb{Z}^n \end{cases} \tag{P}$$

dove i dati A, b, c sono a componenti intere e la regione ammissibile Ω è limitata.

Teorema

Il problema (P) è *NP*-hard.

Definizione

Il problema di Programmazione Lineare

$$\begin{cases} \max \ c^{\mathsf{T}} x \\ A x \leq b \end{cases} \tag{RC}$$

è detto rilassamento continuo di (P).

Relazioni tra PLI e PL

Che relazione c'è tra (P) e (RC)?

Teorema

- Il valore ottimo di (RC) è maggiore o uguale del valore ottimo di (P).

Relazioni tra PLI e PL

Che relazione c'è tra (P) e (RC)?

Teorema

- Il valore ottimo di (RC) è maggiore o uguale del valore ottimo di (P).
- Se la soluzione ottima di (RC) è ammissibile per (P), allora è ottima anche per (P).

Relazioni tra PLI e PL

Che relazione c'è tra (P) e (RC)?

Teorema

- Il valore ottimo di (RC) è maggiore o uguale del valore ottimo di (P).
- Se la soluzione ottima di (RC) è ammissibile per (P), allora è ottima anche per (P).

Spesso la soluzione ottima di (RC) non è ammissibile per (P)...

Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata?

Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

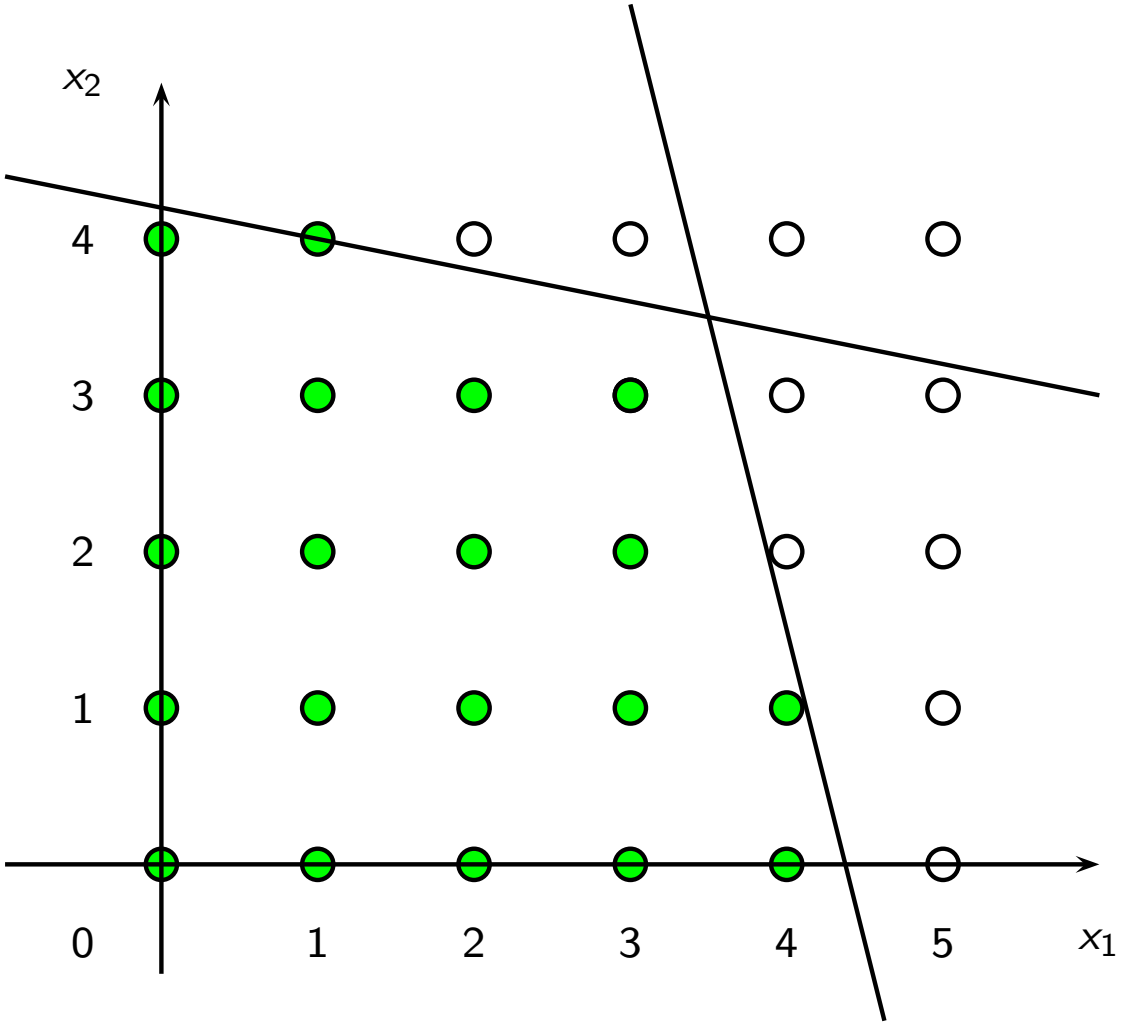
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

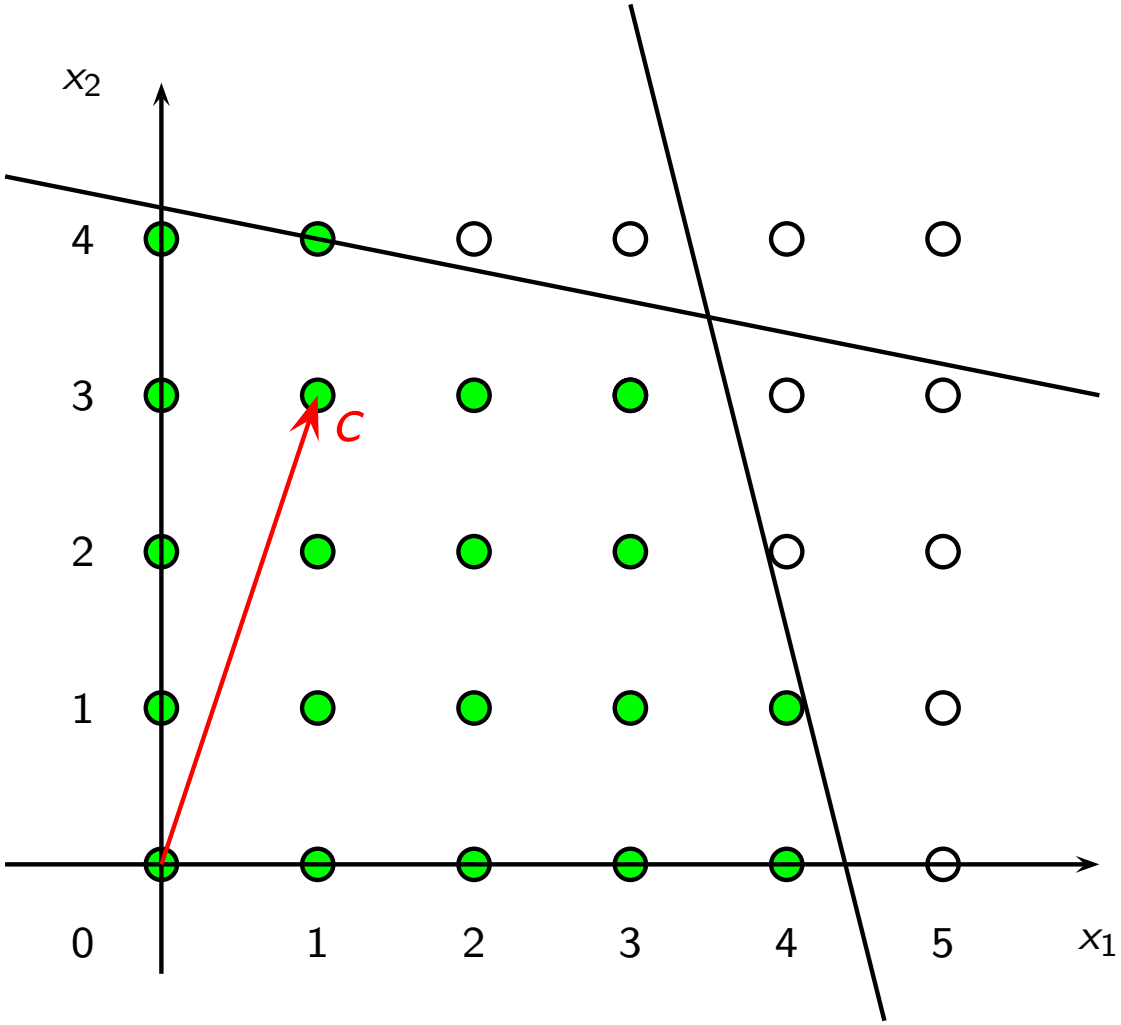


Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

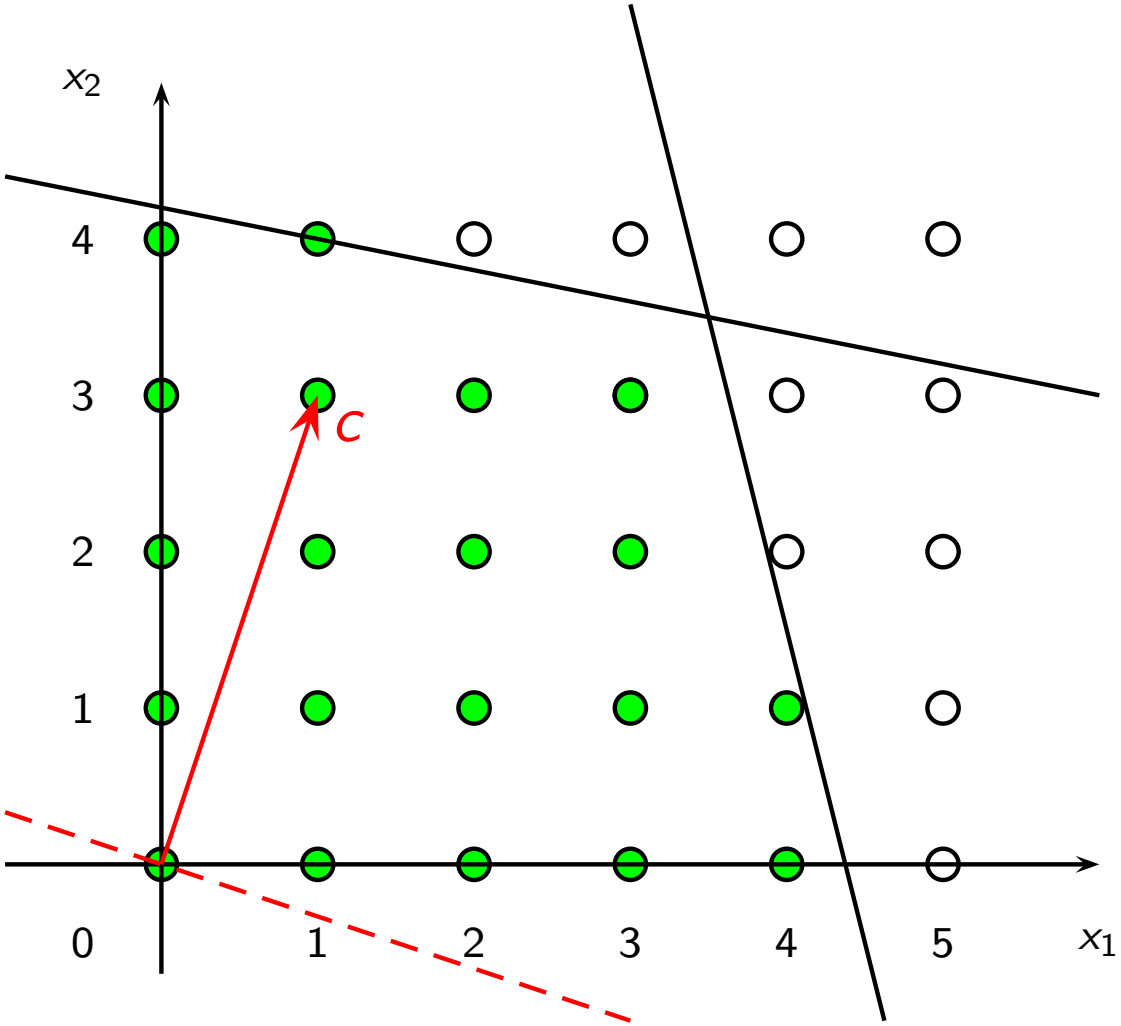


Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

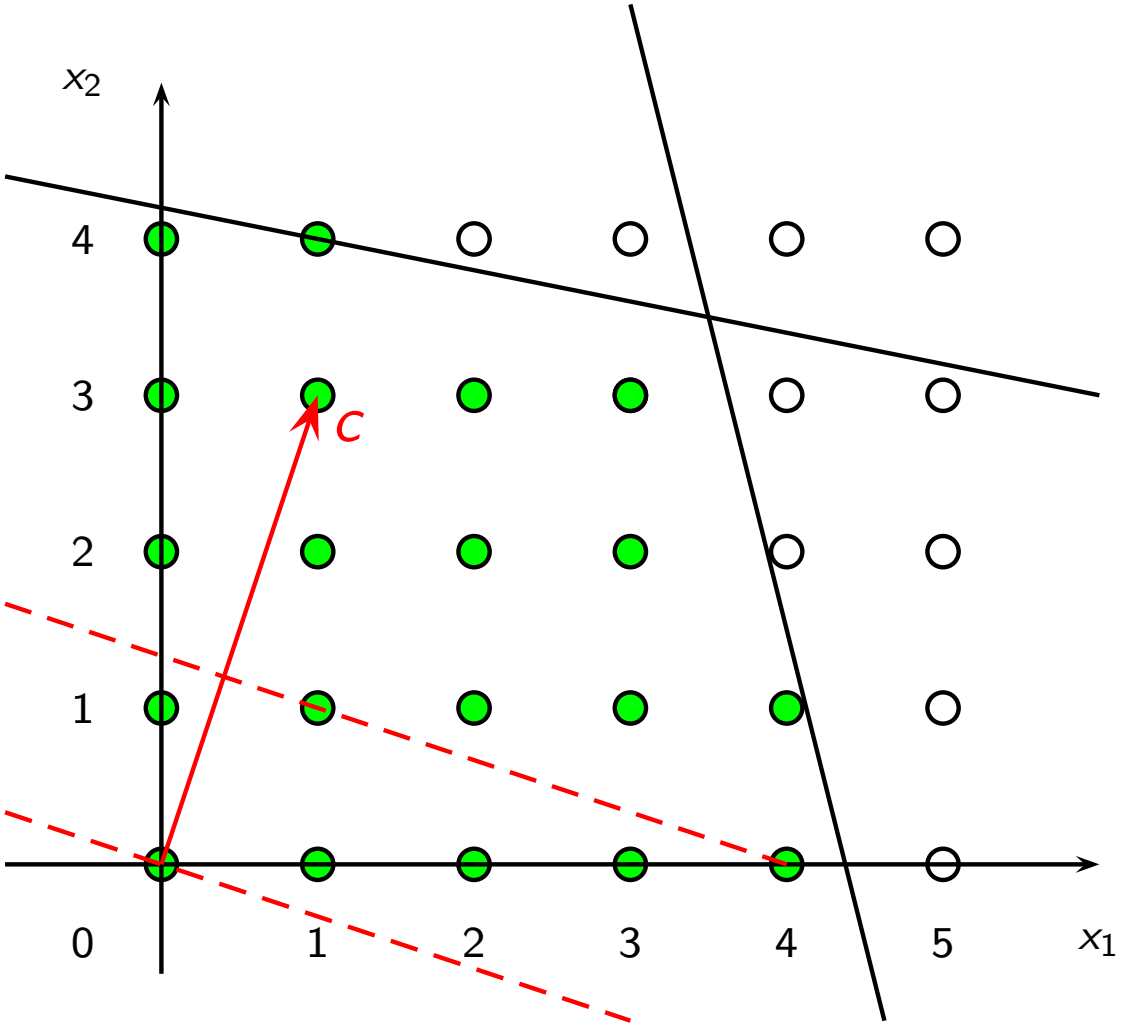


Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

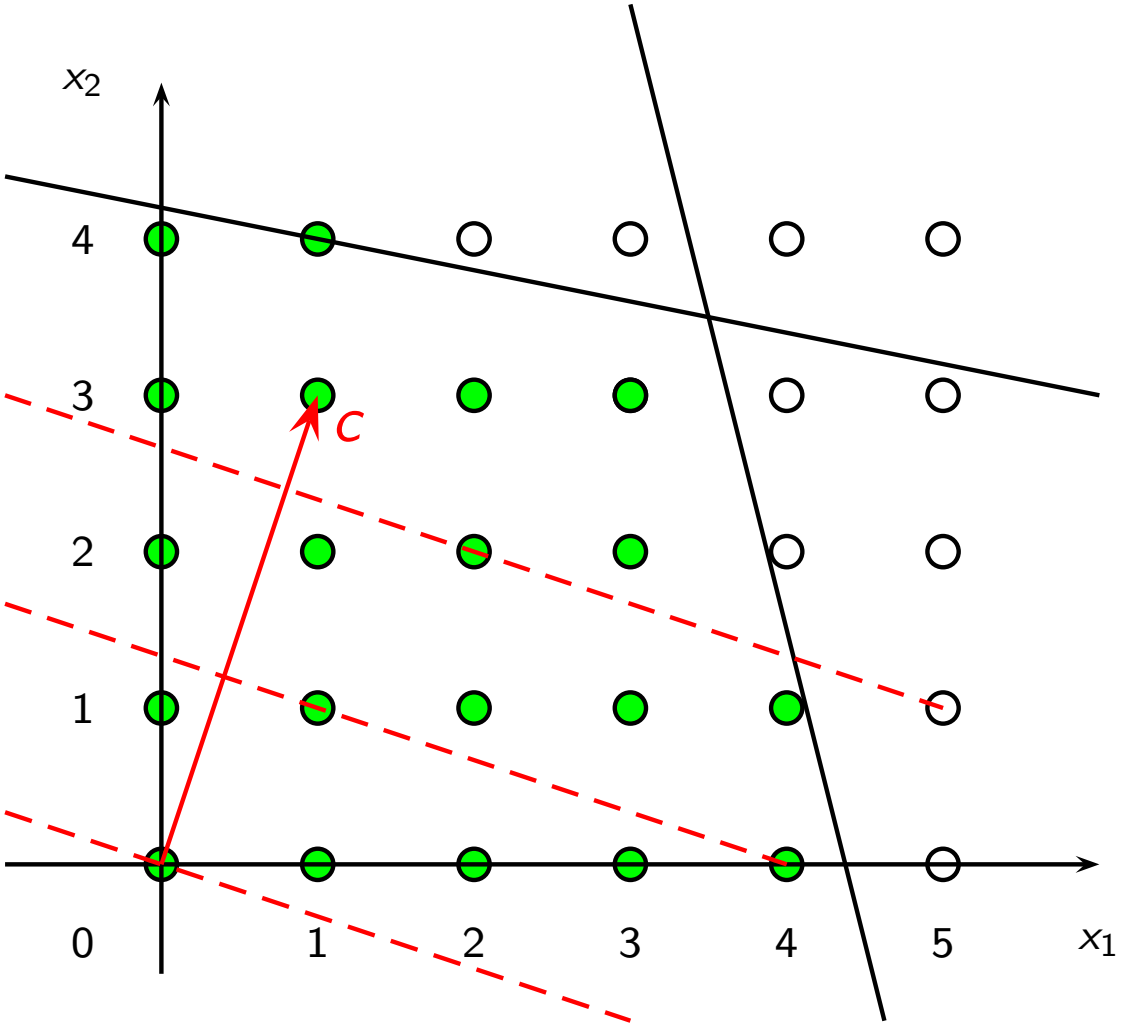


Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

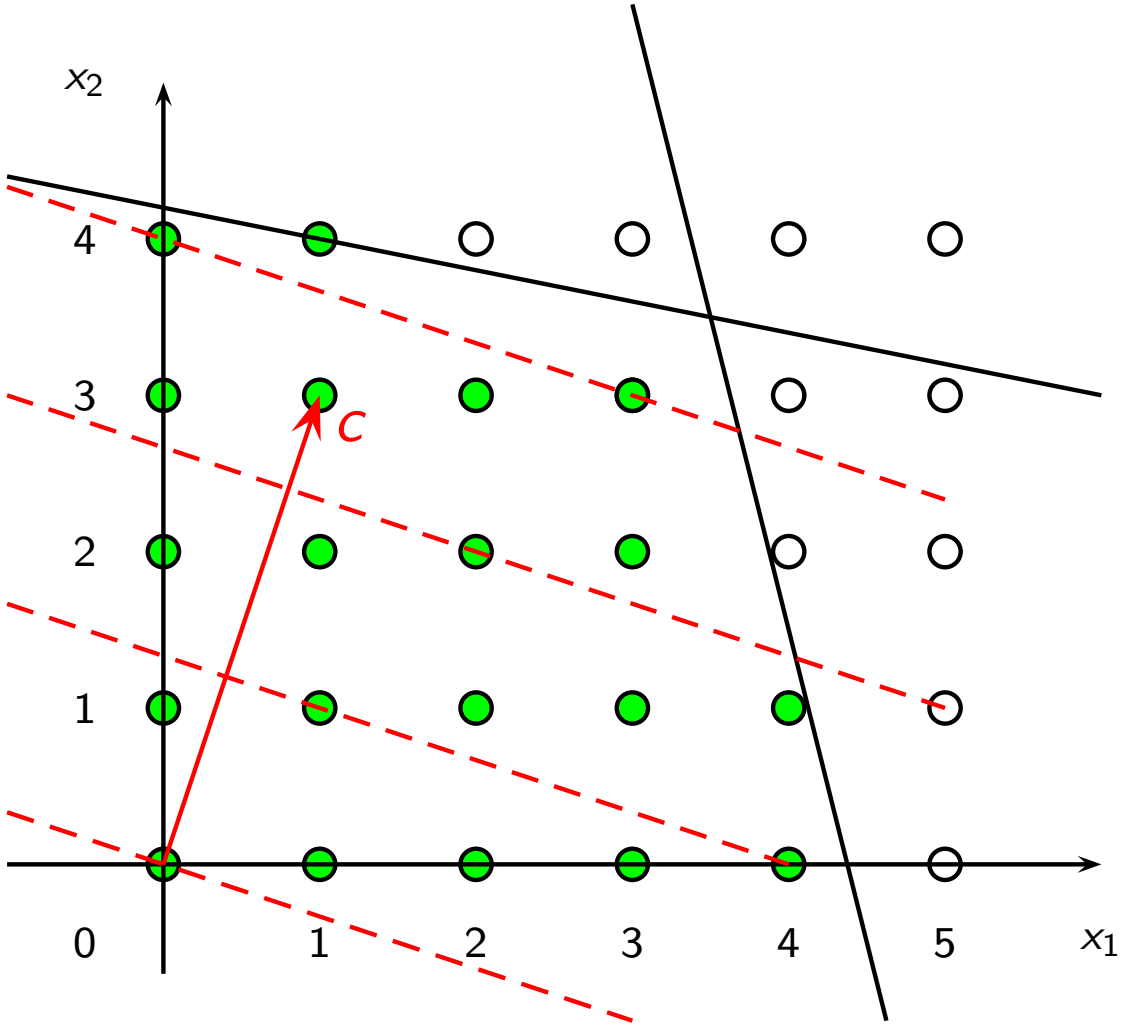


Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$



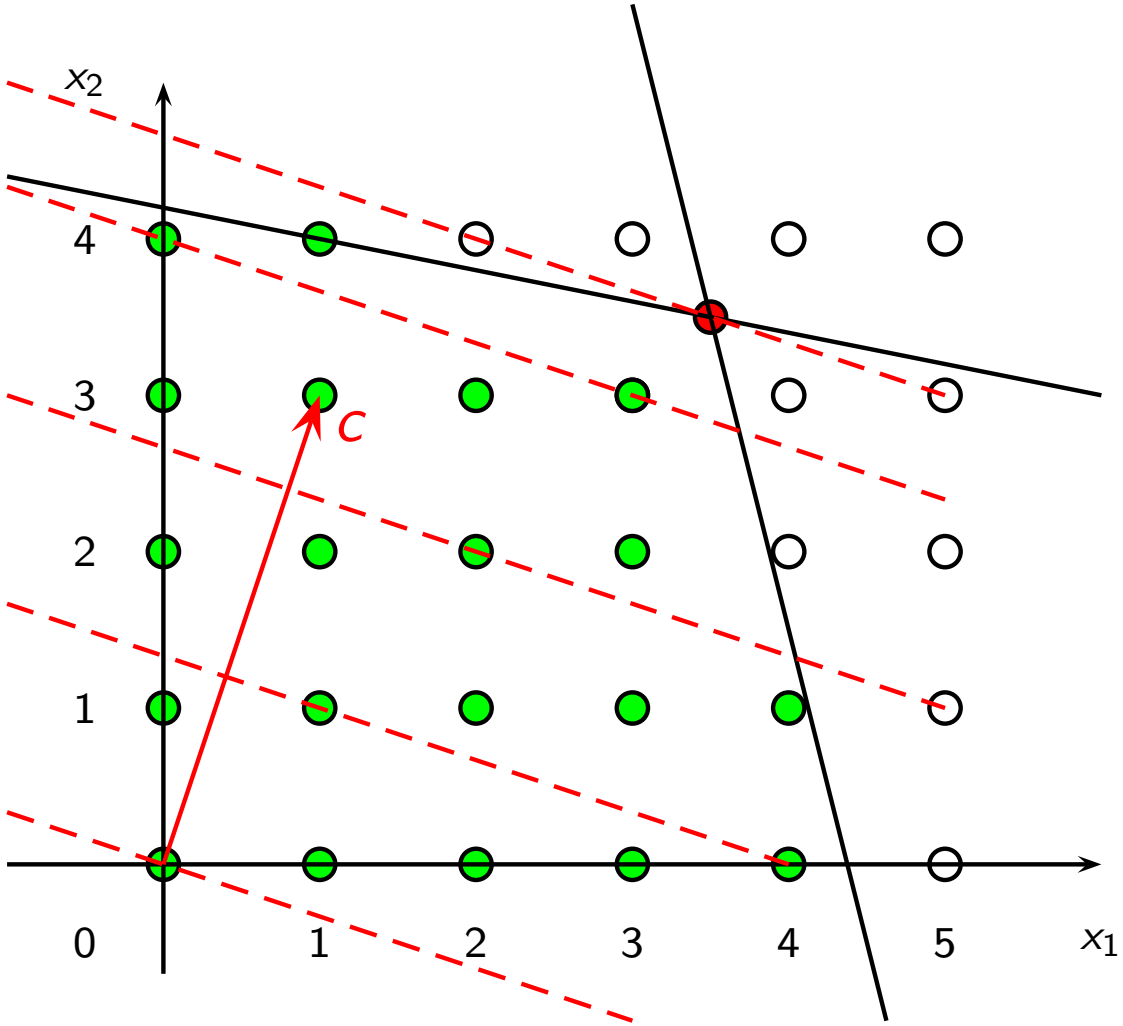
Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottimo rilas. continuo



Relazioni tra PLI e PL

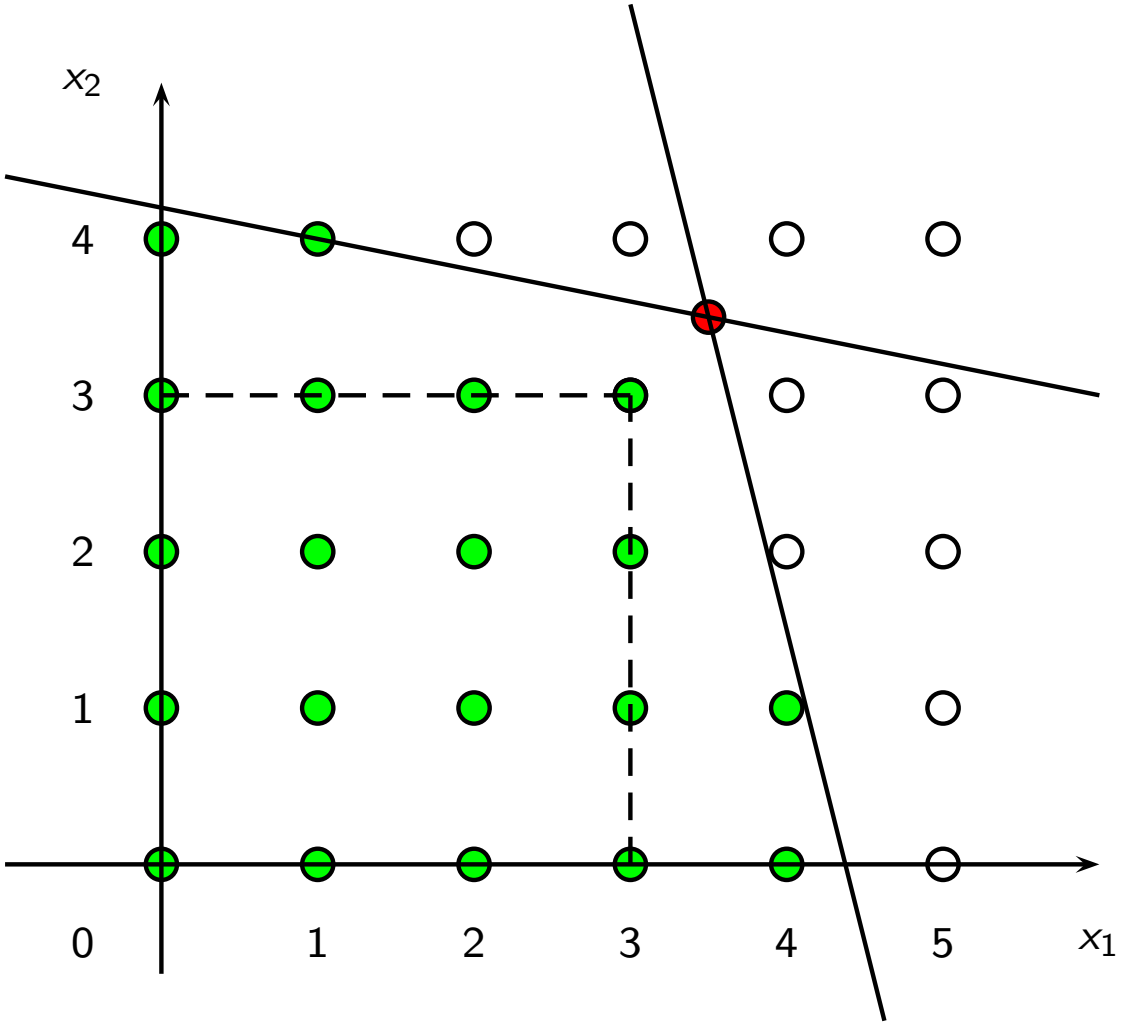
Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottimo rilas. continuo

arrotondamento $\rightarrow (3, 3)$



Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

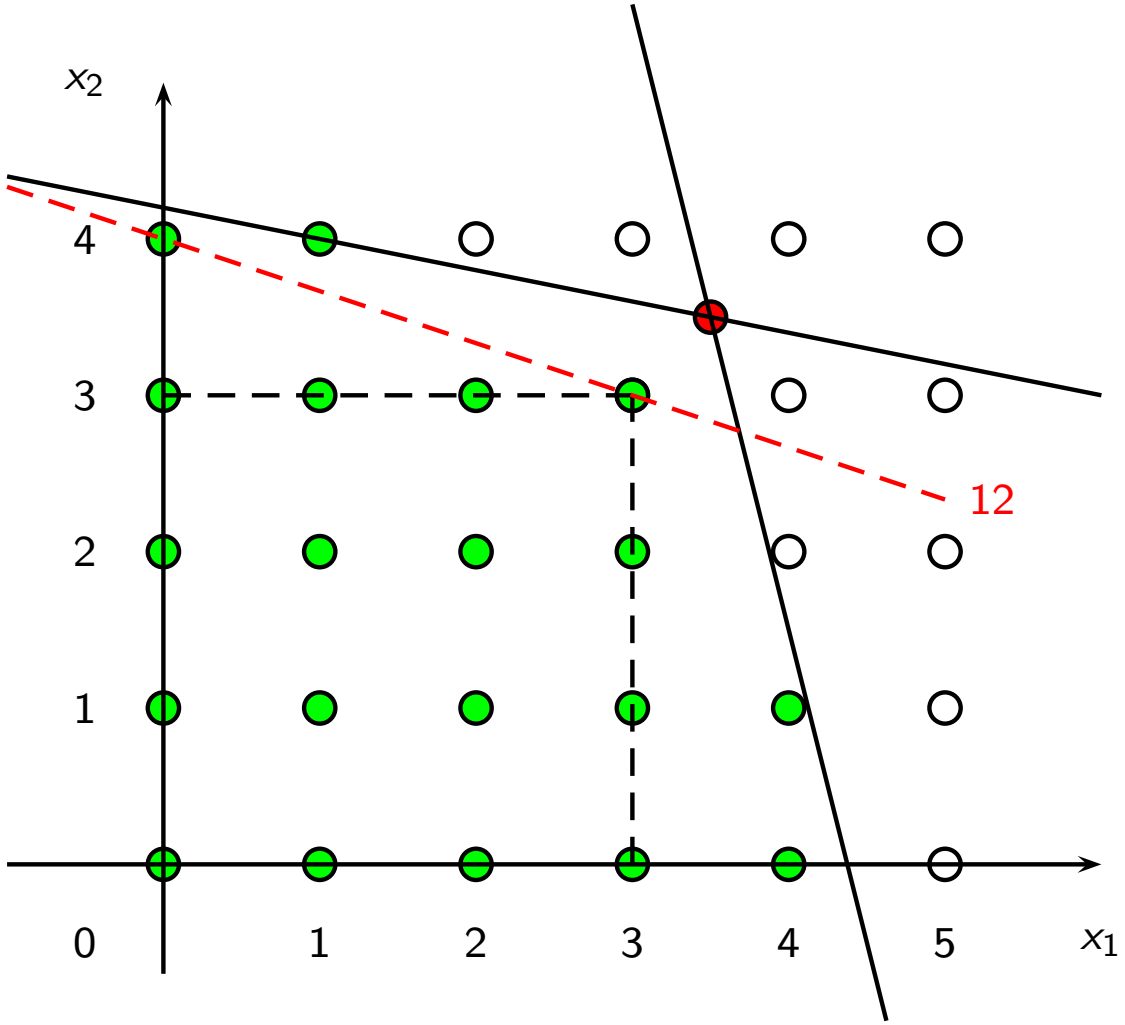
Esempio

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottimo rilas. continuo

arrotondamento $\rightarrow (3, 3)$

$(3, 3)$ non è ottima



Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e arrotondare la soluzione trovata? NO

Esempio

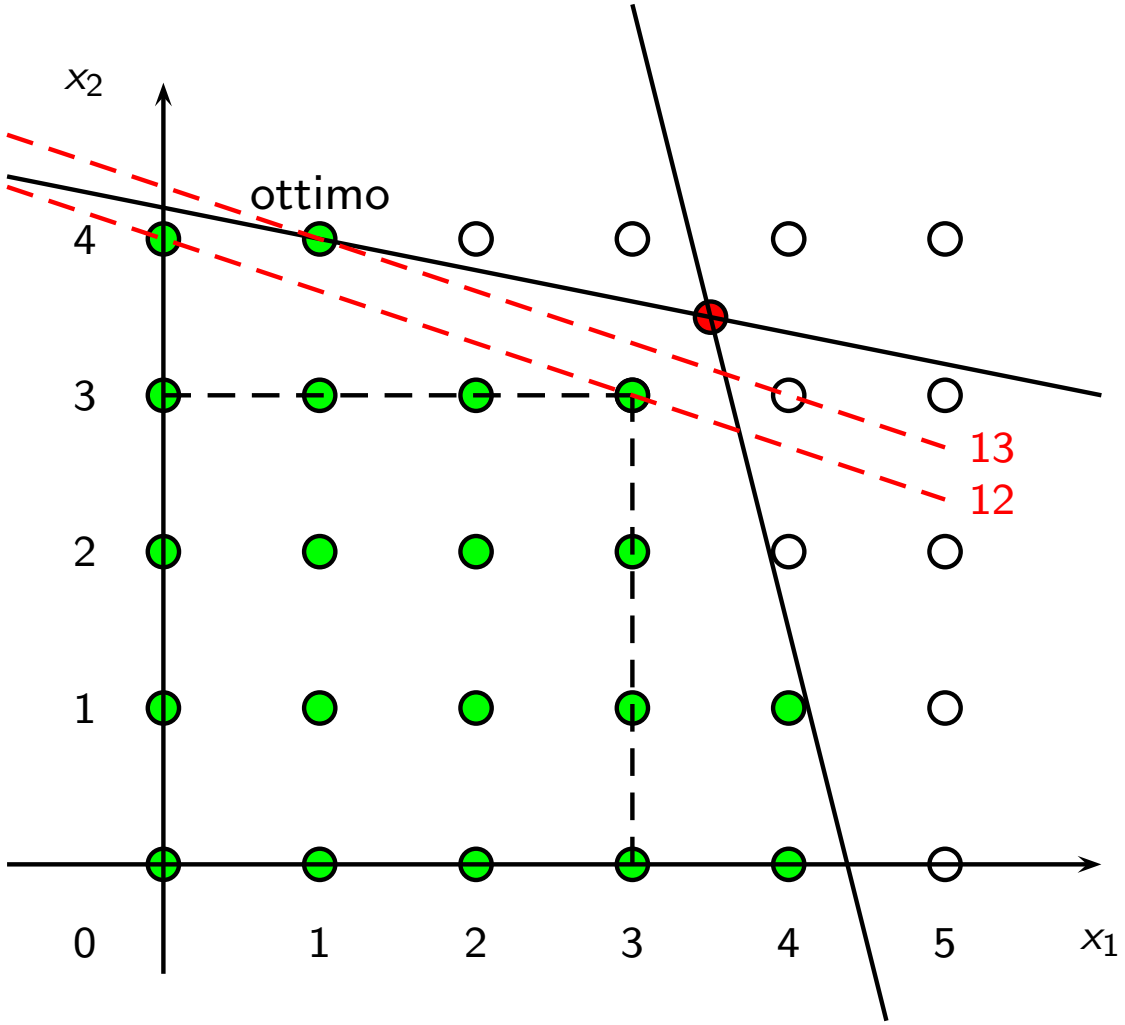
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottimo rilas. continuo

arrotondamento $\rightarrow (3, 3)$

$(3, 3)$ non è ottima

la sol. ottima è $(1, 4)$



Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e scegliere la soluzione ammissibile più vicina rispetto alla distanza euclidea?

Relazioni tra PLI e PL

Per risolvere (P) è sufficiente risolvere il suo rilassamento continuo e scegliere la soluzione ammissibile più vicina rispetto alla distanza euclidea? NO

Esempio

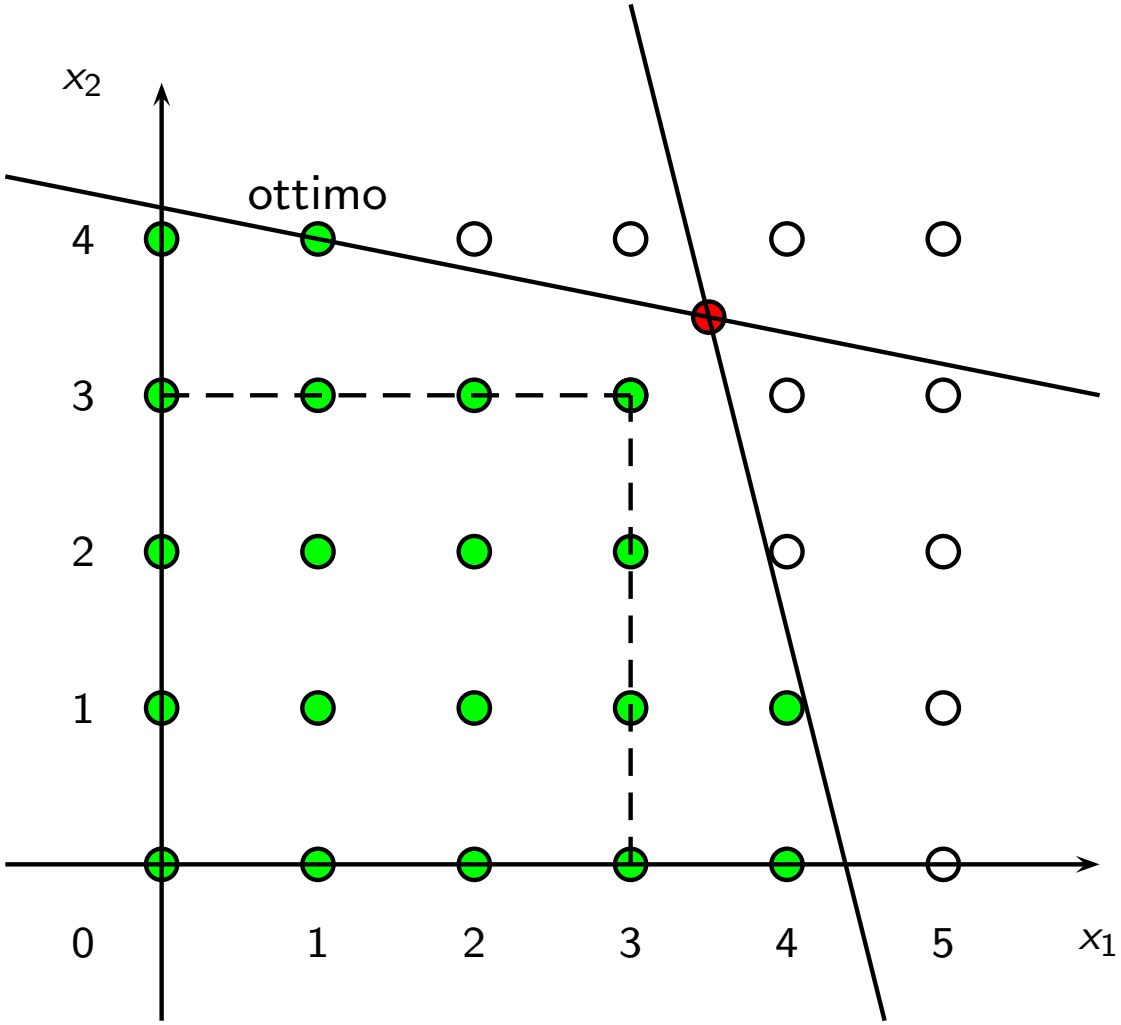
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottimo ril. cont.

sol. più vicina è $(3, 3)$

ma $(3, 3)$ non è ottima

la sol. ottima è $(1, 4)$



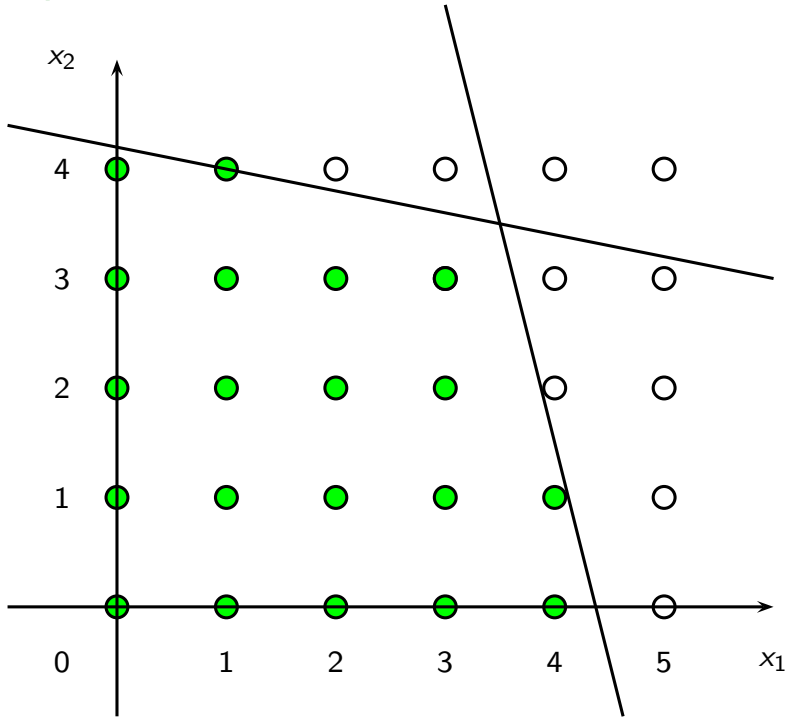
Relazioni tra PLI e PL

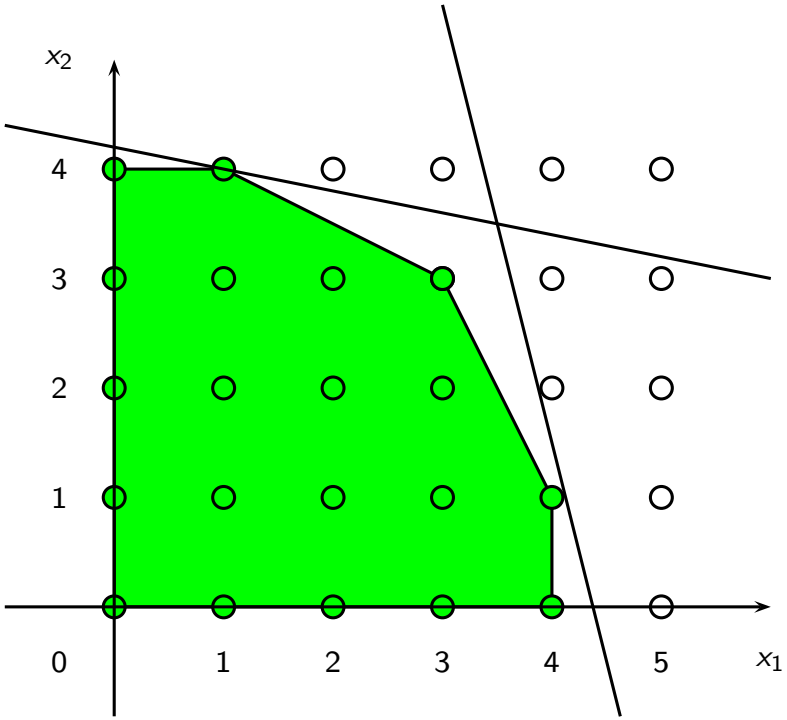
Consideriamo i problemi:

$$\begin{cases} \max c^T x \\ x \in \Omega \end{cases} \qquad \begin{cases} \max c^T x \\ x \in \text{conv}(\Omega) \end{cases}$$

dove conv(Ω) è l'involucro convesso delle soluzioni ammissibili, cioè il più piccolo insieme convesso che contiene Ω.

Esempio





Relazioni tra PLI e PL

Teorema

- $\text{conv}(\Omega)$ è un poliedro

Relazioni tra PLI e PL

Teorema

- conv(Ω) è un poliedro
- I vertici di conv(Ω) appartengono a Ω, cioè sono a componenti intere

Relazioni tra PLI e PL

Teorema

- $\text{conv}(\Omega)$ è un poliedro
- I vertici di $\text{conv}(\Omega)$ appartengono a Ω, cioè sono a componenti intere
- I problemi

$$\begin{cases} \max c^\top x \\ x \in \Omega \end{cases} \qquad \begin{cases} \max c^\top x \\ x \in \text{conv}(\Omega) \end{cases}$$

hanno lo stesso valore ottimo e almeno una soluzione ottima comune

Relazioni tra PLI e PL

Teorema

- $\text{conv}(\Omega)$ è un poliedro
- I vertici di $\text{conv}(\Omega)$ appartengono a Ω, cioè sono a componenti intere
- I problemi

$$\begin{cases} \max c^\top x \\ x \in \Omega \end{cases} \qquad \begin{cases} \max c^\top x \\ x \in \text{conv}(\Omega) \end{cases}$$

  hanno lo stesso valore ottimo e almeno una soluzione ottima comune

Corollario

Il problema di PLI

$$\max_{x \in \Omega} c^\top x$$

è equivalente al problema di PL

$$\max_{x \in \text{conv}(\Omega)} c^\top x$$

Relazioni tra PLI e PL

Teorema

- $\text{conv}(\Omega)$ è un poliedro
- I vertici di $\text{conv}(\Omega)$ appartengono a Ω, cioè sono a componenti intere
- I problemi

$$\begin{cases} \max c^\top x \\ x \in \Omega \end{cases} \qquad \begin{cases} \max c^\top x \\ x \in \text{conv}(\Omega) \end{cases}$$

  hanno lo stesso valore ottimo e almeno una soluzione ottima comune

Corollario

Il problema di PLI

$$\max_{x \in \Omega} c^\top x$$

è equivalente al problema di PL

$$\max_{x \in \text{conv}(\Omega)} c^\top x$$

In generale è difficile trovare i vincoli che definiscono $\text{conv}(\Omega)$ ...

Matrici totalmente unimodulari

Per alcuni particolari problemi si riesce a caratterizzare conv(Ω).

Teorema

Sia $\Omega = \{x \in \mathbb{Z}^n : \ A\,x \leq b\}$. Se A è una matrice totalmente unimodulare (cioè il determinante di ogni sua sottomatrice quadrata è 0 oppure 1 oppure -1), allora

$$\text{conv}(\Omega) = \{x \in \mathbb{R}^n : \ A\,x \leq b\},$$

cioè conv(Ω) coincide con la regione ammissibile del rilassamento continuo.

Matrici totalmente unimodulari

Per alcuni particolari problemi si riesce a caratterizzare conv(Ω).

Teorema

Sia $\Omega = \{x \in \mathbb{Z}^n : \ A\,x \leq b\}$. Se A è una matrice totalmente unimodulare (cioè il determinante di ogni sua sottomatrice quadrata è 0 oppure 1 oppure -1), allora

$$\text{conv}(\Omega) = \{x \in \mathbb{R}^n : \ A\,x \leq b\},$$

cioè conv(Ω) coincide con la regione ammissibile del rilassamento continuo.

Esempi

In un problema di flusso di costo minimo con variabili intere:

$$\begin{cases} \min \ c^\top x \\ E\,x = b \\ 0 \leq x \leq u \\ x \in \mathbb{Z}^n \end{cases}$$

la matrice dei vincoli è totalmente unimodulare. Quindi per risolverlo basta trovare un vertice ottimo del suo rilassamento continuo.
Lo stesso vale per il problema del cammino minimo.

Disuguaglianze valide e piani di taglio

In generale è difficile caratterizzare conv(Ω).
Si aggiungono vincoli alla regione ammissibile del rilassamento continuo in modo da approssimare conv(Ω).

Definizioni

La disequazione $p^\mathsf{T} x \leq p_0$ è detta disuguaglianza valida (DV) per l'insieme Ω se

$$p^\mathsf{T} x \leq p_0 \qquad \forall\, x \in \Omega.$$

Un piano di taglio è una disuguaglianza valida $p^\mathsf{T} x \leq p_0$ per Ω tale che $p^\mathsf{T} \bar{x} > p_0$, dove $\bar{x}$ è l'ottimo del rilassamento continuo.

Disuguaglianze valide e piani di taglio

In generale è difficile caratterizzare conv(Ω).
Si aggiungono vincoli alla regione ammissibile del rilassamento continuo in modo da approssimare conv(Ω).

Definizioni

La disequazione $p^\top x \leq p_0$ è detta disuguaglianza valida (DV) per l'insieme Ω se

$$p^\top x \leq p_0 \qquad \forall\, x \in \Omega.$$

Un piano di taglio è una disuguaglianza valida $p^\top x \leq p_0$ per Ω tale che $p^\top \bar{x} > p_0$, dove $\bar{x}$ è l'ottimo del rilassamento continuo.

Idea alla base del metodo dei piani di taglio:
se P è la regione ammissibile del rilassamento continuo e la soluzione ottima $\bar{x}$ di $\max\limits_{x \in P} c^\top x$ appartiene ad Ω, allora $\bar{x}$ è ottima anche per $\max\limits_{x \in \Omega} c^\top x$;
altrimenti si costruisce un piano di taglio $p^\top x \leq p_0$ in modo da tagliare fuori $\bar{x}$ e poi risolvere il nuovo problema di PL:

$$\begin{cases} \max \ c^\top x \\ x \in P \\ p^\top x \leq p_0 \end{cases}$$

Piani di taglio di Gomory

Supponiamo che il problema di PLI sia nella forma

$$\begin{cases} \max \ c^\top x \\ A\,x = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases} \tag{P}$$

e che $\bar{x}$ sia una soluzione di base (relativa alla base B) ottima del rilassamento continuo di (P). Poniamo:

$$A = (A_B \ A_N) \qquad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} \qquad \widetilde{A} = A_B^{-1} A_N \qquad \widetilde{b} = \bar{x}_B.$$

La parte frazionaria di un numero reale z è $\{z\} := z - \lfloor z \rfloor$, dove $\lfloor z \rfloor$ è la parte intera inferiore di z (o arrotondamento per difetto all'intero più vicino).
Esempio: $\{2.3\} = 2.3 - 2 = 0.3, \qquad \{-1.4\} = -1.4 - (-2) = 0.6.$

Piani di taglio di Gomory

Supponiamo che il problema di PLI sia nella forma

$$\begin{cases} \max \ c^\top x \\ A\,x = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases} \tag{P}$$

e che $\bar{x}$ sia una soluzione di base (relativa alla base B) ottima del rilassamento continuo di (P). Poniamo:

$$A = (A_B \ A_N) \qquad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} \qquad \widetilde{A} = A_B^{-1} A_N \qquad \widetilde{b} = \bar{x}_B.$$

La parte frazionaria di un numero reale z è $\{z\} := z - \lfloor z \rfloor$, dove $\lfloor z \rfloor$ è la parte intera inferiore di z (o arrotondamento per difetto all'intero più vicino).
Esempio: $\{2.3\} = 2.3 - 2 = 0.3, \qquad \{-1.4\} = -1.4 - (-2) = 0.6.$

Teorema

Se esiste un indice $r \in B$ tale che $\widetilde{b}_r \notin \mathbb{Z}$, allora

$$\sum_{j \in N} \{\widetilde{a}_{rj}\} x_j \geq \{\widetilde{b}_r\}$$

è un piano di taglio (detto di Gomory) per il problema (P).

Piani di taglio di Gomory

Dimostrazione

Fissiamo $x \in \Omega$. Allora

$$A\,x = A_B\,x_B + A_N\,x_N = b,$$

quindi

$$x_B = A_B^{-1}b - A_B^{-1}\,A_N\,x_N = \widetilde{b} - \widetilde{A}\,x_N.$$

Definiamo il vettore $\widetilde{x} = x_B$. Quindi:

$$\begin{aligned}
\widetilde{x}_r &= \widetilde{b}_r - \sum_{j \in N} \widetilde{a}_{rj} x_j \\
&= \lfloor \widetilde{b}_r \rfloor + \{\widetilde{b}_r\} - \sum_{j \in N} \left(\lfloor \widetilde{a}_{rj} \rfloor + \{\widetilde{a}_{rj}\} \right) x_j \\
&= \lfloor \widetilde{b}_r \rfloor + \{\widetilde{b}_r\} - \sum_{j \in N} \lfloor \widetilde{a}_{rj} \rfloor x_j - \sum_{j \in N} \{\widetilde{a}_{rj}\} x_j,
\end{aligned}$$

di conseguenza

Piani di taglio di Gomory

Dimostrazione (segue)

$$\sum_{j \in N} \{\widetilde{a}_{rj}\} x_j - \{\widetilde{b}_r\} = \lfloor \widetilde{b}_r \rfloor - \sum_{j \in N} \lfloor \widetilde{a}_{rj} \rfloor x_j - \widetilde{x}_r \in \mathbb{Z}. \tag{1}$$

Inoltre si ha:

$$\sum_{j \in N} \{\widetilde{a}_{rj}\} x_j - \{\widetilde{b}_r\} \geq -\{\widetilde{b}_r\} > -1. \tag{2}$$

Dalle relazioni (1) e (2), otteniamo che:

$$\sum_{j \in N} \{\widetilde{a}_{rj}\} x_j \geq \{\widetilde{b}_r\}. \tag{3}$$

quindi la (3) è una DV per Ω. Inoltre la (3) non è soddisfatta da $\bar{x}$, infatti:

$$0 = \sum_{j \in N} \{\widetilde{a}_{rj}\} \bar{x}_j < \{\widetilde{b}_r\},$$

quindi la (3) è un piano di taglio. $\square$

Piani di taglio di Gomory

Esempio

Consideriamo di nuovo il problema

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

Trasformiamo i vincoli di $\leq$ in vincoli di $=$ aggiungendo variabili di scarto x_3, x_4:

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 + x_3 = 21 \\ 8\,x_1 + 2\,x_2 + x_4 = 35 \\ x \geq 0 \\ x \in \mathbb{Z}^4 \end{cases}$$

$$A = \begin{pmatrix} 1 & 5 & 1 & 0 \\ 8 & 2 & 0 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 21 \\ 35 \end{pmatrix}, \qquad c^{\mathsf{T}} = (1, 3, 0, 0).$$

Piani di taglio di Gomory

Esempio

La soluzione ottima del rilassamento continuo è $\bar{x} = \left(\frac{7}{2}, \frac{7}{2}, 0, 0\right)$.

Quindi la base ottima è $B = \{1, 2\}$,

$$A_B = \begin{pmatrix} 1 & 5 \\ 8 & 2 \end{pmatrix} \qquad A_B^{-1} = \begin{pmatrix} -\frac{1}{19} & \frac{5}{38} \\ \frac{4}{19} & -\frac{1}{38} \end{pmatrix}, \qquad \widetilde{A} = \begin{pmatrix} -\frac{1}{19} & \frac{5}{38} \\ \frac{4}{19} & -\frac{1}{38} \end{pmatrix}.$$

Poiché $\widetilde{b} = \left(\frac{7}{2}, \frac{7}{2}\right)$ ha entrambe le componenti non intere, esistono due tagli di Gomory.

Piani di taglio di Gomory

Esempio

Se $r = 1$, allora il piano di taglio di Gomory è

$$\left\{-\frac{1}{19}\right\} x_3 + \left\{\frac{5}{38}\right\} x_4 \geq \left\{\frac{7}{2}\right\},$$

cioè

$$\frac{18}{19} x_3 + \frac{5}{38} x_4 \geq \frac{1}{2},$$

ossia

$$36x_3 + 5x_4 \geq 19,$$

che nelle variabili (x_1, x_2) equivale a

$$36(21 - x_1 - 5x_2) + 5(35 - 8x_1 - 2x_2) \geq 19$$

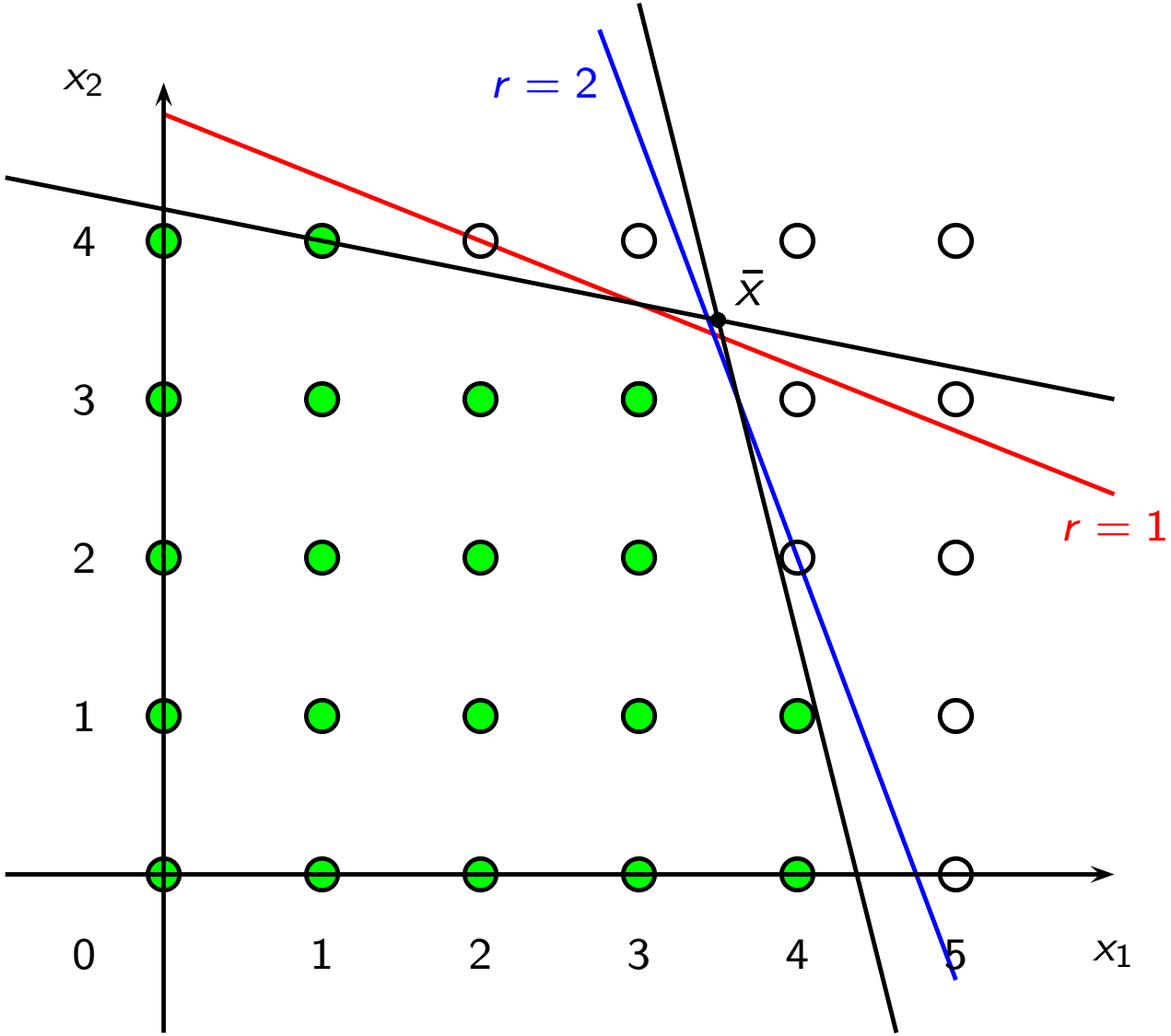
cioè

$$2x_1 + 5x_2 \leq 24.$$

Analogamente, per $r = 2$ si ottiene il piano di taglio $8x_1 + 3x_2 \leq 38$.

Piani di taglio di Gomory

Esempio



Esercizio 7.1

Si consideri il seguente problema di programmazione lineare intera:

$$\begin{cases} \max\ 12\ x_1 + 5\ x_2 \\ 8\ x_1 + 15\ x_2 \leq 45 \\ 15\ x_1 + 11\ x_2 \leq 63 \\ x_1, x_2 \geq 0 \\ x_1, x_2 \in \mathbb{Z} \end{cases}$$

a) Disegnare l'insieme conv(Ω), dove Ω è la regione ammissibile del problema.

b) Risolvere graficamente il rilassamento continuo del problema.

c) Calcolare un taglio di Gomory.

Esercizio 7.2

Si consideri il seguente problema di programmazione lineare intera:

$$\begin{cases} \min \ 6\,x_1 + 13\,x_2 \\ 17\,x_1 + 7\,x_2 \geq 63 \\ 7\,x_1 + 6\,x_2 \geq 30 \\ x_1, x_2 \geq 0 \\ x_1, x_2 \in \mathbb{Z} \end{cases}$$

a) Disegnare l'insieme conv(Ω), dove Ω è la regione ammissibile del problema.

b) Risolvere graficamente il rilassamento continuo del problema.

c) Calcolare un taglio di Gomory.

Enumerazione esplicita

Consideriamo il problema di PLI

$$\begin{cases} \max\ 5\,x_1 + 6\,x_2 \\ 3\,x_1 + 4\,x_2 \leq 7 \\ x1 \geq 0,\ x_2 \geq 0 \\ x \in \mathbb{Z}^2 \end{cases} \qquad \text{(P)}$$

Enumerazione esplicita

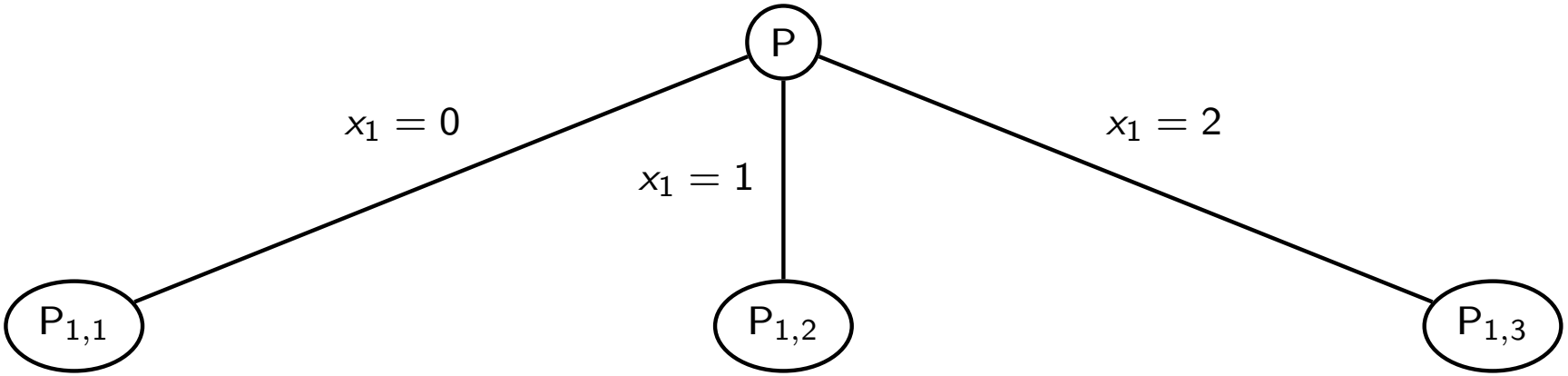
Consideriamo il problema di PLI

$$\begin{cases} \max\ 5\,x_1 + 6\,x_2 \\ 3\,x_1 + 4\,x_2 \leq 7 \\ x1 \geq 0,\ x_2 \geq 0 \\ x \in \mathbb{Z}^2 \end{cases} \qquad \text{(P)}$$

I vincoli implicano che $x_1 = 0$ oppure $x_1 = 1$ oppure $x_1 = 2$.
Facciamo una partizione della regione ammissibile Ω in tre sottoinsiemi:

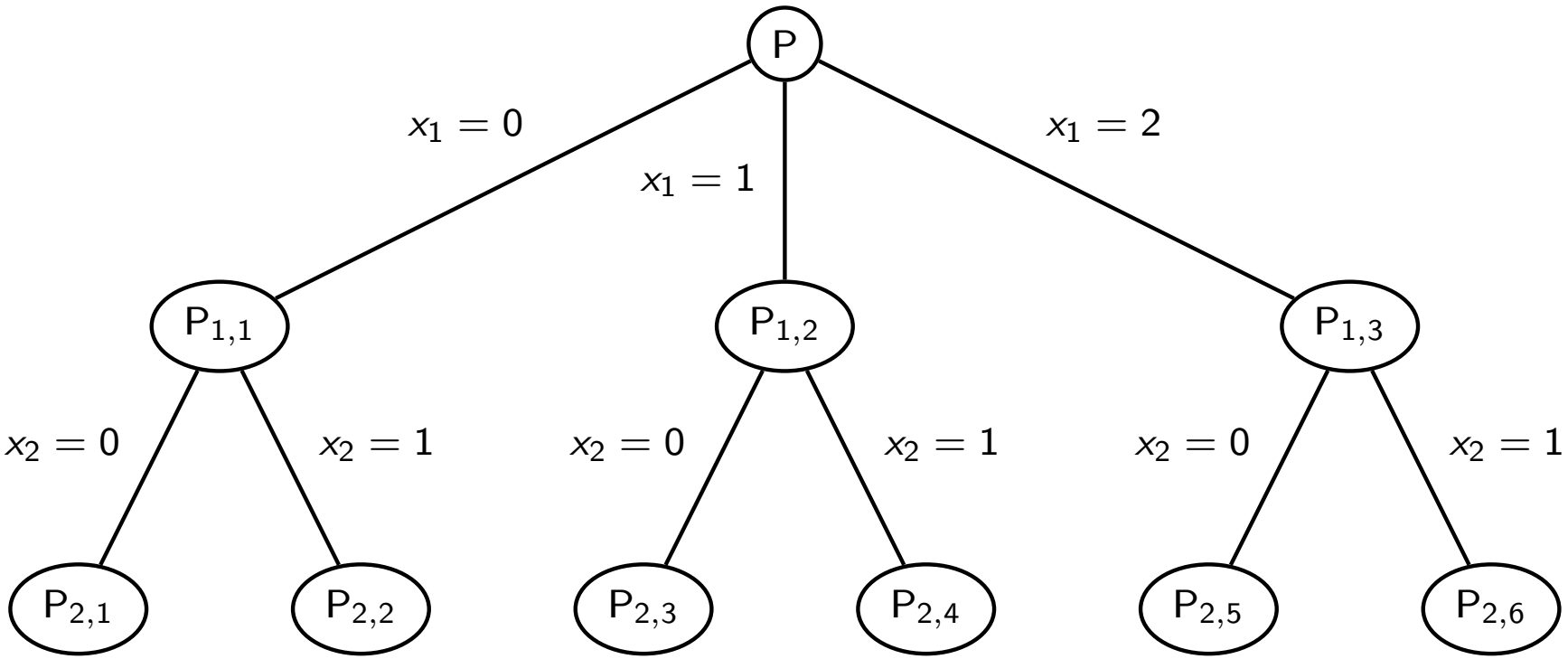
$$\Omega = (\Omega \cap \{x_1 = 0\}) \cup (\Omega \cap \{x_1 = 1\}) \cup (\Omega \cap \{x_1 = 2\})$$

che corrispondono al primo livello dell'albero decisionale:



Enumerazione esplicita

Analogamente, $x_2 = 0$ oppure $x_2 = 1$. Pertanto, l'albero decisionale completo è



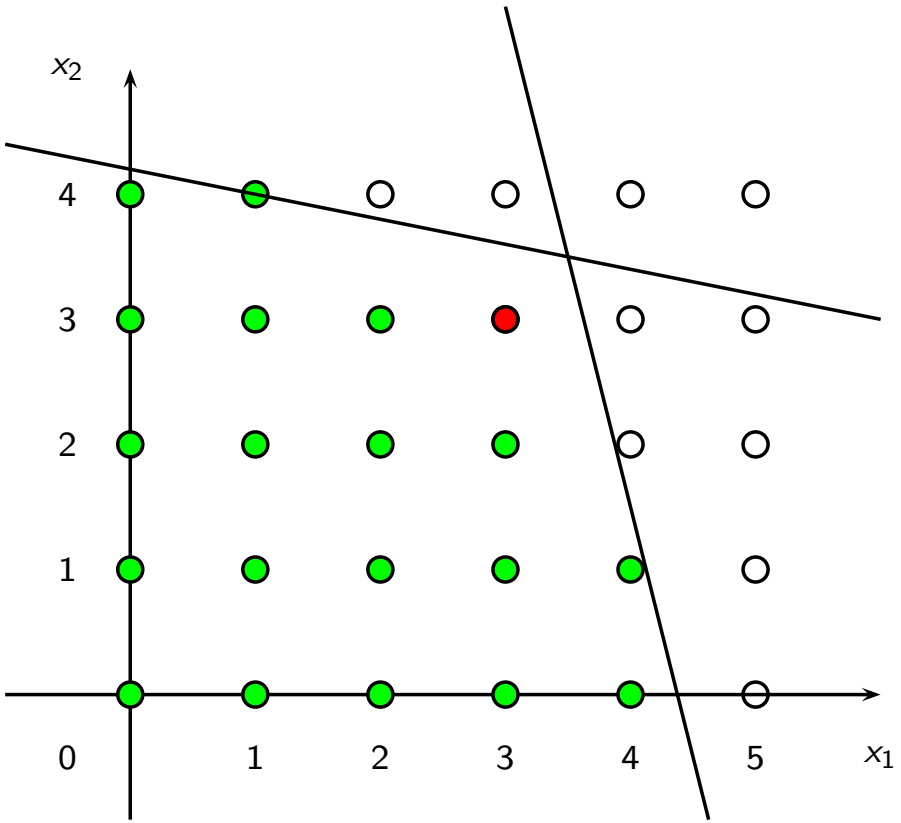
I nodi $P_{2,1}, \ldots, P_{2,5}$ corrispondono a soluzioni ammissibili di $(\mathcal{P})$, mentre il nodo $P_{2,6}$ corrisponde alla soluzione $x = (2, 1)$ che non è ammissibile.
I valori della funzione obiettivo in corrispondenza dei nodi $P_{2,1}, \ldots, P_{2,5}$ sono rispettivamente 0, 6, 5, 11, 10. Pertanto, la soluzione ottima è ottenuta nel nodo $P_{2,4}$, cioè $x^* = (1, 1)$.

Enumerazione implicita

- L'enumerazione esplicita è molto costosa dal punto di vista computazionale
- Possiamo però esaminare (e scartare) le soluzioni a gruppi, anziché singolarmente $\longrightarrow$ enumerazione implicita

Enumerazione implicita

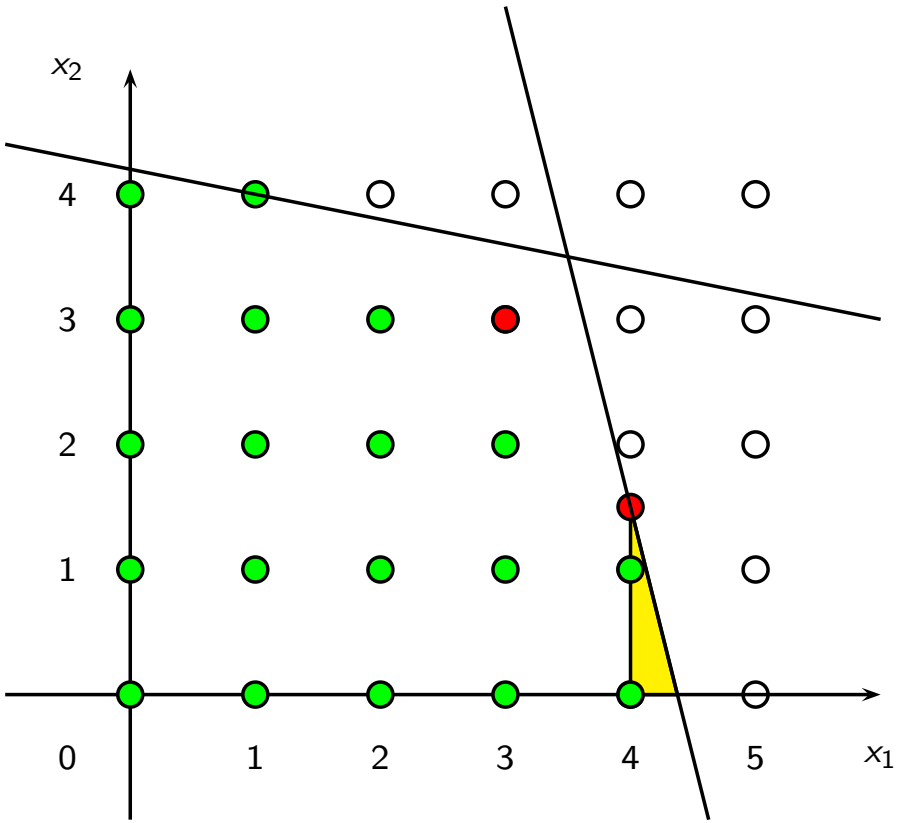
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0, \ x \in \mathbb{Z}^2 \end{cases}$$



- Sappiamo che il punto $(3, 3)$ è ammissibile con valore 12.

Enumerazione implicita

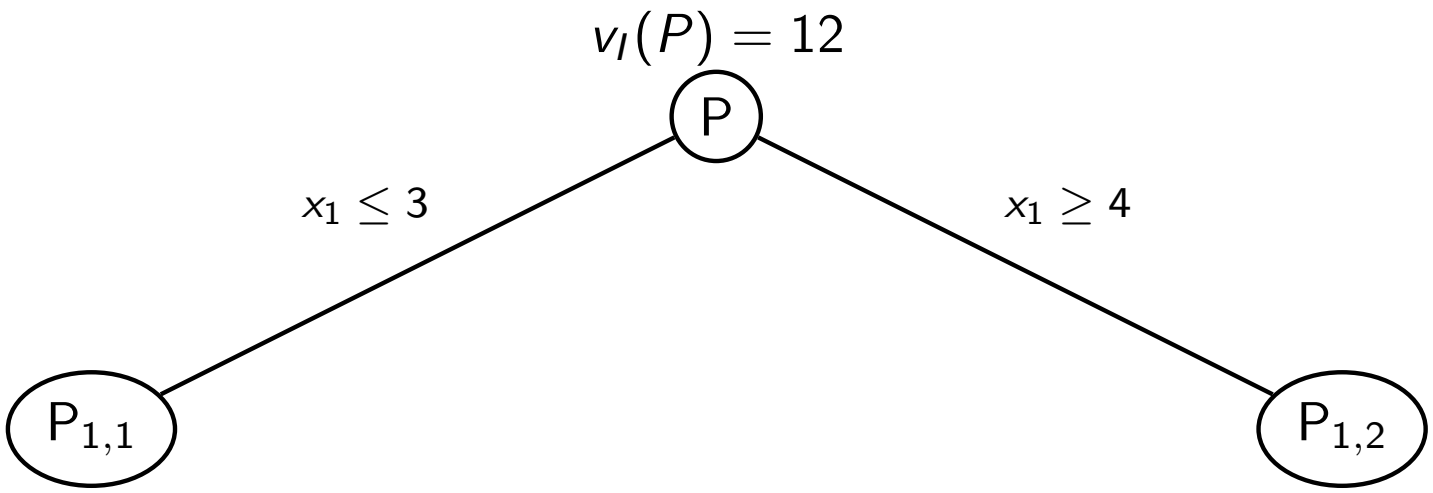
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0, \ x \in \mathbb{Z}^2 \end{cases}$$



- Sappiamo che il punto $(3, 3)$ è ammissibile con valore 12.
- Consideriamo il sottoproblema in cui aggiungiamo il vincolo $x_1 \geq 4$: la soluzione ottima del suo rilassamento continuo è $(4, 3/2)$ con valore 8.5, quindi tutte le soluzioni ammissibili del sottoproblema sono peggiori di $(3, 3)$
$\rightarrow$ enumerazione implicita.

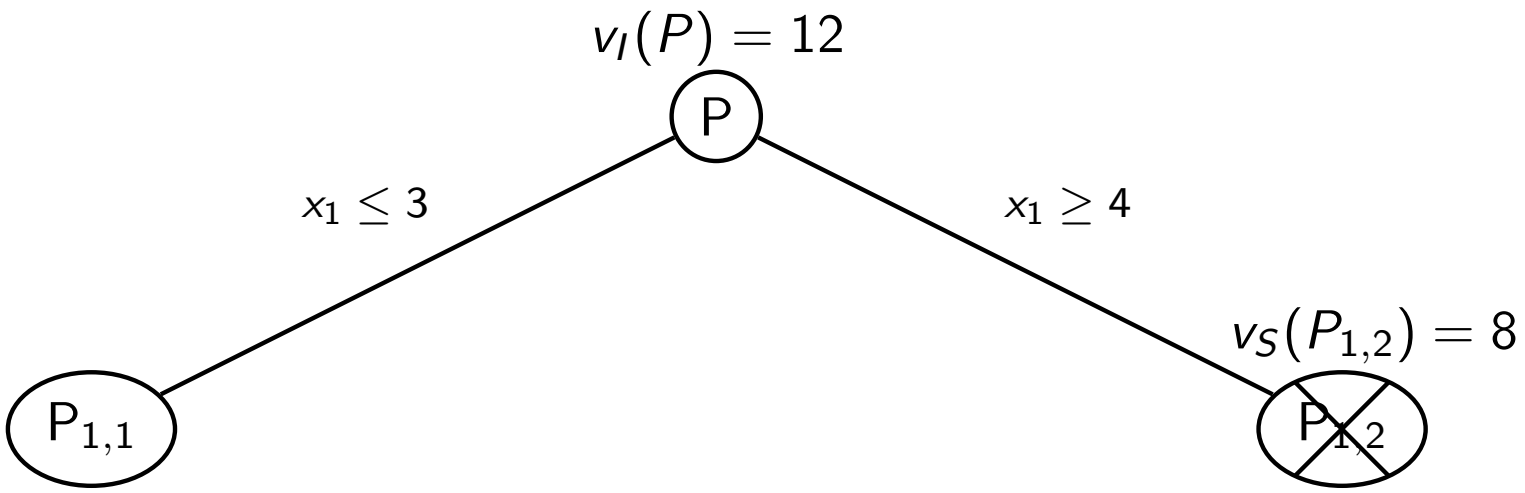
Albero decisionale ridotto

Indichiamo con
$v_I(P)$ una valutazione inferiore (stima per difetto) del valore ottimo di P
$v_S(P)$ una valutazione superiore (stima per eccesso) del valore ottimo di P



Albero decisionale ridotto

Indichiamo con
$v_I(P)$ una valutazione inferiore (stima per difetto) del valore ottimo di P
$v_S(P)$ una valutazione superiore (stima per eccesso) del valore ottimo di P



Metodo Branch and Bound

Componenti principali

Metodo Branch and Bound

Componenti principali

- *Branch (ramificazione)*: partizionare la regione ammissibile in sotto-regioni

Metodo Branch and Bound

Componenti principali

- *Branch (ramificazione)*: partizionare la regione ammissibile in sotto-regioni
- *Bound (valutazione)*: stimare il valore ottimo di ogni sotto-problema

Metodo Branch and Bound

Componenti principali

- *Branch (ramificazione)*: partizionare la regione ammissibile in sotto-regioni
- *Bound (valutazione)*: stimare il valore ottimo di ogni sotto-problema
- *Potatura*: scartare le sotto-regioni che non contengono soluzioni migliori di quella corrente (o chiudere i corrispondenti nodi nell'albero decisionale)

Metodo Branch and Bound

Componenti principali

- *Branch (ramificazione)*: partizionare la regione ammissibile in sotto-regioni
- *Bound (valutazione)*: stimare il valore ottimo di ogni sotto-problema
- *Potatura*: scartare le sotto-regioni che non contengono soluzioni migliori di quella corrente (o chiudere i corrispondenti nodi nell'albero decisionale)
- *Visita*: in quale ordine visitare i nodi nell'albero decisionale

Regole di Branch

Come partizionare la regione ammissibile?

- Le sotto-regioni si generano aggiungendo vincoli

Regole di Branch

Come partizionare la regione ammissibile?

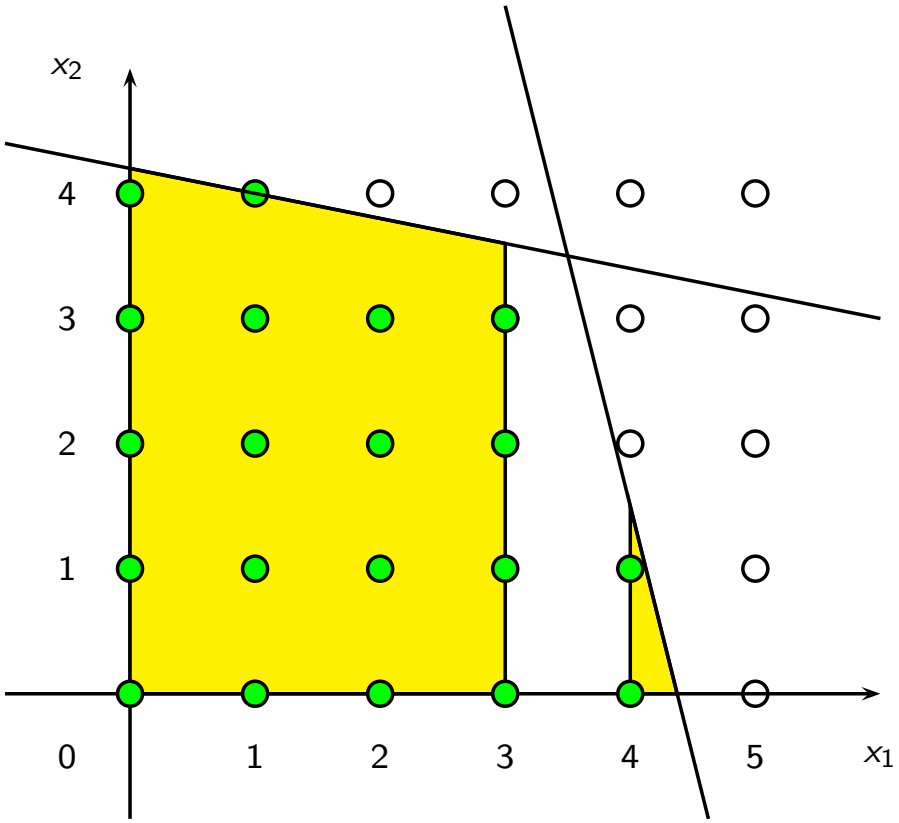
- Le sotto-regioni si generano aggiungendo vincoli
- Il risultato deve essere una partizione della regione ammissibile Ω, per non perdere nessuna soluzione ammissibile (che potrebbe essere ottima)

Branch binario

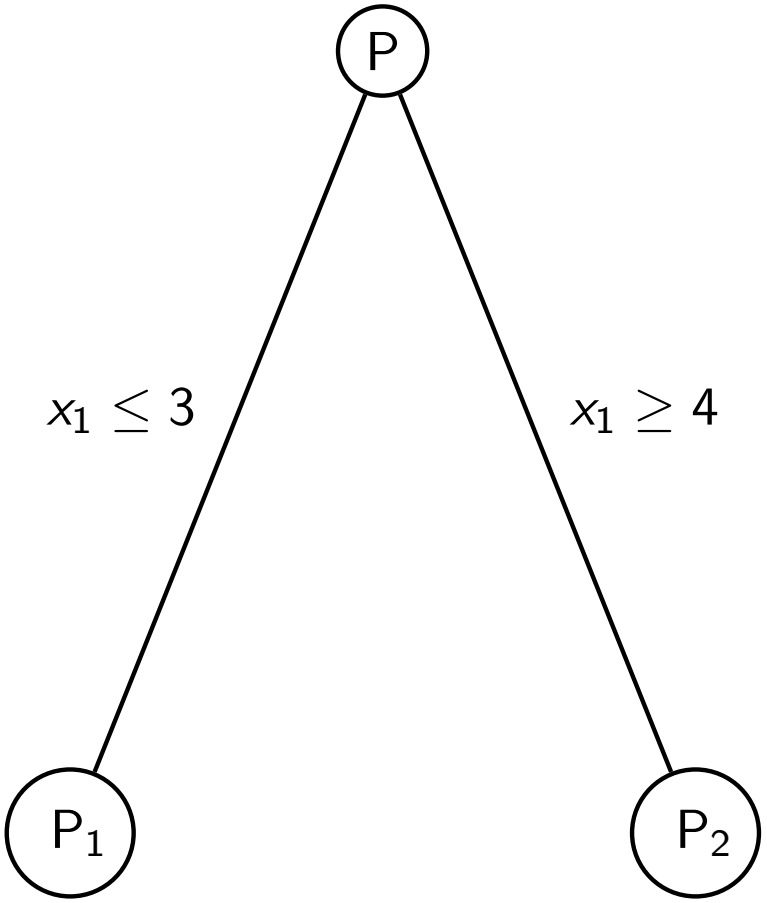
Branch binario

Ogni regione viene partizionata in due sotto-regioni, aggiungendo due vincoli

Esempio: $x_1 \leq 3$ e $x_1 \geq 4$

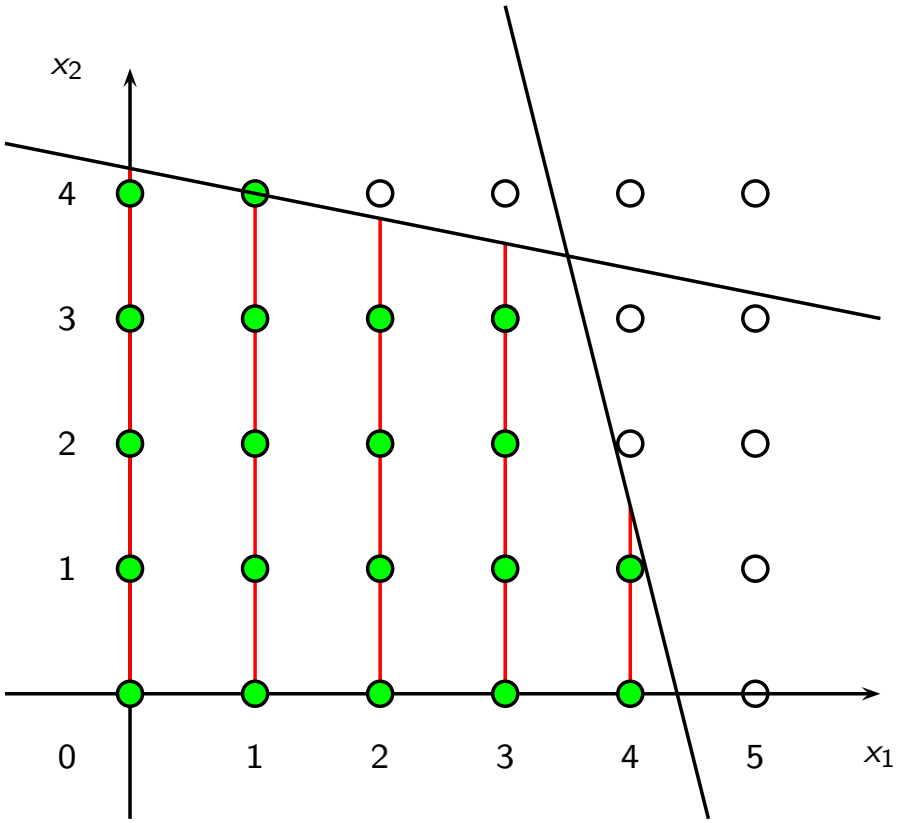


Branch binario

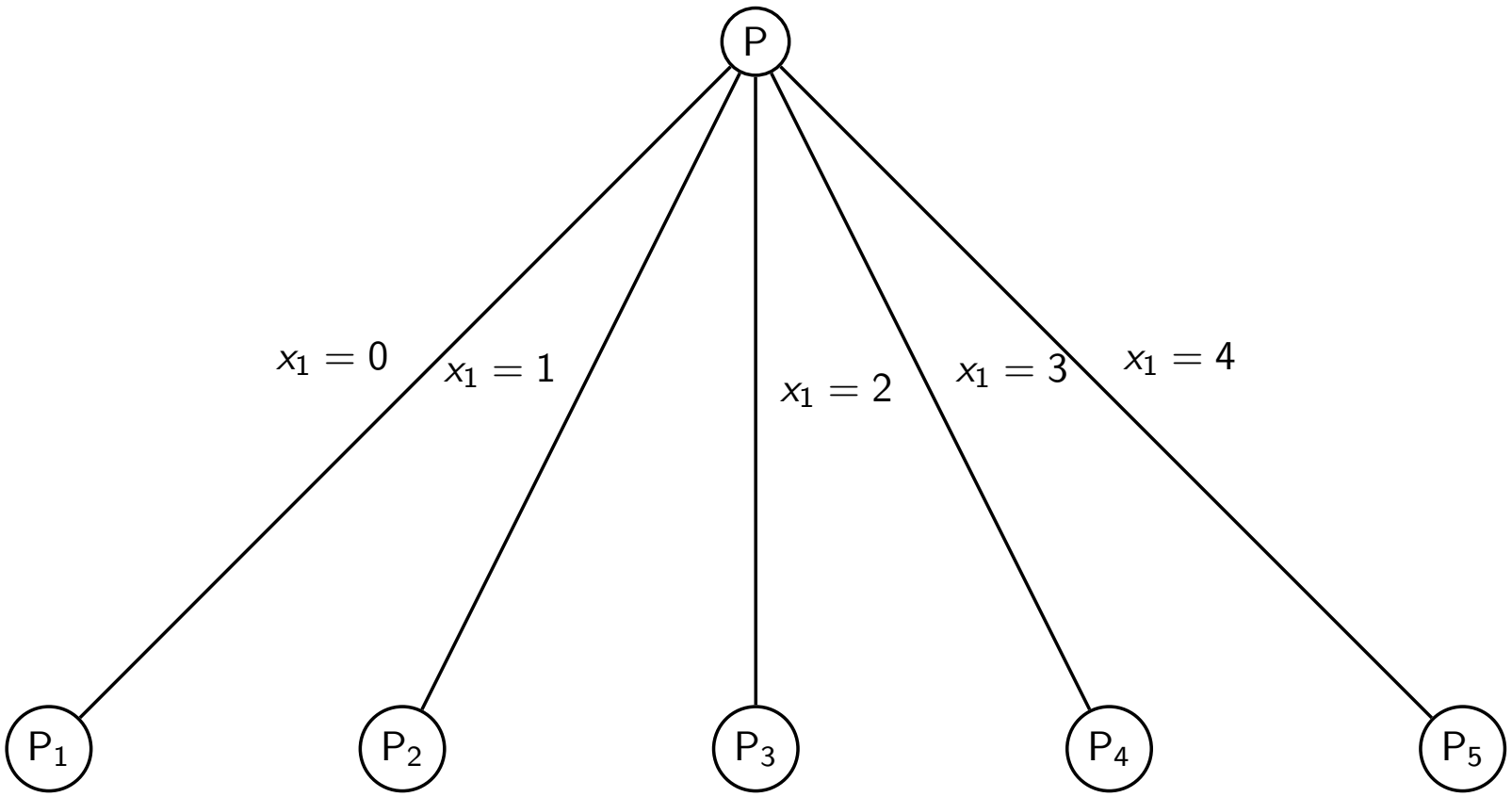


Branch non binario

Ogni regione si partiziona in k sotto-regioni, aggiungendo k vincoli.
Esempio: $x_1 = 0$, $x_1 = 1$, $x_1 = 2$, $x_1 = 3$ e $x_1 = 4$



Branch non binario



Bound

Una valutazione inferiore del valore ottimo di P è data dal valore della funzione obiettivo in una qualunque soluzione ammissibile.

Una valutazione superiore del valore ottimo di ogni sotto-problema è data dal valore ottimo di un suo *rilassamento*:

- Rilassamento continuo (eliminare i vincoli di interezza sulle variabili)

  $x \in \{0, 1\} \Rightarrow 0 \leq x \leq 1$

  $x \in \mathbb{Z}_+ \Rightarrow x \geq 0$
- Eliminazione di vincoli
- Somma di due o più vincoli
- ...

Criteri di potatura dell'albero decisionale

Un nodo P_i dell'albero decisionale può essere chiuso se vale una qualunque delle seguenti condizioni:

- il sotto-problema P_i non ha soluzioni ammissibili

Criteri di potatura dell'albero decisionale

Un nodo P_i dell'albero decisionale può essere chiuso se vale una qualunque delle seguenti condizioni:

- il sotto-problema P_i non ha soluzioni ammissibili
- $v_S(P_i) \leq v_I(P)$, cioè le soluzioni ammissibili di P_i non sono migliori della soluzione ammissibile corrente

Criteri di potatura dell'albero decisionale

Un nodo P_i dell'albero decisionale può essere chiuso se vale una qualunque delle seguenti condizioni:

- il sotto-problema P_i non ha soluzioni ammissibili
- $v_S(P_i) \leq v_I(P)$, cioè le soluzioni ammissibili di P_i non sono migliori della soluzione ammissibile corrente
- $v_S(P_i) > v_I(P)$ e la soluzione ottima di P_i è ammissibile per P. In tal caso si aggiorna $v_I(P) = v_S(P_i)$.

Visita dell'albero decisionale

In quale ordine visitiamo i nodi dell'albero?

Depth first (in profondità)

- Il prossimo nodo da visitare è uno dei figli del nodo attualmente visitato (se rimasto aperto)
- questa strategia trova rapidamente una soluzione ammissibile, occupa poca memoria, ma non tiene conto della qualità della soluzione trovata

Visita dell'albero decisionale

In quale ordine visitiamo i nodi dell'albero?

Depth first (in profondità)

- Il prossimo nodo da visitare è uno dei figli del nodo attualmente visitato (se rimasto aperto)
- questa strategia trova rapidamente una soluzione ammissibile, occupa poca memoria, ma non tiene conto della qualità della soluzione trovata

Best first

- Il prossimo nodo da visitare è il più promettente, cioè quello con il massimo valore di v_S
- trova rapidamente una soluzione ammissibile, occupa molto memoria, ma tiene conto della qualità della soluzione trovata

Visita dell'albero decisionale

In quale ordine visitiamo i nodi dell'albero?

Depth first (in profondità)

- Il prossimo nodo da visitare è uno dei figli del nodo attualmente visitato (se rimasto aperto)
- questa strategia trova rapidamente una soluzione ammissibile, occupa poca memoria, ma non tiene conto della qualità della soluzione trovata

Best first

- Il prossimo nodo da visitare è il più promettente, cioè quello con il massimo valore di v_S
- trova rapidamente una soluzione ammissibile, occupa molto memoria, ma tiene conto della qualità della soluzione trovata

Breadth first (in ampiezza)

- Si esplorano prima tutti i nodi dello stesso livello
- in generale non fornisce buone prestazioni dal punto di vista computazionale

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.
2. Se tutti i nodi sono stati visitati, allora STOP (la soluzione ammissibile corrente è ottima).

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.
2. Se tutti i nodi sono stati visitati, allora STOP (la soluzione ammissibile corrente è ottima).
3. Seleziona un nodo P_i da visitare.

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.
2. Se tutti i nodi sono stati visitati, allora STOP (la soluzione ammissibile corrente è ottima).
3. Seleziona un nodo P_i da visitare.
4. Se P_i non contiene soluzioni ammissibili, allora chiudi il nodo P_i e torna al passo 2.

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.
2. Se tutti i nodi sono stati visitati, allora STOP (la soluzione ammissibile corrente è ottima).
3. Seleziona un nodo P_i da visitare.
4. Se P_i non contiene soluzioni ammissibili, allora chiudi il nodo P_i e torna al passo 2.
5. (Bound) risolvi un rilassamento di P_i e calcola $v_S(P_i)$.

Metodo Branch and bound

1. Genera il nodo radice P (da visitare), trova una soluzione ammissibile per P e calcola una $v_I(P)$.
2. Se tutti i nodi sono stati visitati, allora STOP (la soluzione ammissibile corrente è ottima).
3. Seleziona un nodo P_i da visitare.
4. Se P_i non contiene soluzioni ammissibili, allora chiudi il nodo P_i e torna al passo 2.
5. (Bound) risolvi un rilassamento di P_i e calcola $v_S(P_i)$.
   - se $v_S(P_i) \leq v_I(P)$, allora chiudi il nodo P_i e torna al passo 2.
   - se $v_S(P_i) > v_I(P)$ e la soluzione ottima del rilassamento di P_i è ammissibile per P, allora chiudi il nodo P_i, aggiorna la soluzione ammissibile, poni $v_I(P) = v_S(P_i)$ e torna al passo 2.
6. (Branch) Partiziona la regione ammissibile di P_i in sotto-regioni e genera nuovi nodi da visitare. Torna al passo 2.

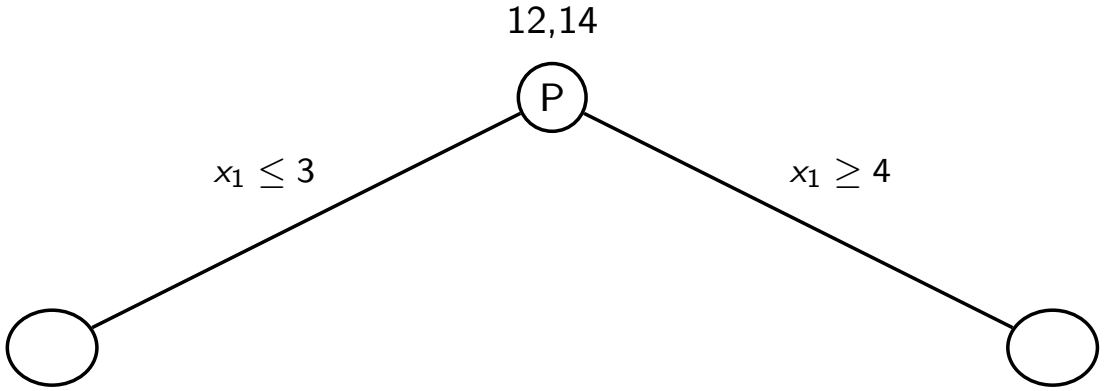
Esempio

Applichiamo il metodo Branch and Bound per risolvere il problema di PLI

$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0, \ x \in \mathbb{Z}^2 \end{cases} \qquad (P)$$

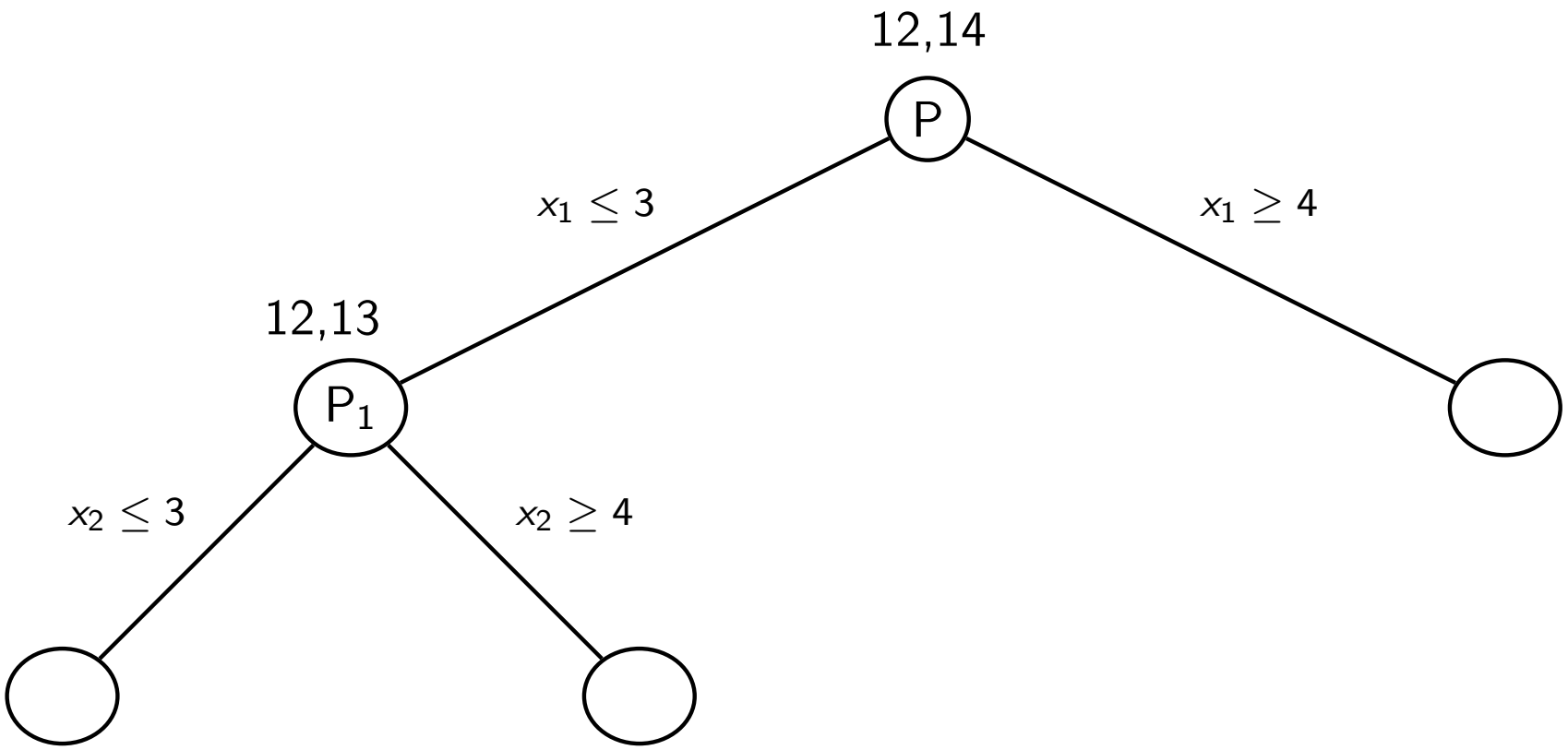
In ogni nodo dell'albero decisionale risolviamo il rilassamento continuo, usiamo un *branch* binario basato sulla prima componente non intera della soluzione ottima del rilassamento. Visitiamo l'albero decisionale in profondità.

Sappiamo che $(3, 3)$ è una soluzione ammissibile, quindi $v_I(P) = 12$.
Sappiamo che la soluzione ottima del rilassamento continuo è $(7/2, 7/2)$, quindi $v_S(P) = 14$. Il nodo P rimane aperto. Branch: $x_1 \leq \lfloor 7/2 \rfloor = 3$ oppure $x_1 \geq \lceil 7/2 \rceil = 4$.



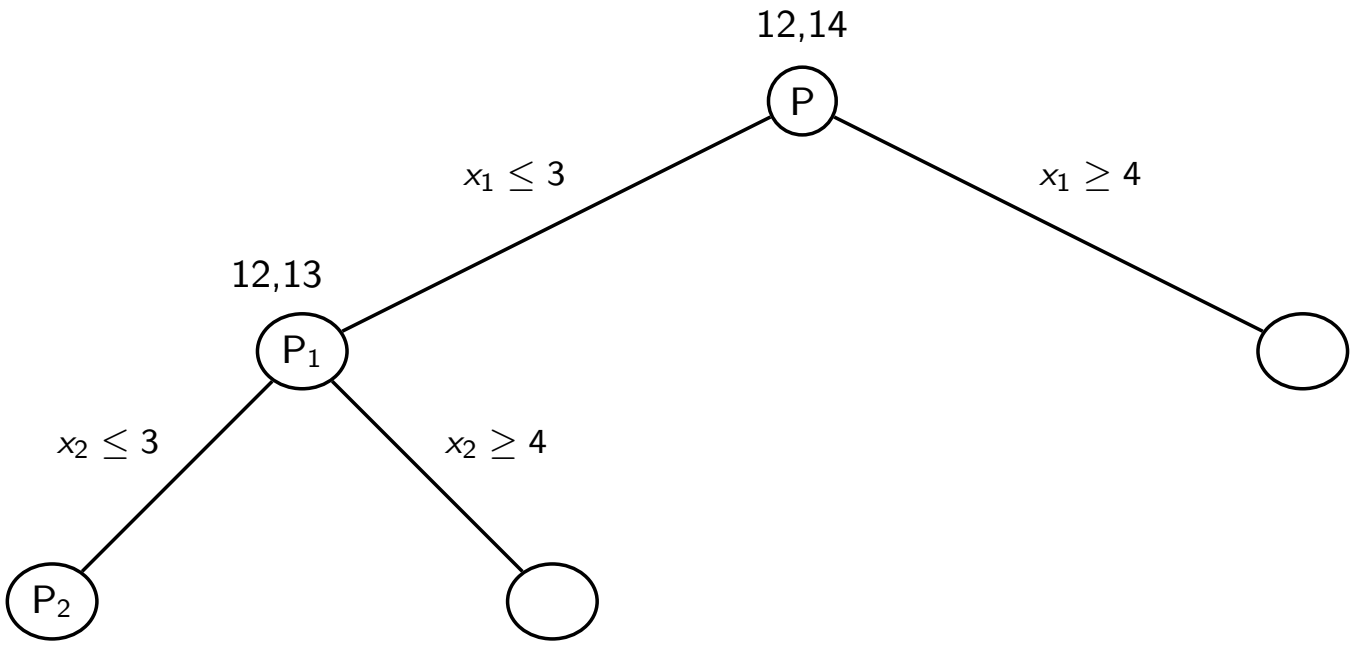
Esempio

La soluzione ottima del rilassamento continuo di P_1 è $(3, 18/5)$, non è ammissibile ed ha valore 13.8, quindi $v_S(P_1) = 13 > 12 = v_I(P)$. Pertanto il nodo P_1 rimane aperto.
Dal nodo P_1 facciamo *branch* binario sulla variabile x_2: $x_2 \leq 3$ oppure $x_2 \geq 4$.



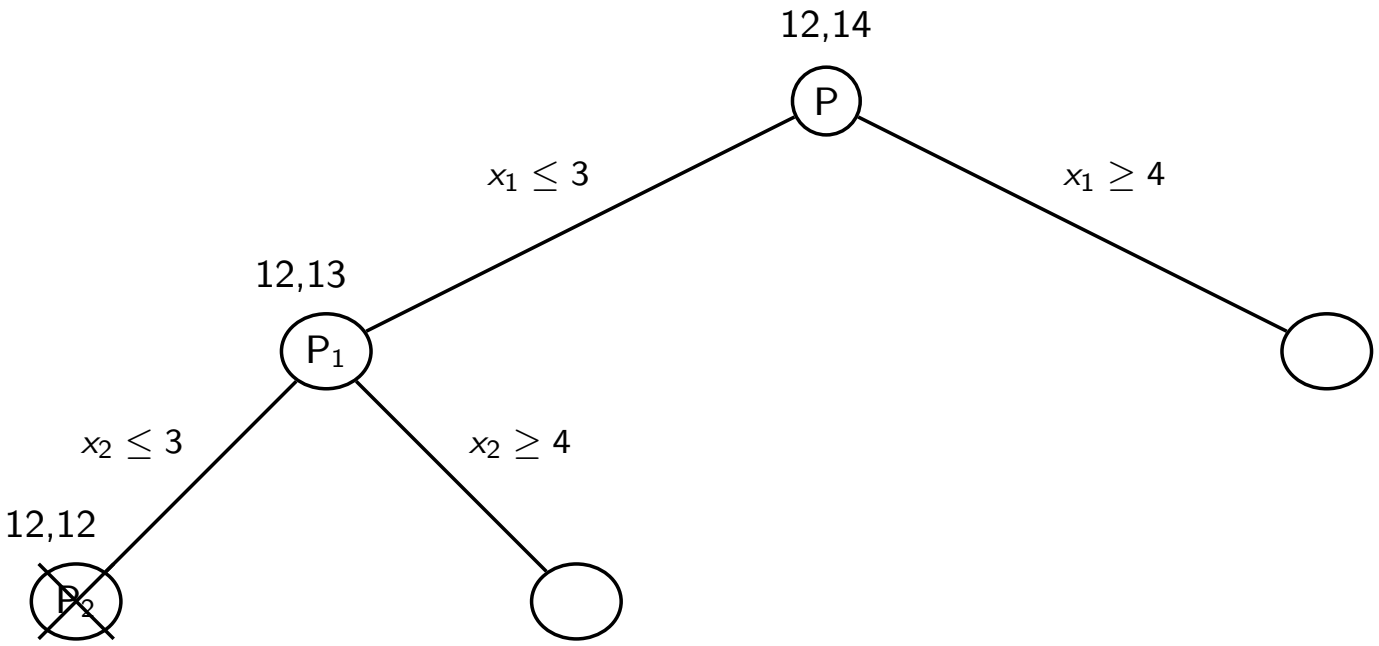
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.



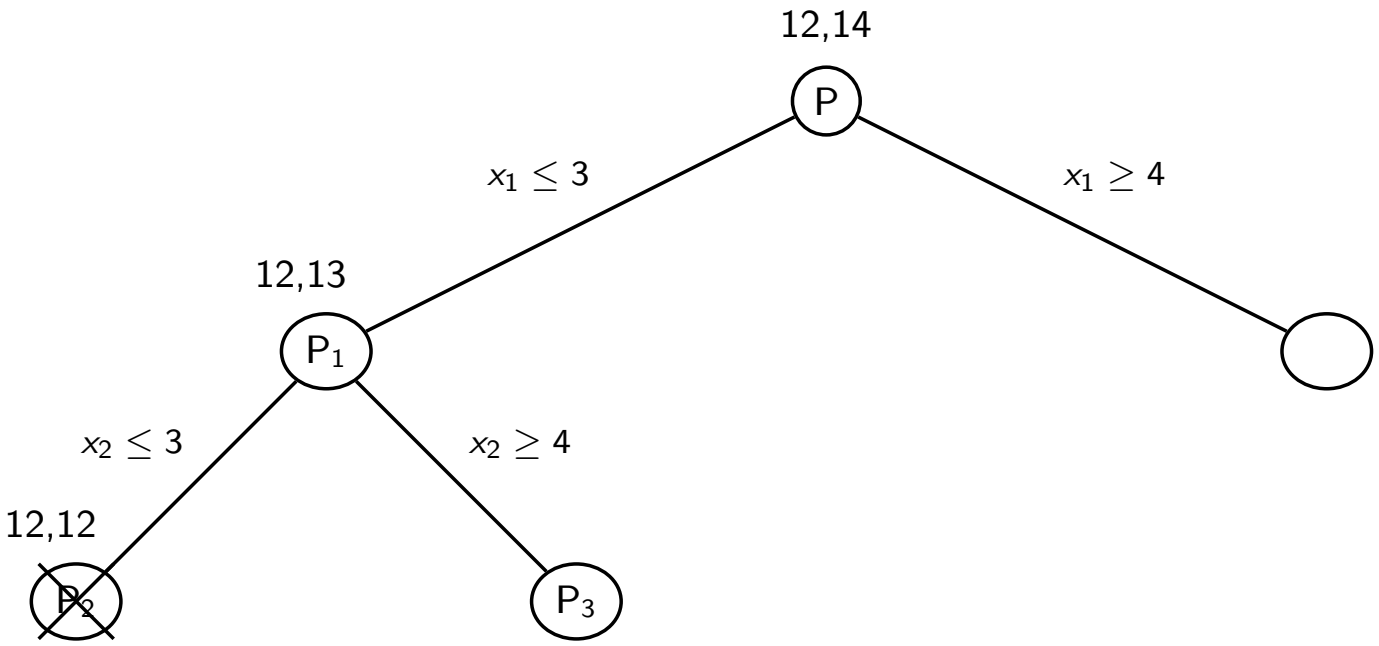
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.



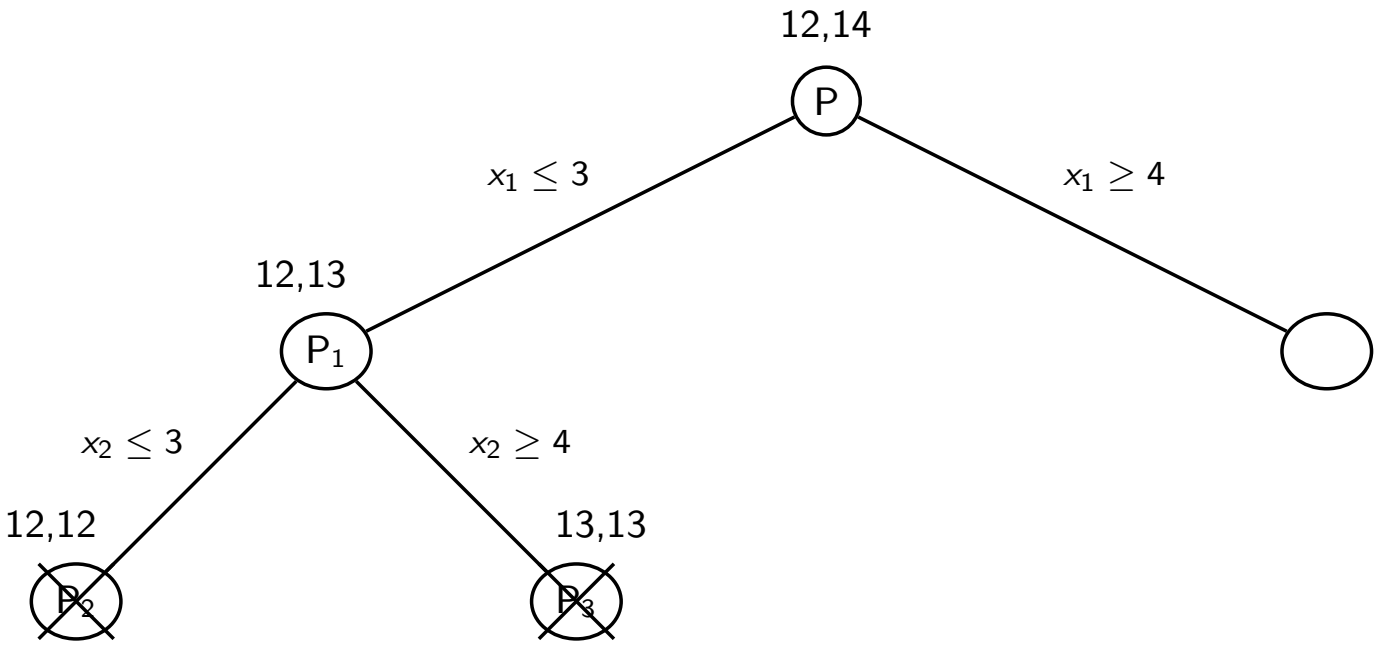
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.
Visitiamo ora l'altro nodo figlio di P_1. La soluzione ottima del rilassamento continuo di P_3 è $(1, 4)$ e $v_S(P_3) = 13 > 12 = v_I(P)$. Poiché $(1, 4)$ è ammissibile per P, aggiorniamo $v_I(P) = 13$ e chiudiamo il nodo P_3.



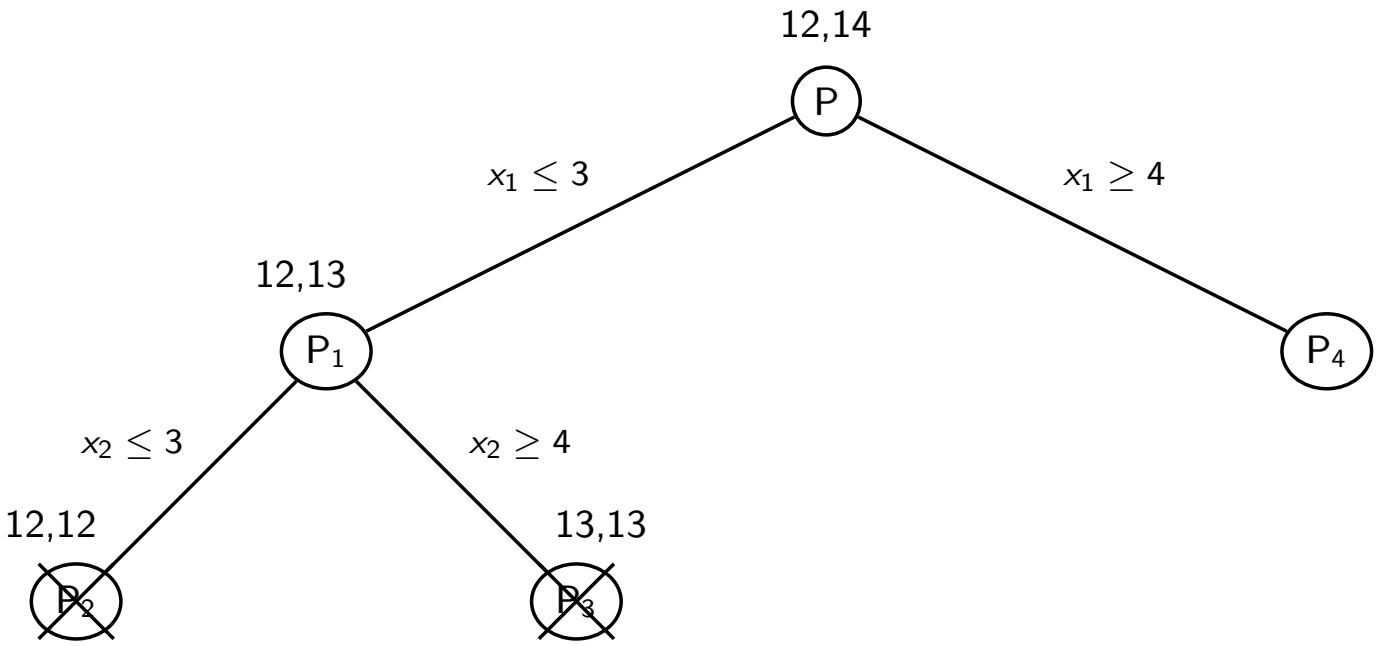
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.
Visitiamo ora l'altro nodo figlio di P_1. La soluzione ottima del rilassamento continuo di P_3 è $(1, 4)$ e $v_S(P_3) = 13 > 12 = v_I(P)$. Poiché $(1, 4)$ è ammissibile per P, aggiorniamo $v_I(P) = 13$ e chiudiamo il nodo P_3.



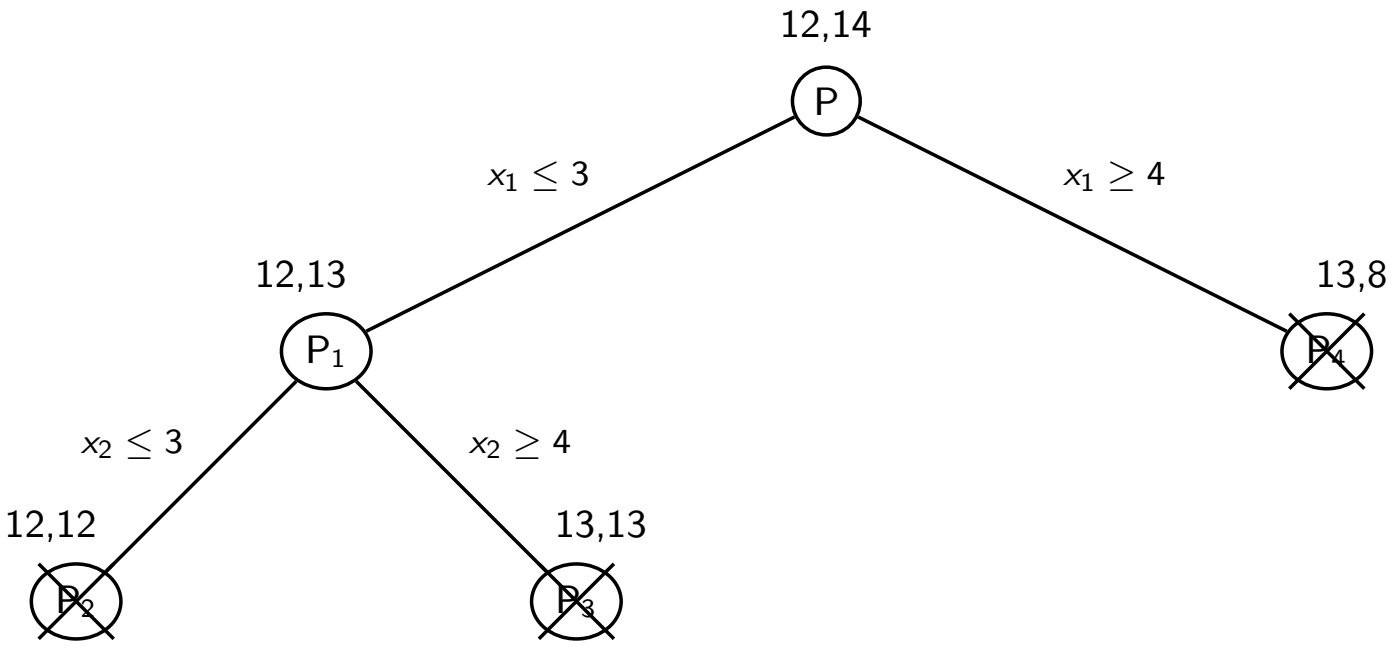
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.
Visitiamo ora l'altro nodo figlio di P_1. La soluzione ottima del rilassamento continuo di P_3 è $(1, 4)$ e $v_S(P_3) = 13 > 12 = v_I(P)$. Poiché $(1, 4)$ è ammissibile per P, aggiorniamo $v_I(P) = 13$ e chiudiamo il nodo P_3.
È rimasto da visitare l'altro figlio del nodo P. La soluzione ottima del rilassamento continuo di P_4 è $(4, 3/2)$ con valore 8.5, quindi $v_S(P_4) = 8 < 13 = v_I(P)$. Pertanto, chiudiamo il nodo P_4.



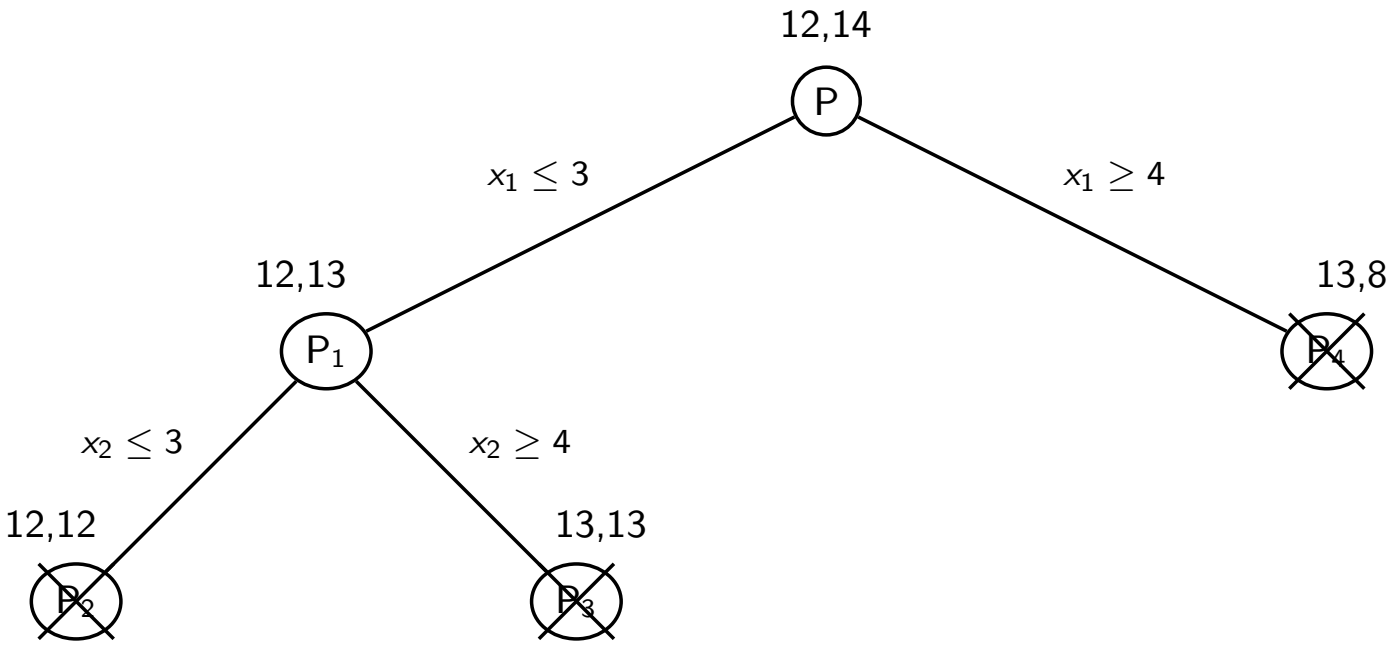
Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.
Visitiamo ora l'altro nodo figlio di P_1. La soluzione ottima del rilassamento continuo di P_3 è $(1, 4)$ e $v_S(P_3) = 13 > 12 = v_I(P)$. Poiché $(1, 4)$ è ammissibile per P, aggiorniamo $v_I(P) = 13$ e chiudiamo il nodo P_3.
È rimasto da visitare l'altro figlio del nodo P. La soluzione ottima del rilassamento continuo di P_4 è $(4, 3/2)$ con valore 8.5, quindi $v_S(P_4) = 8 < 13 = v_I(P)$. Pertanto, chiudiamo il nodo P_4.



Esempio

La soluzione ottima del rilassamento continuo di P_2 è $(3, 3)$, quindi $v_S(P_2) = 12 = v_I(P)$, quindi chiudiamo il nodo P_2.
Visitiamo ora l'altro nodo figlio di P_1. La soluzione ottima del rilassamento continuo di P_3 è $(1, 4)$ e $v_S(P_3) = 13 > 12 = v_I(P)$. Poiché $(1, 4)$ è ammissibile per P, aggiorniamo $v_I(P) = 13$ e chiudiamo il nodo P_3.
È rimasto da visitare l'altro figlio del nodo P. La soluzione ottima del rilassamento continuo di P_4 è $(4, 3/2)$ con valore 8.5, quindi $v_S(P_4) = 8 < 13 = v_I(P)$. Pertanto, chiudiamo il nodo P_4.



Poiché tutti i nodi dell'albero sono stati visitati, la soluzione ottima di P è $(1, 4)$ ed il valore ottimo è 13.

Problema dello zaino

Dati n oggetti di valore $v_1, \ldots, v_n$ e peso $p_1, \ldots, p_n$, ed un contenitore di capacità C, quali oggetti inserisco nel contenitore, rispettando la sua capacità, in modo da massimizzare il valore totale degli oggetti inseriti?

Variabili: $$x_i = \begin{cases} 1 & \text{se oggetto } i \text{ viene inserito,} \\ 0 & \text{altrimenti.} \end{cases}$$

Modello di PLI:

$$\begin{cases} \max \displaystyle\sum_{i=1}^{n} v_i\, x_i \\ \displaystyle\sum_{i=1}^{n} p_i\, x_i \leq C \\ x_i \in \{0,1\}, \qquad i = 1, \ldots, n \end{cases}$$

Problema dello zaino: rilassamento continuo

Il rilassamento continuo è

$$\begin{cases} \max \sum_{i=1}^{n} v_i\, x_i \\ \sum_{i=1}^{n} p_i\, x_i \leq C \\ 0 \leq x_i \leq 1 \qquad i = 1, \ldots, n \end{cases}$$

Le condizioni degli scarti complementari garantiscono che la soluzione ottima $\bar{x}$ del rilassamento continuo si può calcolare nel modo seguente:

1. Ordina gli oggetti in ordine decrescente di valore per unità di peso (rendimento) $\frac{v_i}{p_i}$
2. Trova l'indice h tale che $\sum_{i=1}^{h} p_i \leq C$ e $\sum_{i=1}^{h+1} p_i > C$,
3. Poni $\bar{x}_1 = 1, \ldots, \bar{x}_h = 1,\ \bar{x}_{h+1} = \dfrac{C - \sum_{i=1}^{h} p_i}{p_{h+1}},\ \bar{x}_{h+2} = 0, \ldots, \bar{x}_n = 0.$

Problema dello zaino: rilassamento continuo

Esempio.

$$\begin{cases} \max\ 11\,x_1 + 23\,x_2 + 18\,x_3 + 6\,x_4 \\ 7\,x_1 + 6\,x_2 + 3\,x_3 + 2\,x_4 \leq 8 \\ x_i \in \{0,1\} \qquad i = 1,\ldots,4 \end{cases}$$

Ordiniamo gli oggetti in ordine decrescente di rendimento:

i	3	2	4	1
v_i/p_i	6	3.8	3	1.6

La soluzione ottima del rilassamento continuo è $\bar{x}_3 = 1$, $\bar{x}_2 = 5/6$, $\bar{x}_4 = \bar{x}_1 = 0$, cioè $\bar{x} = \left(0, \frac{5}{6}, 1, 0\right)$ di valore 37.2, quindi $v_S(P) = 37$.

Problema dello zaino: algoritmo euristico

Una soluzione ammissibile può essere ottenuta utilizzando il seguente algoritmo euristico:

Ordina gli oggetti in ordine decrescente di rendimento $\frac{v_i}{p_i}$

Poni $\bar{C} = C$ ($\bar{C}$ rappresenta la capacità residua del contenitore)

for $i = 1, \ldots, n$ **do**
  if $p_i \leq \bar{C}$
    then $x_i = 1$, $\bar{C} = \bar{C} - p_i$
  else $x_i = 0$
end

Problema dello zaino: algoritmo euristico

Una soluzione ammissibile può essere ottenuta utilizzando il seguente algoritmo euristico:

Ordina gli oggetti in ordine decrescente di rendimento $\dfrac{v_i}{p_i}$

Poni $\bar{C} = C$ ($\bar{C}$ rappresenta la capacità residua del contenitore)

for $i = 1, \ldots, n$ **do**
  if $p_i \leq \bar{C}$
    then $x_i = 1,\ \bar{C} = \bar{C} - p_i$
  else $x_i = 0$
end

Esempio (continua).

$$\begin{cases} \max\ 11\,x_1 + 23\,x_2 + 18\,x_3 + 6\,x_4 \\ 7\,x_1 + 6\,x_2 + 3\,x_3 + 2\,x_4 \leq 8 \\ x_i \in \{0,1\} \qquad i = 1, \ldots, 4 \end{cases}$$

L'algoritmo euristico fornisce la soluzione ammissibile $x_3 = 1$, $x_2 = 0$, $x_4 = 1$, $x_1 = 0$, quindi $x = (0, 0, 1, 1)$ e $v_I(P) = 24$.

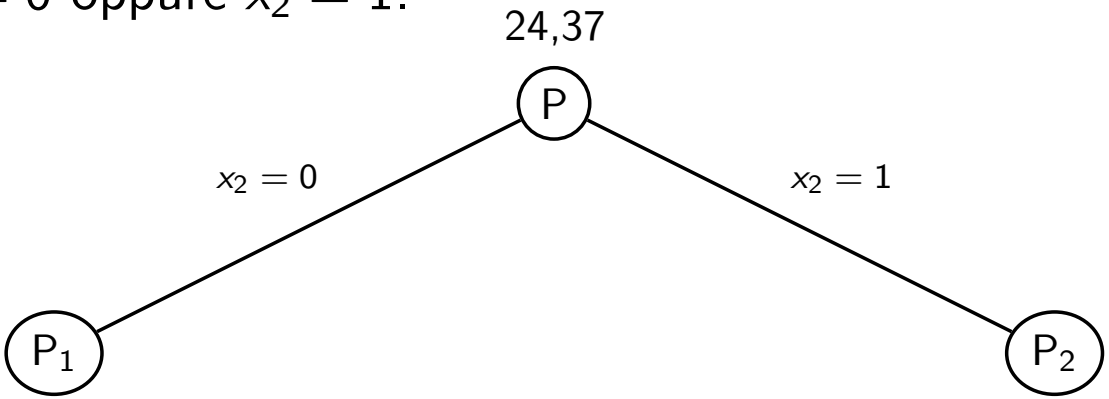
Problema dello zaino: metodo Branch and Bound

Utilizziamo il metodo Branch and Bound per trovare una soluzione ottima del problema

$$\begin{cases} \max\ 11\,x_1 + 23\,x_2 + 18\,x_3 + 6\,x_4 \\ 7\,x_1 + 6\,x_2 + 3\,x_3 + 2\,x_4 \leq 8 \\ x_i \in \{0,1\} \qquad i = 1,\ldots,4 \end{cases}$$

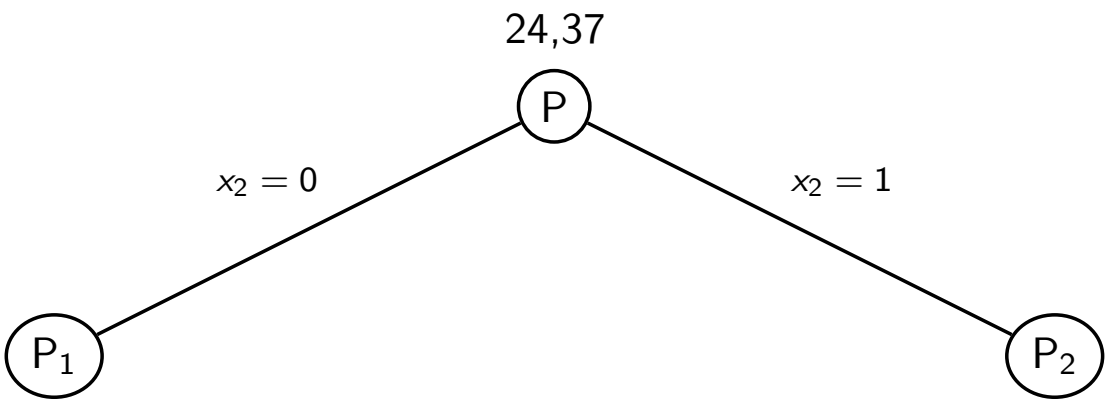
Bound: rilassamento continuo.
Branch binario: istanziamo l'eventuale variabile frazionaria della soluzione ottima del rilassamento continuo.
Visita dell'albero decisionale: in ampiezza.

La soluzione ottima del rilassamento continuo di P è $\left(0, \frac{5}{6}, 1, 0\right)$, istanziamo la variabile x_2: $x_2 = 0$ oppure $x_2 = 1$.



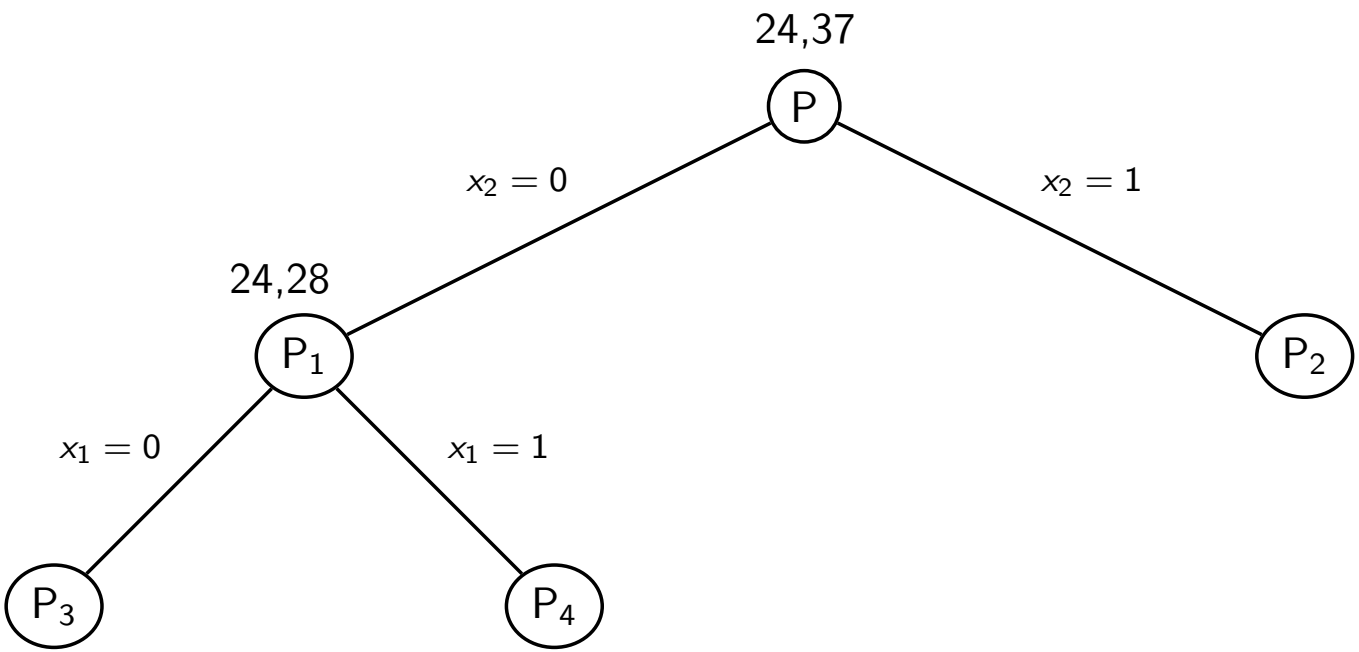
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_1 è $(3/7, 0, 1, 1)$ di valore $28 = v_S(P_1) > 24 = v_I(P)$, quindi P_1 rimane aperto.



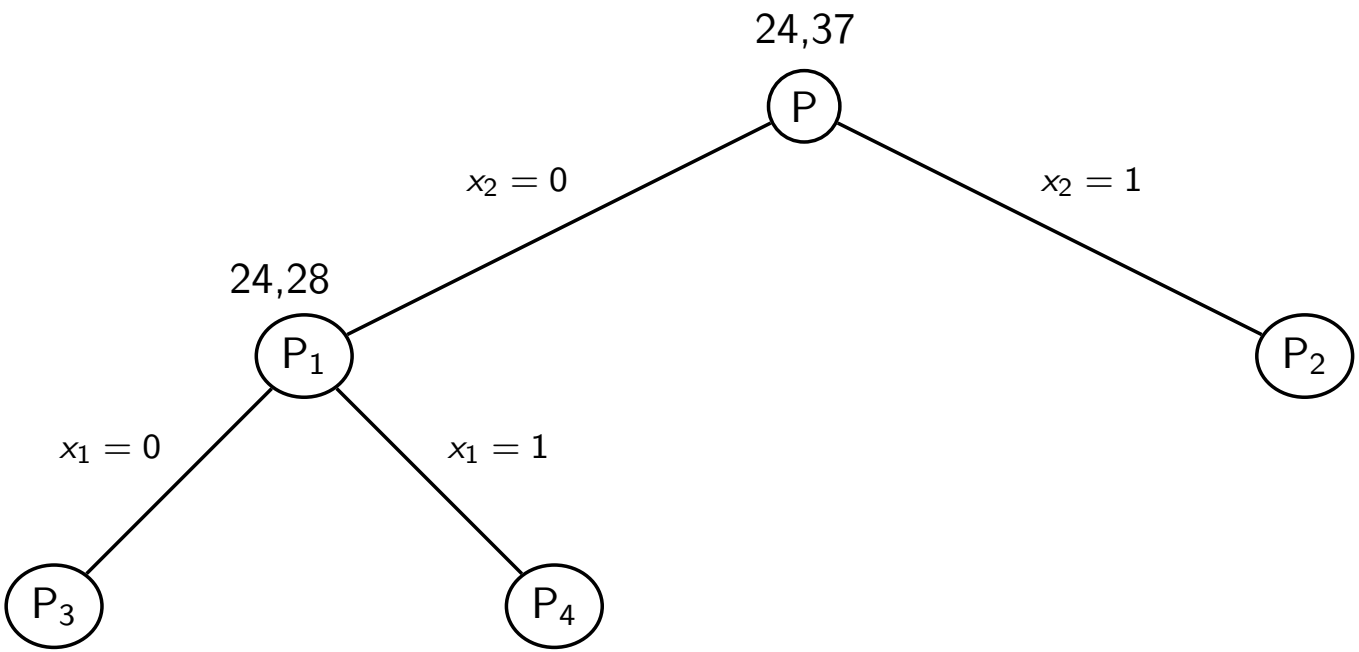
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_1 è $(3/7, 0, 1, 1)$ di valore $28 = v_S(P_1) > 24 = v_I(P)$, quindi P_1 rimane aperto. Generiamo due nodi figli P_3 e P_4 istanziando la variabile x_1.



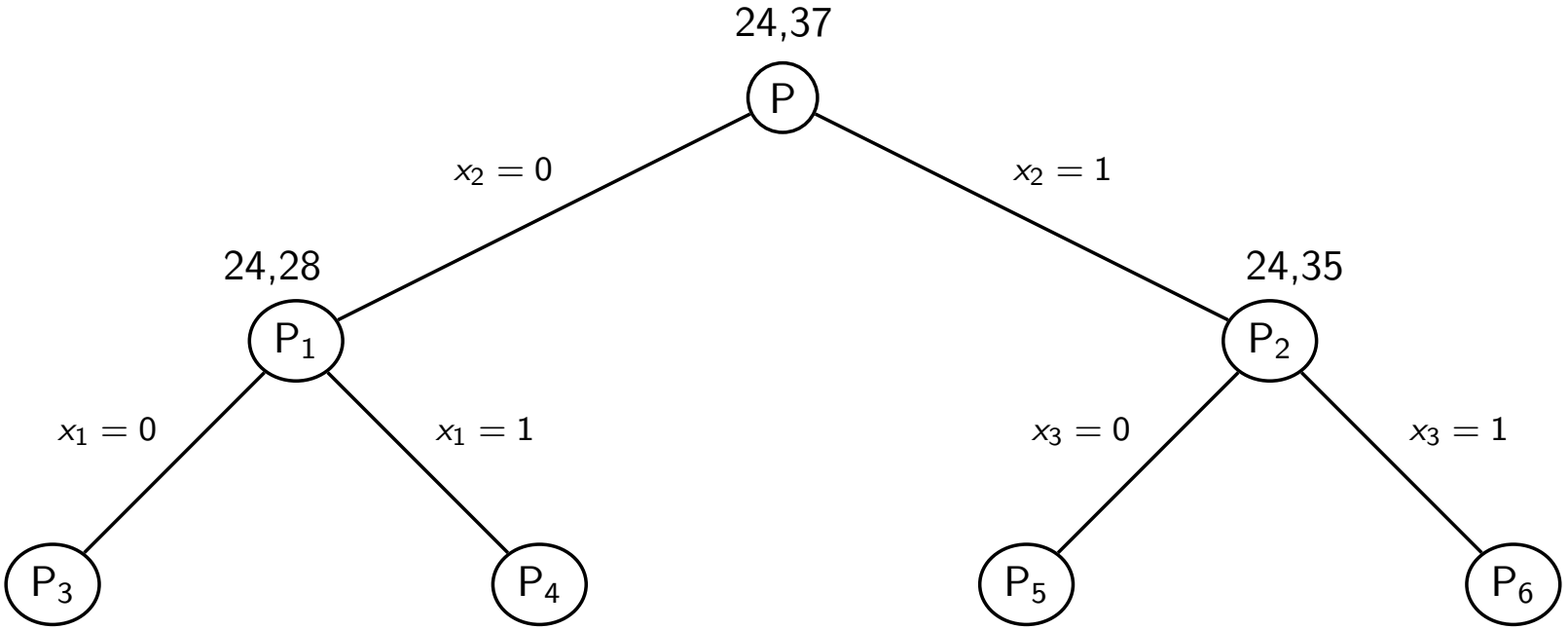
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_1 è $(3/7, 0, 1, 1)$ di valore $28 = v_S(P_1) > 24 = v_I(P)$, quindi P_1 rimane aperto. Generiamo due nodi figli P_3 e P_4 istanziando la variabile x_1.
La soluzione ottima del rilassamento continuo di P_2 è $(0, 1, 2/3, 0)$ di valore $35 = v_S(P_2) > 24 = v_I(P)$, quindi P_2 rimane aperto.



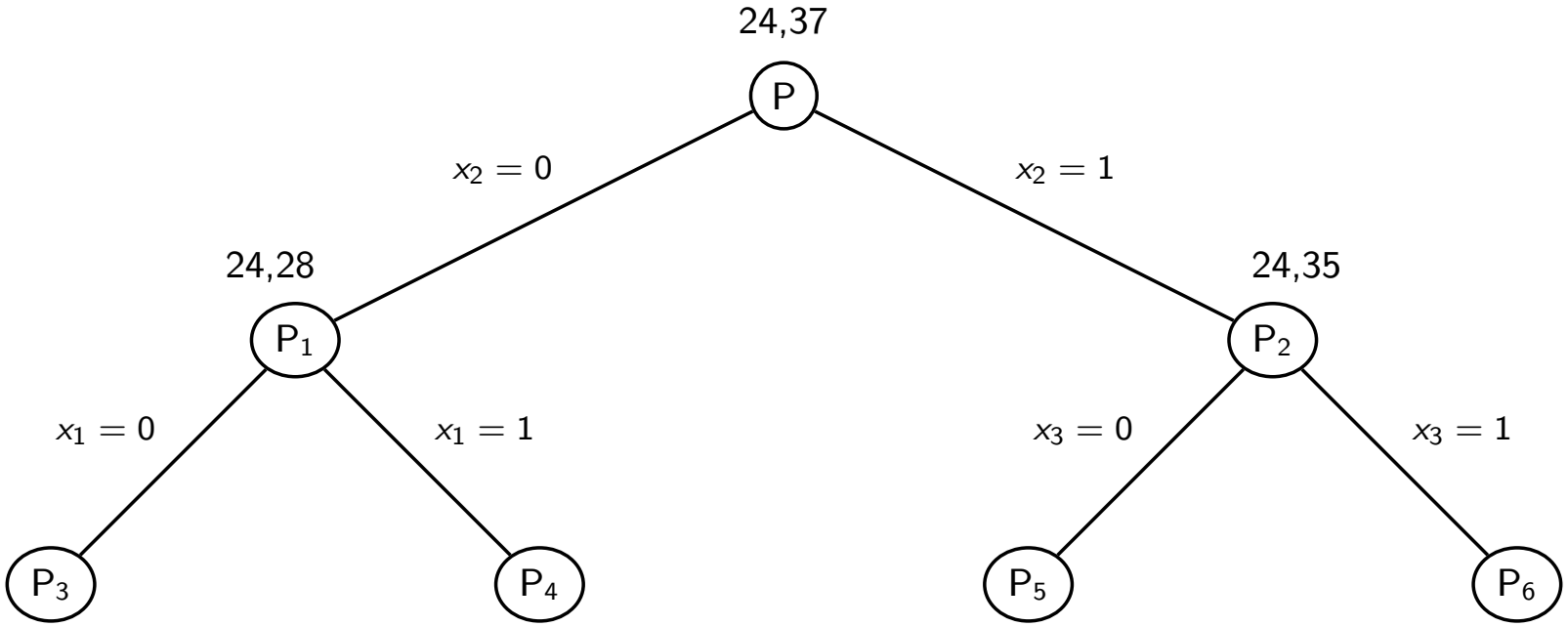
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_1 è $(3/7, 0, 1, 1)$ di valore $28 = v_S(P_1) > 24 = v_I(P)$, quindi P_1 rimane aperto. Generiamo due nodi figli P_3 e P_4 istanziando la variabile x_1.
La soluzione ottima del rilassamento continuo di P_2 è $(0, 1, 2/3, 0)$ di valore $35 = v_S(P_2) > 24 = v_I(P)$, quindi P_2 rimane aperto. Generiamo due nodi figli P_5 e P_6 istanziando la variabile x_3.



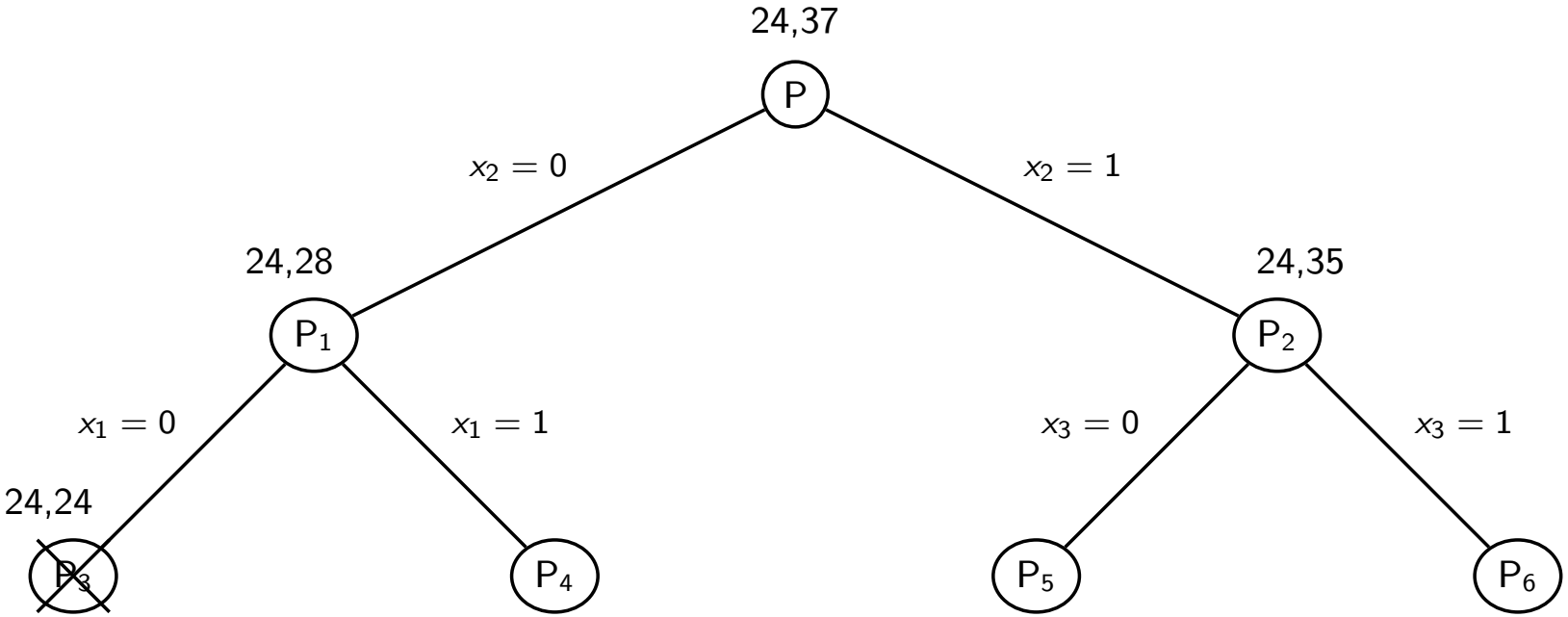
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.



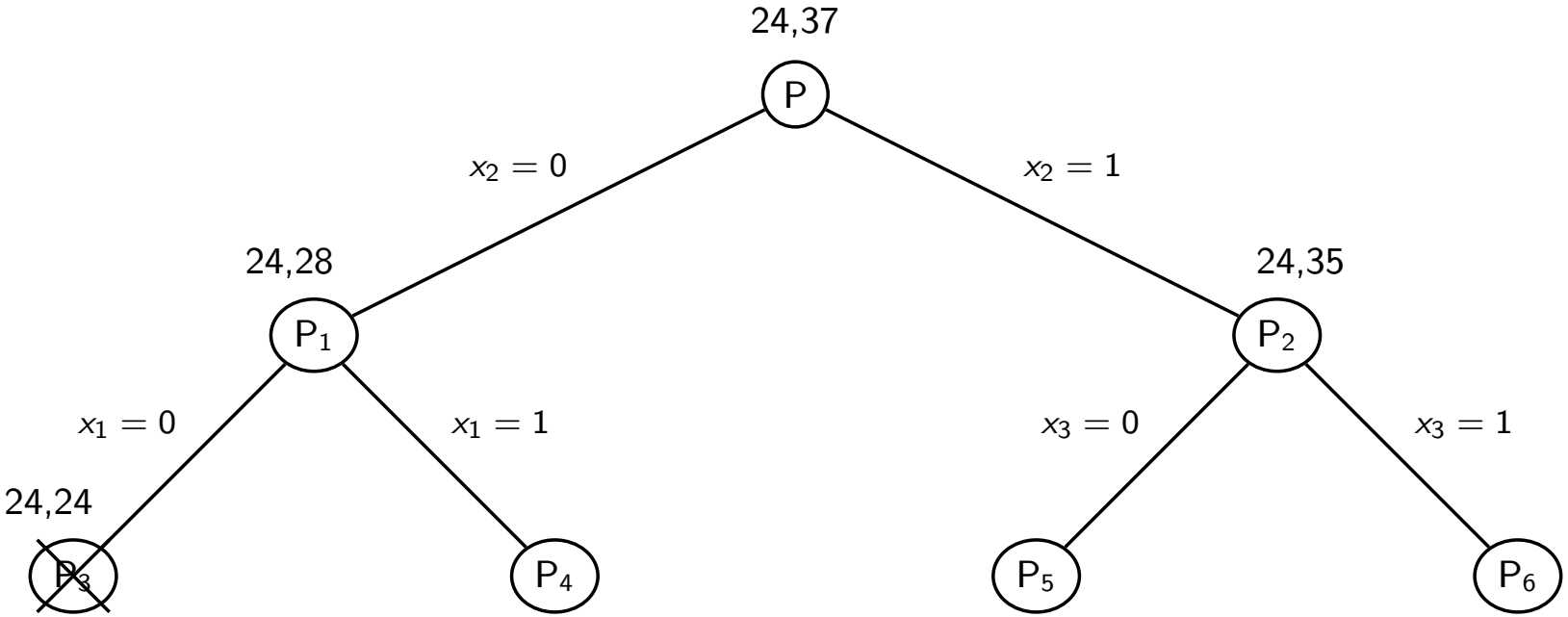
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.



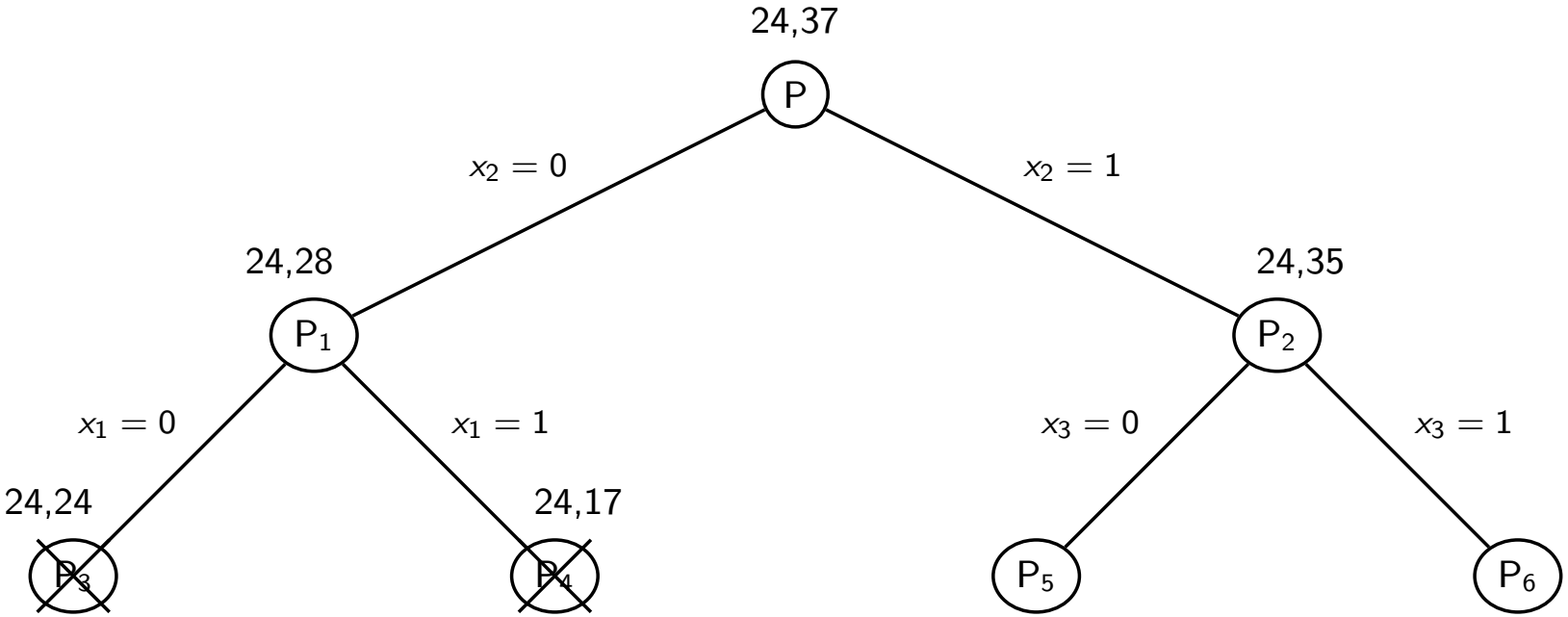
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.



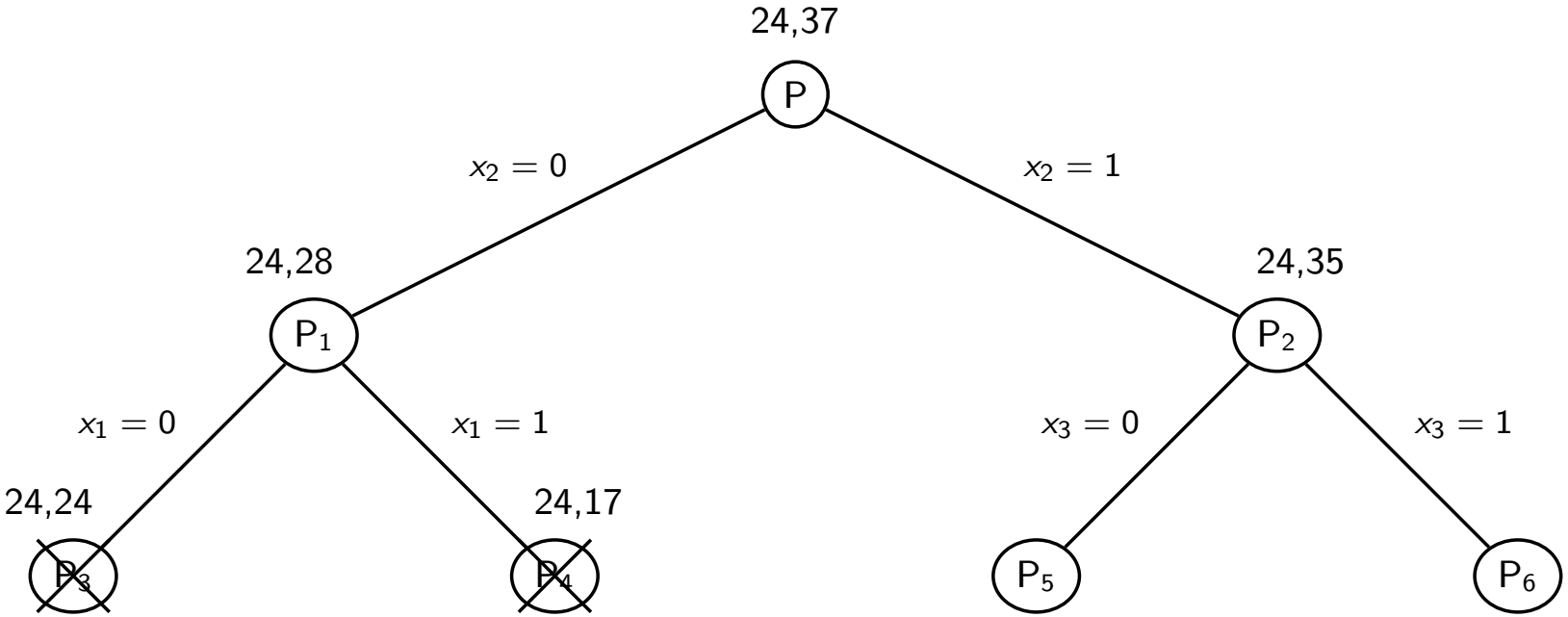
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.



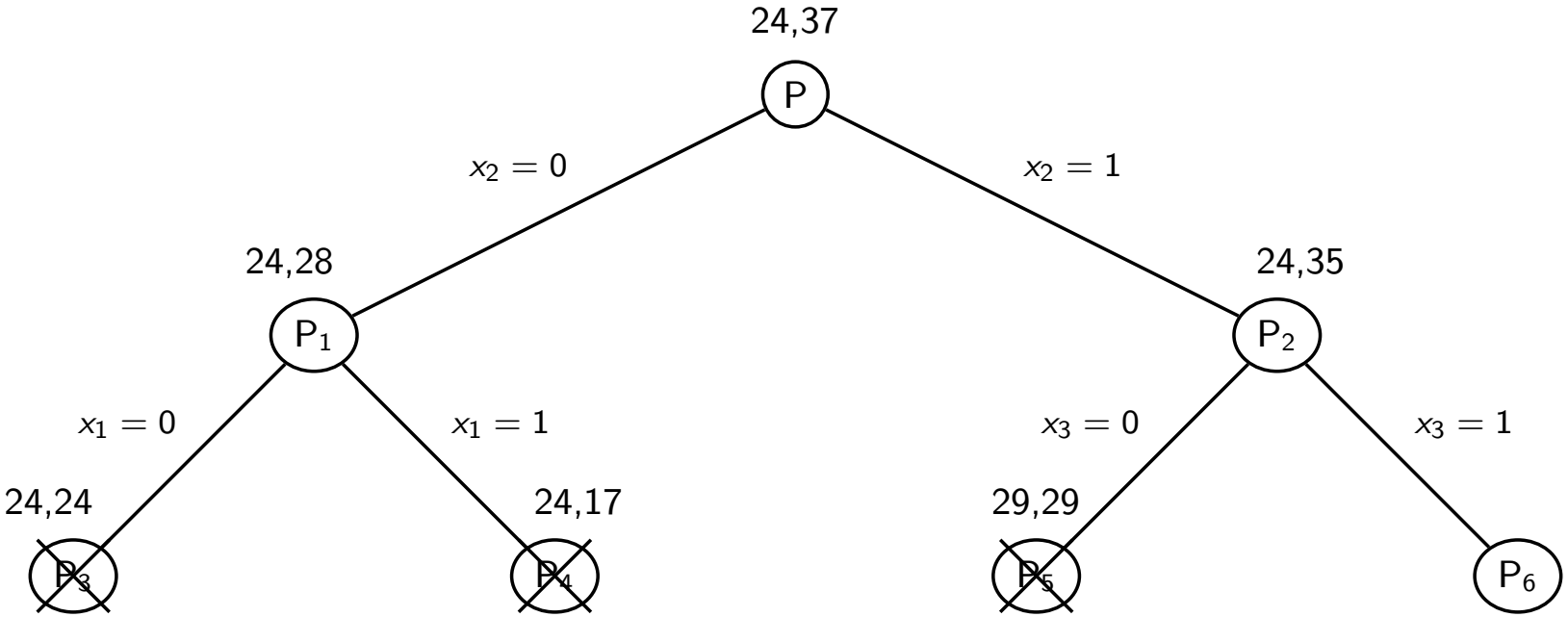
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.
La soluzione ottima del rilassamento continuo di P_5 è $(0, 1, 0, 1)$ di valore $29 = v_S(P_5) > 24 = v_I(P)$. Poiché tale soluzione è ammissibile per P, aggiorniamo $v_I(P) = 29$ e chiudiamo P_5.



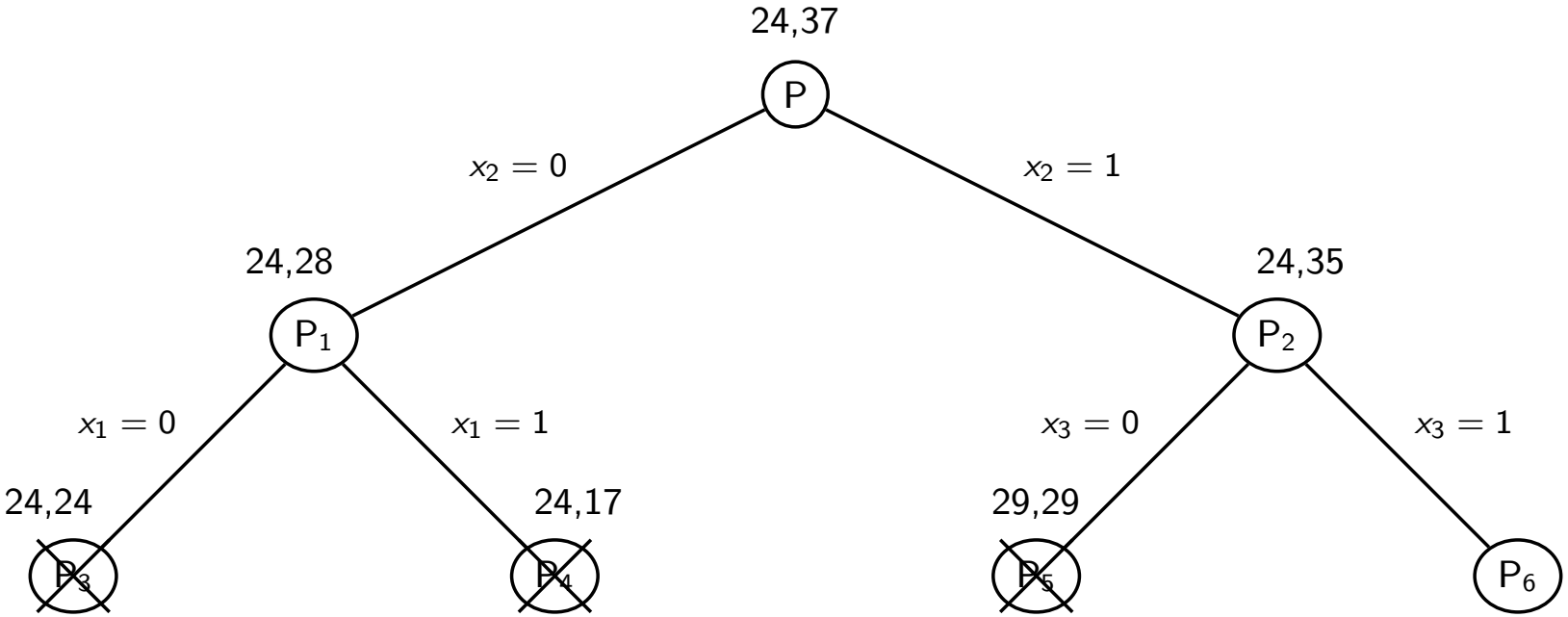
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.
La soluzione ottima del rilassamento continuo di P_5 è $(0, 1, 0, 1)$ di valore $29 = v_S(P_5) > 24 = v_I(P)$. Poiché tale soluzione è ammissibile per P, aggiorniamo $v_I(P) = 29$ e chiudiamo P_5.



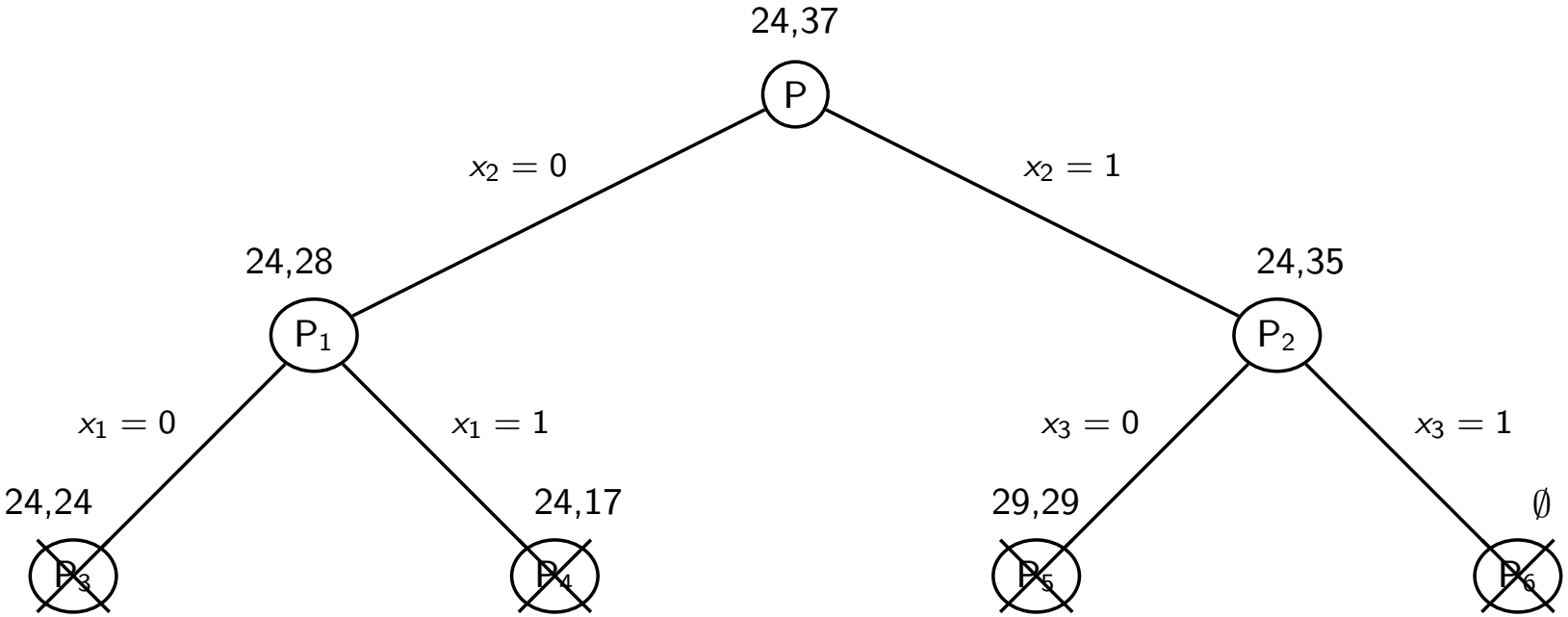
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.
La soluzione ottima del rilassamento continuo di P_5 è $(0, 1, 0, 1)$ di valore $29 = v_S(P_5) > 24 = v_I(P)$. Poiché tale soluzione è ammissibile per P, aggiorniamo $v_I(P) = 29$ e chiudiamo P_5.
P_6 non ammette soluzioni ammissibili, quindi chiudiamo P_6.



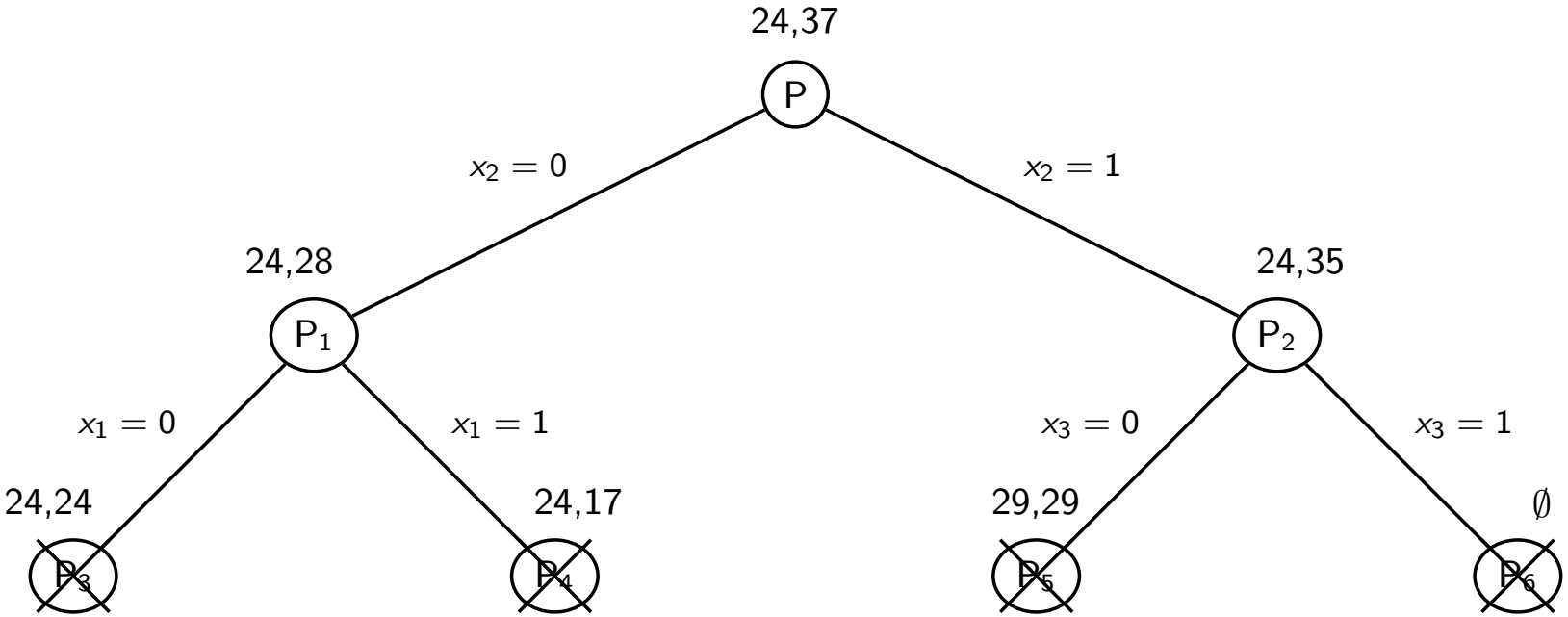
Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.
La soluzione ottima del rilassamento continuo di P_5 è $(0, 1, 0, 1)$ di valore $29 = v_S(P_5) > 24 = v_I(P)$. Poiché tale soluzione è ammissibile per P, aggiorniamo $v_I(P) = 29$ e chiudiamo P_5.
P_6 non ammette soluzioni ammissibili, quindi chiudiamo P_6.



Problema dello zaino: metodo Branch and Bound

La soluzione ottima del rilassamento continuo di P_3 è $(0, 0, 1, 1)$ di valore $24 = v_S(P_3) = v_I(P)$, quindi chiudiamo P_3.
La soluzione ottima del rilassamento continuo di P_4 è $(1, 0, 1/3, 0)$ di valore $17 = v_S(P_4) < 24 = v_I(P)$, quindi chiudiamo P_4.
La soluzione ottima del rilassamento continuo di P_5 è $(0, 1, 0, 1)$ di valore $29 = v_S(P_5) > 24 = v_I(P)$. Poiché tale soluzione è ammissibile per P, aggiorniamo $v_I(P) = 29$ e chiudiamo P_5.
P_6 non ammette soluzioni ammissibili, quindi chiudiamo P_6.



Poiché tutti i nodi dell'albero sono stati visitati, la soluzione ottima di P è $(0, 1, 0, 1)$ ed il valore ottimo è 29.

Problema del commesso viaggiatore (Traveling Salesman Problem)

Problema

Dato un grafo (N, A) completo, in cui c_{ij} è il costo dell'arco (i, j), trovare un ciclo che passi su tutti i nodi una ed una sola volta (ciclo hamiltoniano) di costo totale minimo.

Se la matrice dei costi è simmetrica, cioè $c_{ij} = c_{ji}$ per ogni arco (i, j), il problema è detto simmetrico; altrimenti è detto asimmetrico.
Nel seguito studieremo solo il caso simmetrico, quindi supponiamo che il grafo (N, A) sia non orientato.

Teorema

Questo problema è NP-hard.

Esercizio. Quanti cicli hamiltoniani esistono in un grafo completo con n nodi?

Problema del commesso viaggiatore (TSP)

Applicazioni

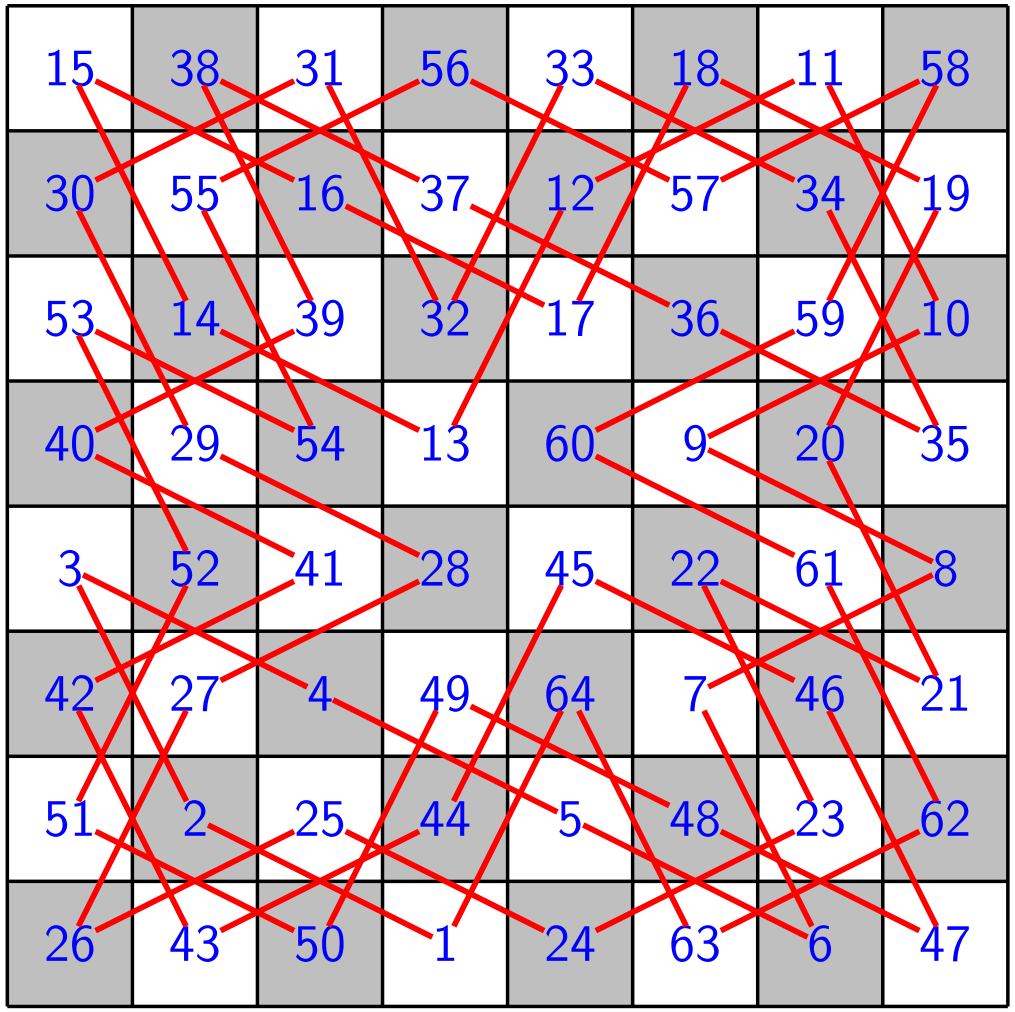
- trasporti, logistica: (N', A') rete stradale. $S \subseteq N'$, cerco ciclo di costo minimo che passi su tutti i nodi di S. Il problema è un TSP sul grafo (N, A), dove $N = S$, $A = S \times S$, $c_{ij} =$ costo cammino minimo da i a j sul grafo (N', A').

Problema del commesso viaggiatore (TSP)

Applicazioni

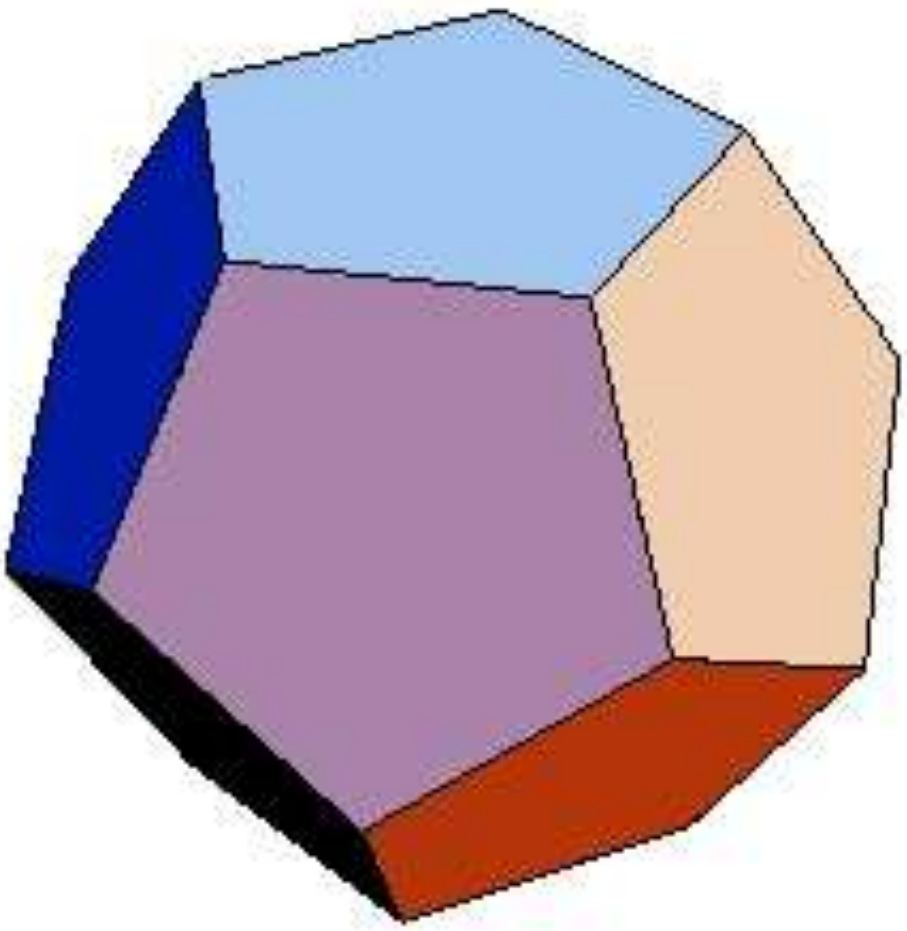
- trasporti, logistica: (N', A') rete stradale. $S \subseteq N'$, cerco ciclo di costo minimo che passi su tutti i nodi di S. Il problema è un TSP sul grafo (N, A), dove $N = S$, $A = S \times S$, c_{ij} = costo cammino minimo da i a j sul grafo (N', A').
- scheduling
- produzione di circuiti integrati
- data analysis
- sequenze DNA
- ...(vedi http://www.tsp.gatech.edu/)

Problema del commesso viaggiatore (TSP)



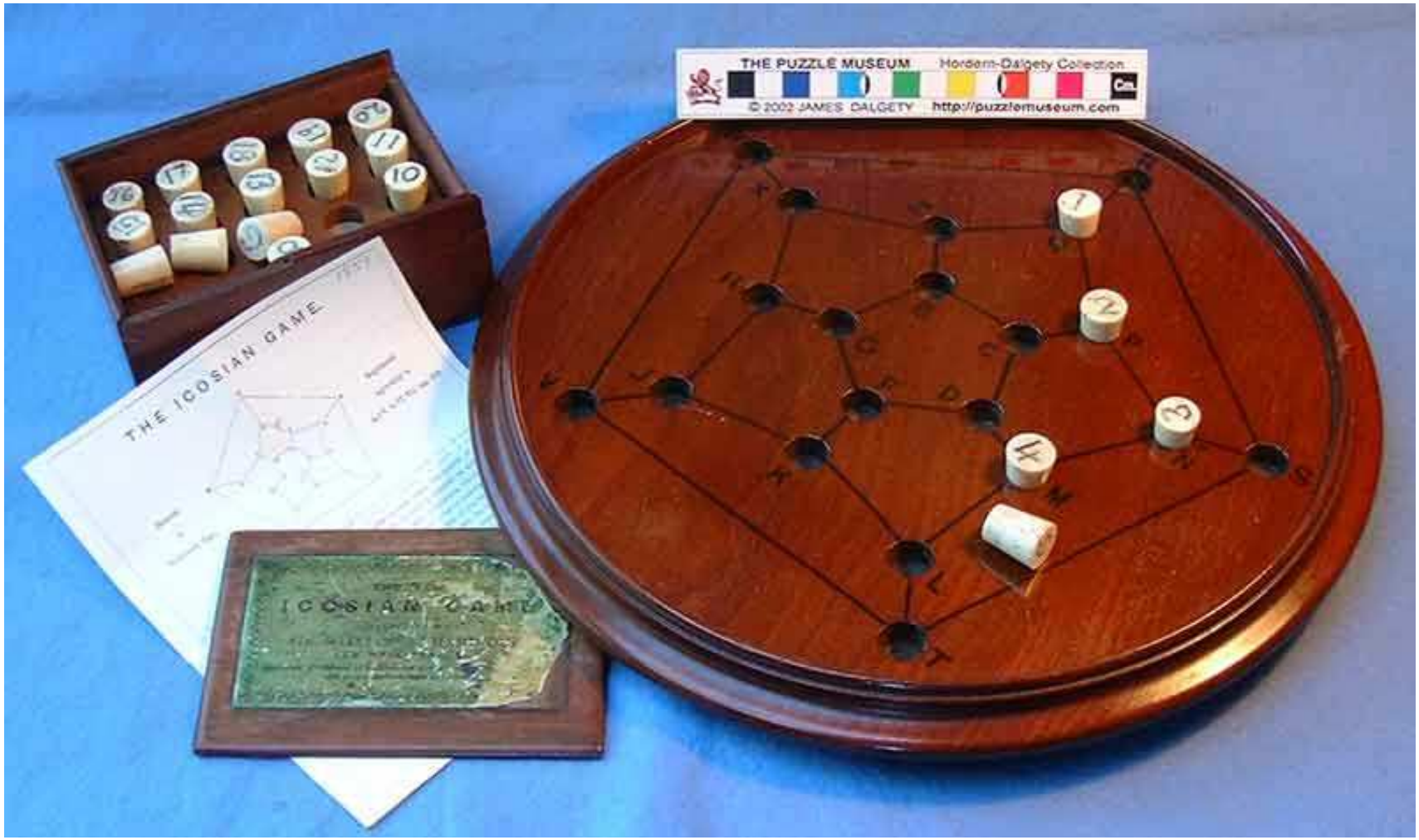
Perché ciclo hamiltoniano?

William Rowan Hamilton (1805-1865): in un dodecaedro regolare è possibile partire da un vertice e, passando sugli spigoli, toccare tutti i vertici una ed una sola volta fino a tornare al vertice di partenza?



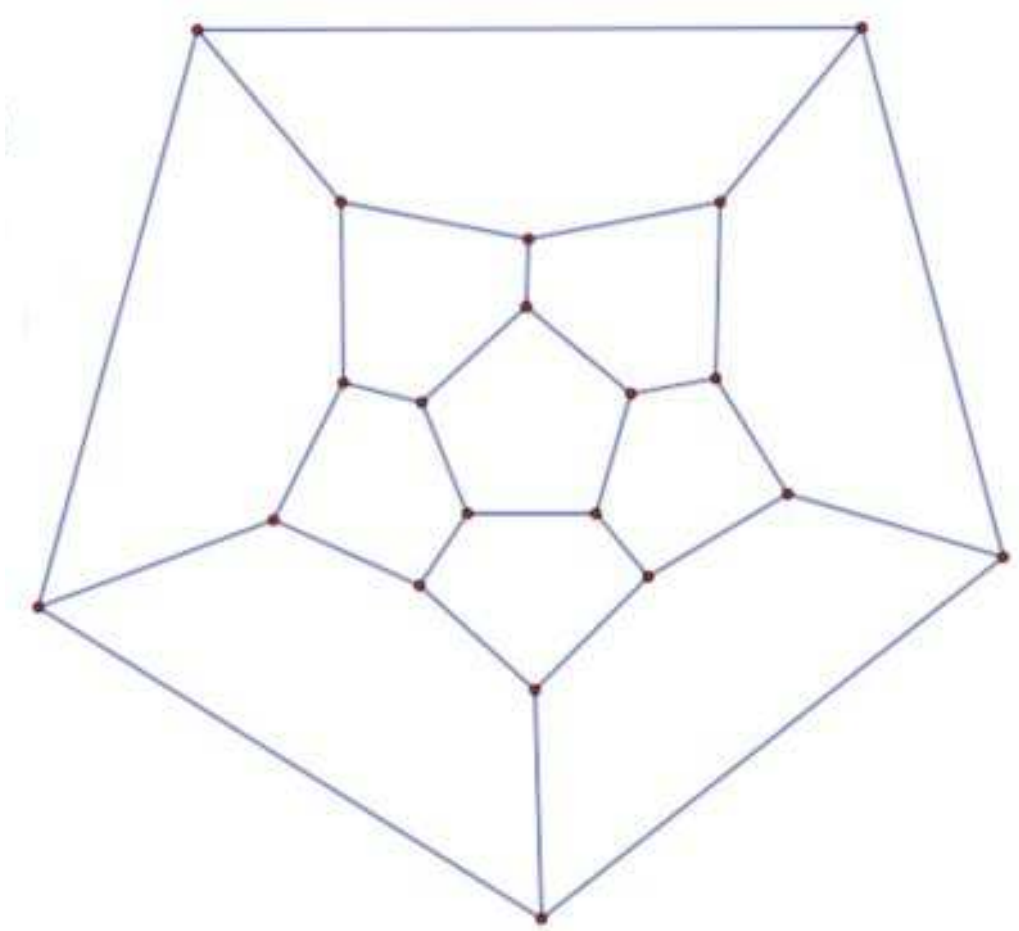
Perché ciclo hamiltoniano?

Icosian game: in un dodecaedro regolare è possibile partire da un vertice e, passando sugli spigoli, toccare tutti i vertici una ed una sola volta fino a tornare al vertice di partenza?



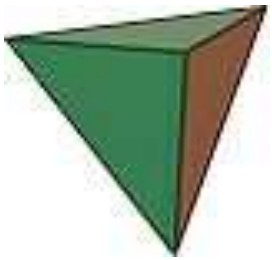
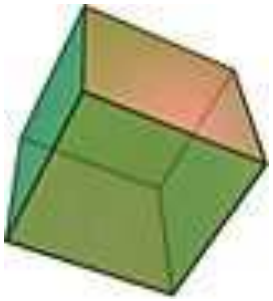
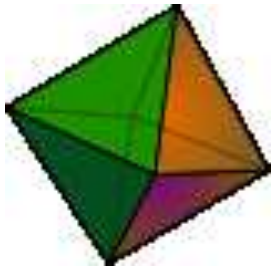
Perché ciclo hamiltoniano?

Esercizio 7.3. Risolvere l'icosian game:



Perché ciclo hamiltoniano?

Esercizio 7.4. Per ognuno degli altri 4 poliedri regolari in $\mathbb{R}^3$ (tetraedro, cubo, ottaedro, icosaedro)

costruire un grafo in cui i nodi rappresentano i vertici del poliedro e gli archi rappresentano gli spigoli e cercare un ciclo hamiltoniano in tale grafo.

Modello 1

Variabili: $x_{ij} = \begin{cases} 1 & \text{se } \{i,j\} \in \text{ciclo hamiltoniano,} \\ 0 & \text{altrimenti,} \end{cases}$ con $i < j$.

$$\min \sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} c_{ij}\, x_{ij}$$

$$\sum_{\substack{j \in N \\ j > i}} x_{ij} + \sum_{\substack{j \in N \\ j < i}} x_{ji} = 2 \qquad \forall\, i \in N \tag{4}$$

$$\sum_{i \in S} \sum_{\substack{j \notin S \\ j > i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j > i}} x_{ij} \geq 1 \quad \forall\, S \subset N, \quad |S| \geq 1 \tag{5}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i,j \in N, \quad i < j$$

(4): due archi incidenti su ogni nodo (cioè ogni nodo ha grado 2).
(5): eliminazione di sottocicli.

Modello 2

Variabili: $x_{ij} = \begin{cases} 1 & \text{se } \{i,j\} \in \text{ciclo hamiltoniano,} \\ 0 & \text{altrimenti,} \end{cases}$ con $i < j$.

$$\min \sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} c_{ij}\, x_{ij}$$

$$\sum_{\substack{j \in N \\ j > i}} x_{ij} + \sum_{\substack{j \in N \\ j < i}} x_{ji} = 2 \qquad \forall\, i \in N \tag{6}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j > i}} x_{ij} \leq |S| - 1 \quad \forall\, S \subset N, \quad |S| \geq 3 \tag{7}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i,j \in N, \quad i < j$$

(6): ogni nodo ha grado 2.
(7): eliminazione di sottocicli.

r-albero

Fissiamo un nodo r nel grafo. Un r–albero è un insieme di n archi di cui

- 2 sono incidenti sul nodo r
- $n - 2$ formano un albero di copertura sui nodi diversi da r

Ogni ciclo hamiltoniano è un r–albero.

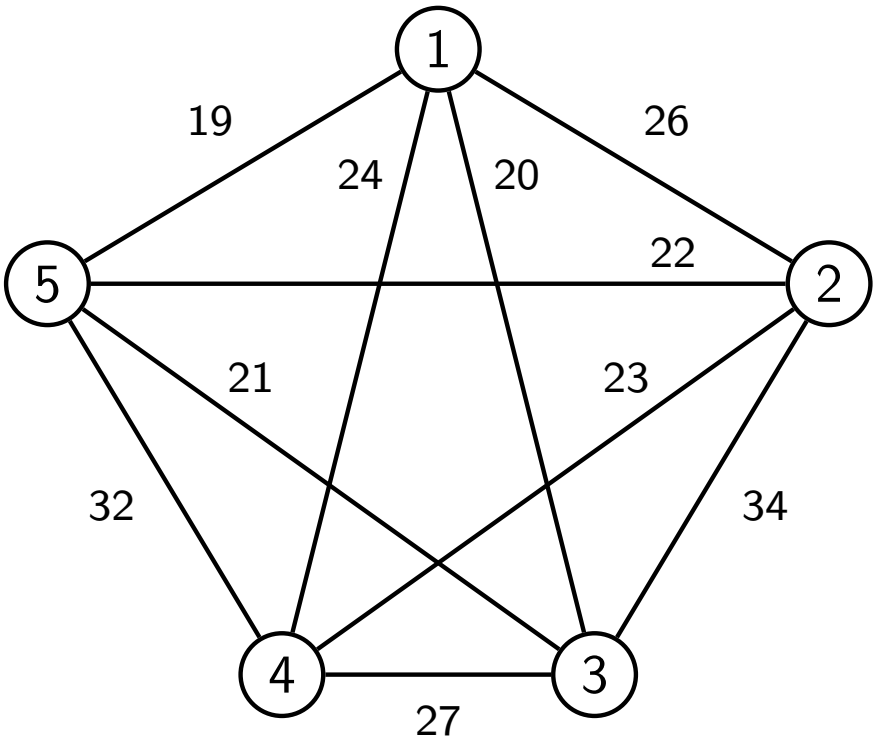
Il problema dell'r–albero di costo minimo è un rilassamento del TSP simmetrico. Inoltre è facile da risolvere, perché basta trovare

- 2 archi di costo minimo incidenti sul nodo r (ovvio)
- un albero di copertura di costo minimo sui nodi diversi da r (alg. Kruskal/Prim)

Rilassamento: r-albero di costo minimo

Esempio

Il 3–albero di costo minimo sul grafo

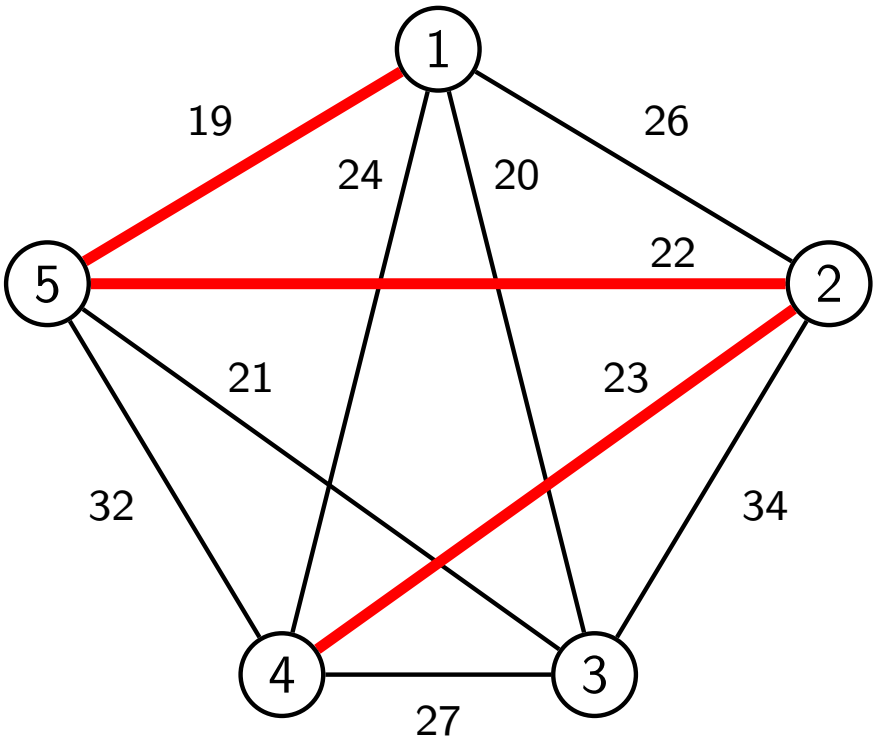


è formato dagli archi:

Rilassamento: r-albero di costo minimo

Esempio

Il 3–albero di costo minimo sul grafo

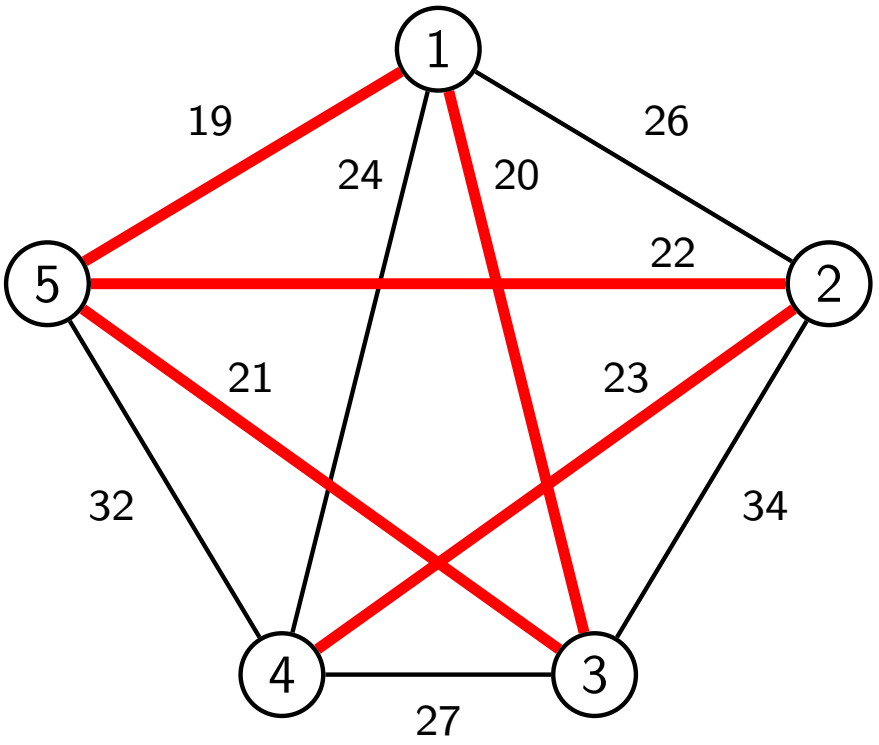


è formato dagli archi:
$\{1,5\}$, $\{2,5\}$, $\{2,4\}$ (albero di copertura sui nodi diversi da 3).

Rilassamento: r-albero di costo minimo

Esempio

Il 3–albero di costo minimo sul grafo



è formato dagli archi:
$\{1,5\}$, $\{2,5\}$, $\{2,4\}$ (albero di copertura sui nodi diversi da 3).
$\{1,3\}$, $\{3,5\}$ (incidenti sul nodo 3)
e ha costo $105 = v_I(P)$.

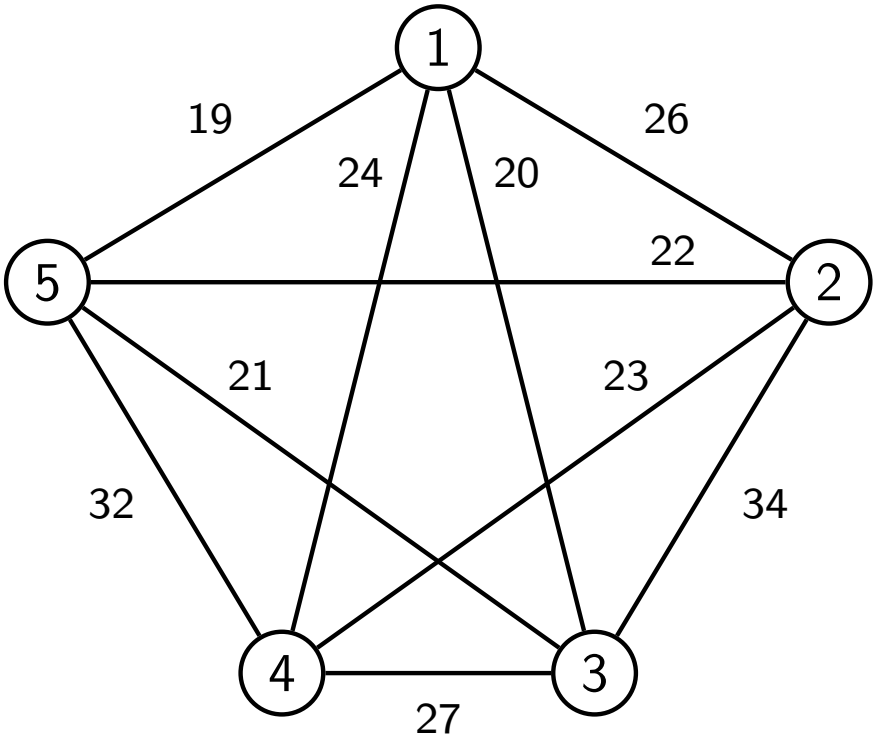
Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).

1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.

2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.

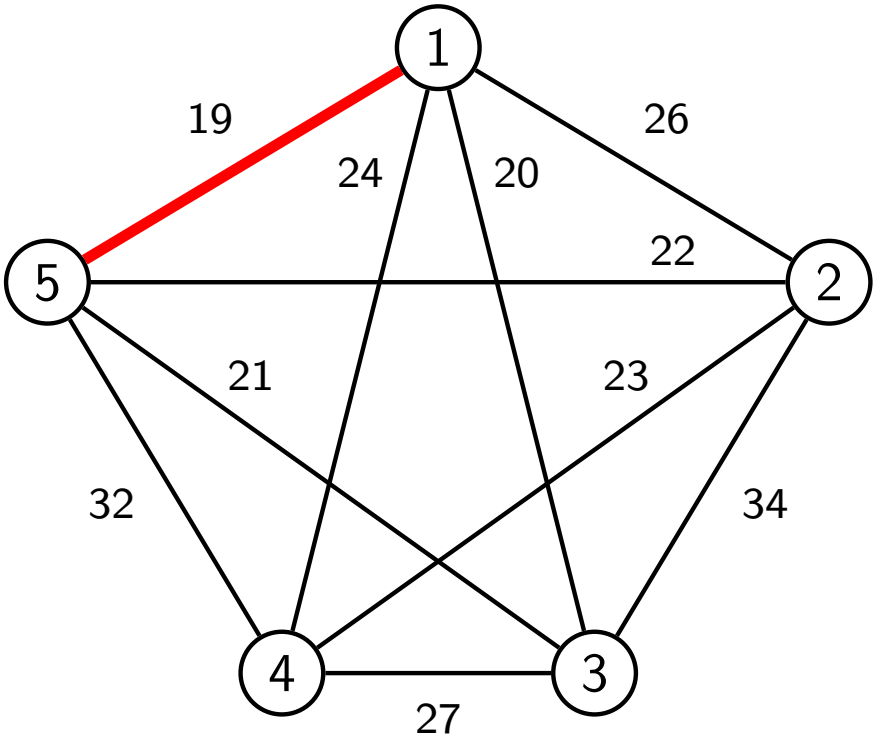
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo

Metodo euristico: algoritmo del nodo più vicino

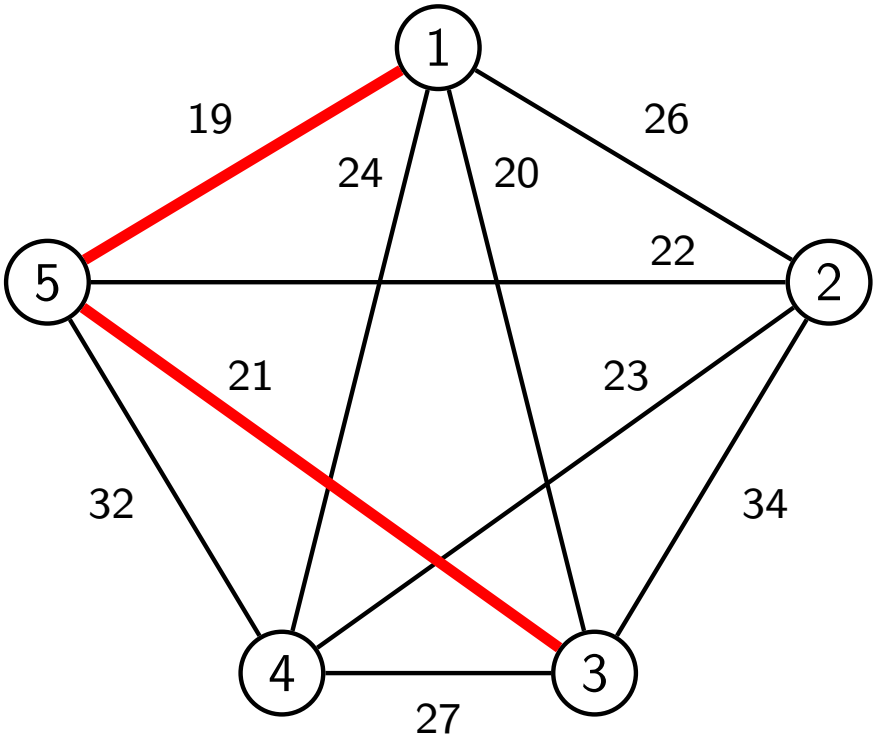
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5

Metodo euristico: algoritmo del nodo più vicino

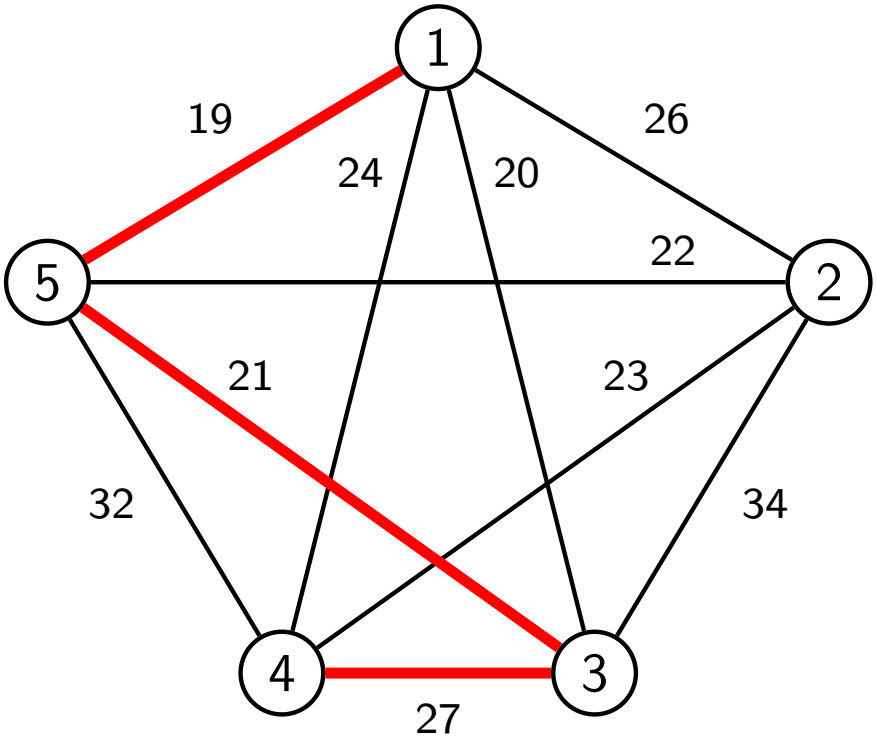
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3

Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4

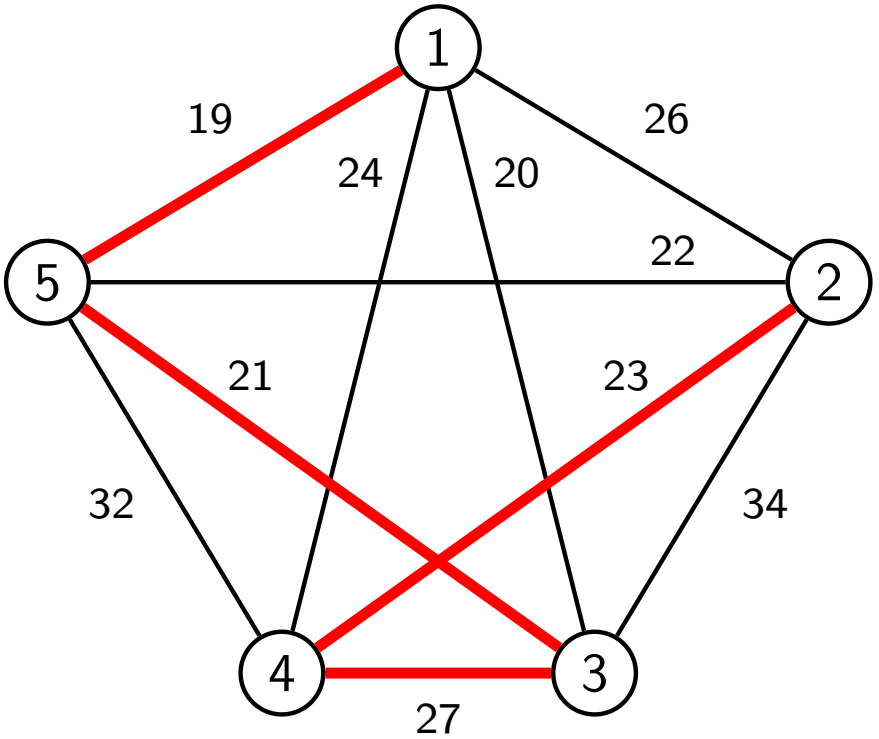
Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).

1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.

2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.

3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2

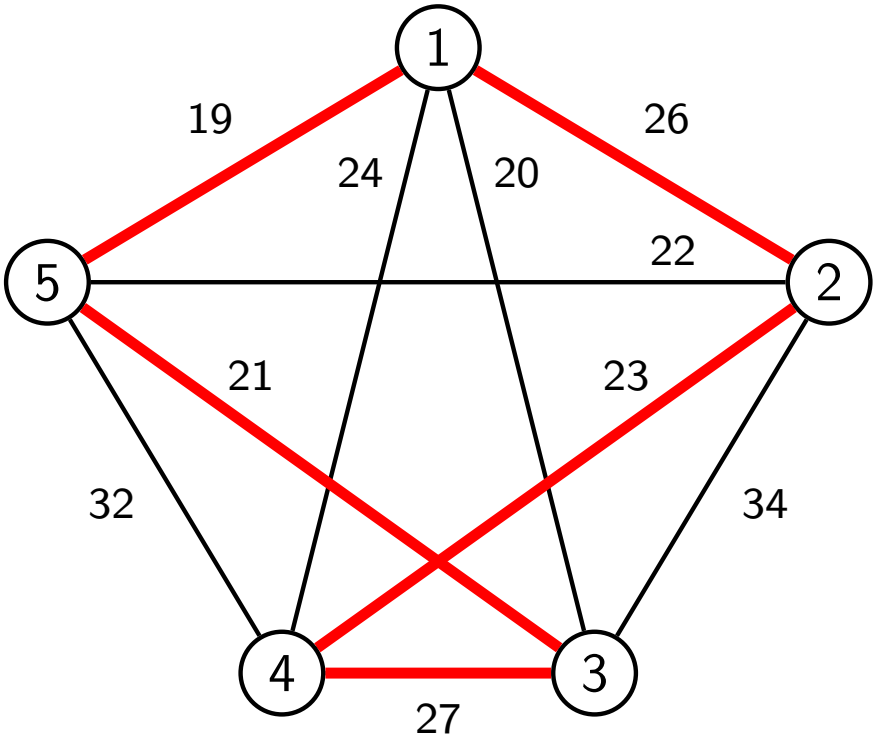
Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).

1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.

2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.

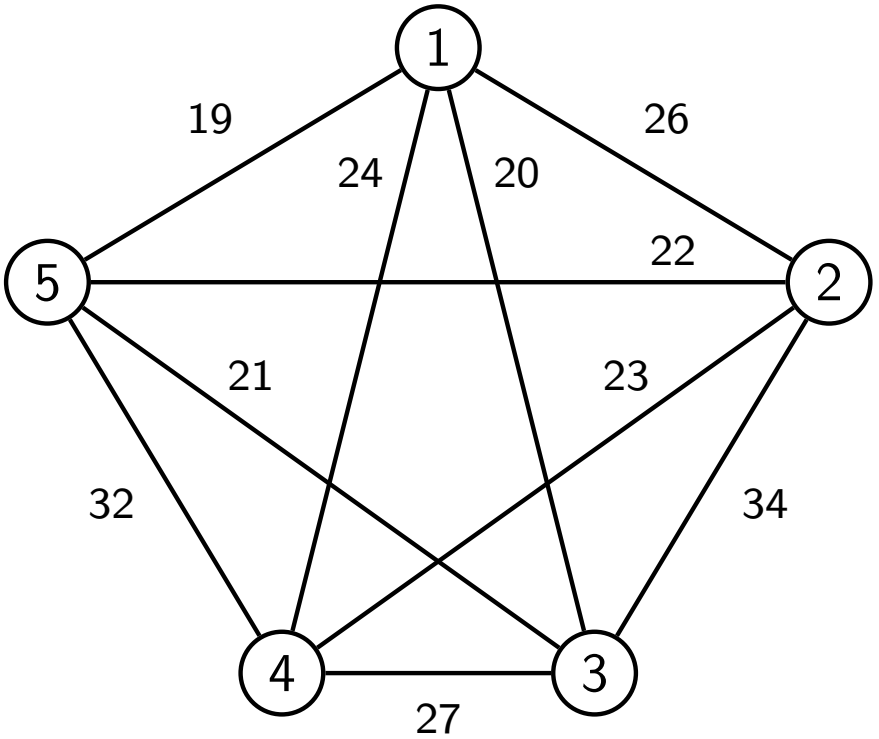
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.

Metodo euristico: algoritmo del nodo più vicino

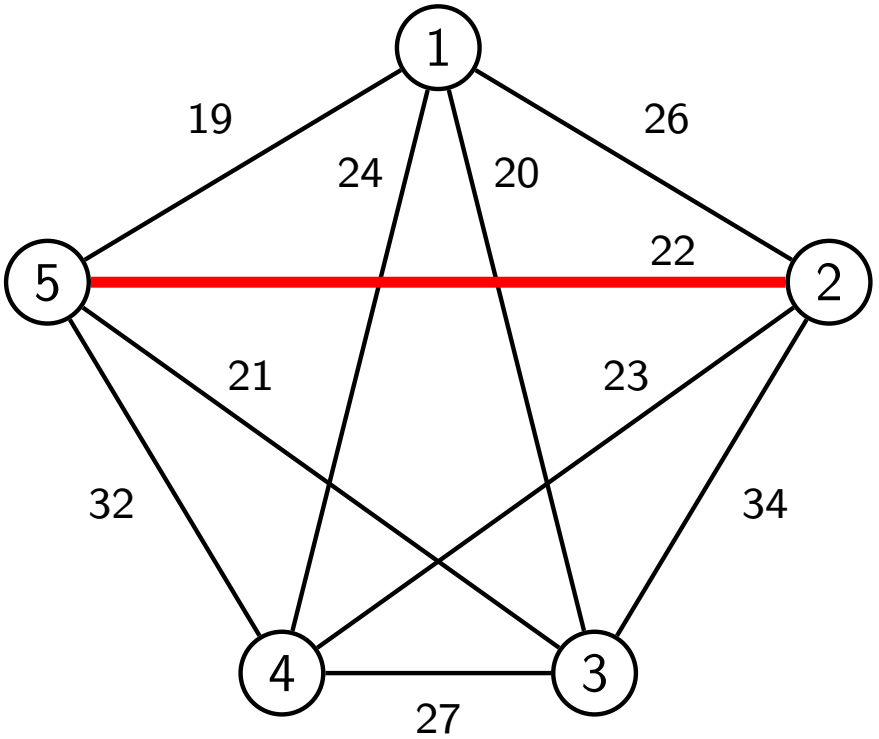
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo

Metodo euristico: algoritmo del nodo più vicino

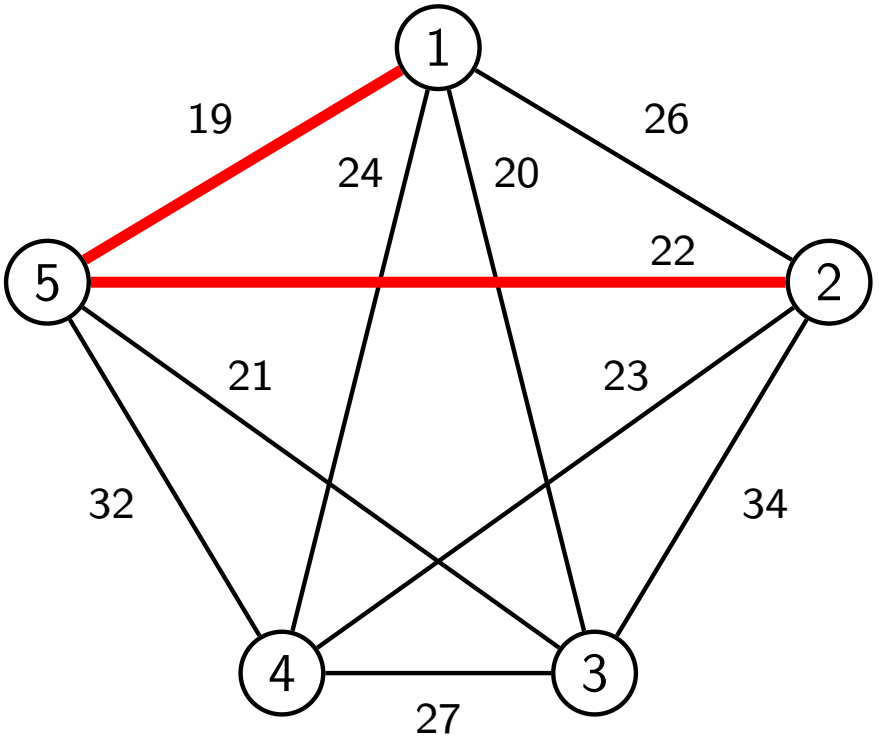
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo 2–5

Metodo euristico: algoritmo del nodo più vicino

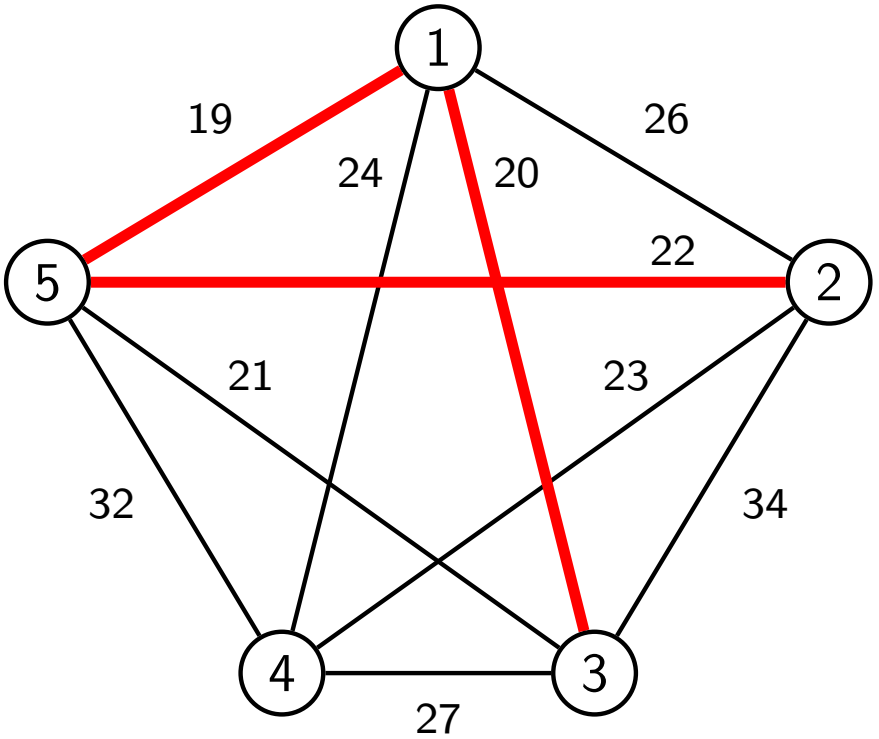
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo 2–5–1

Metodo euristico: algoritmo del nodo più vicino

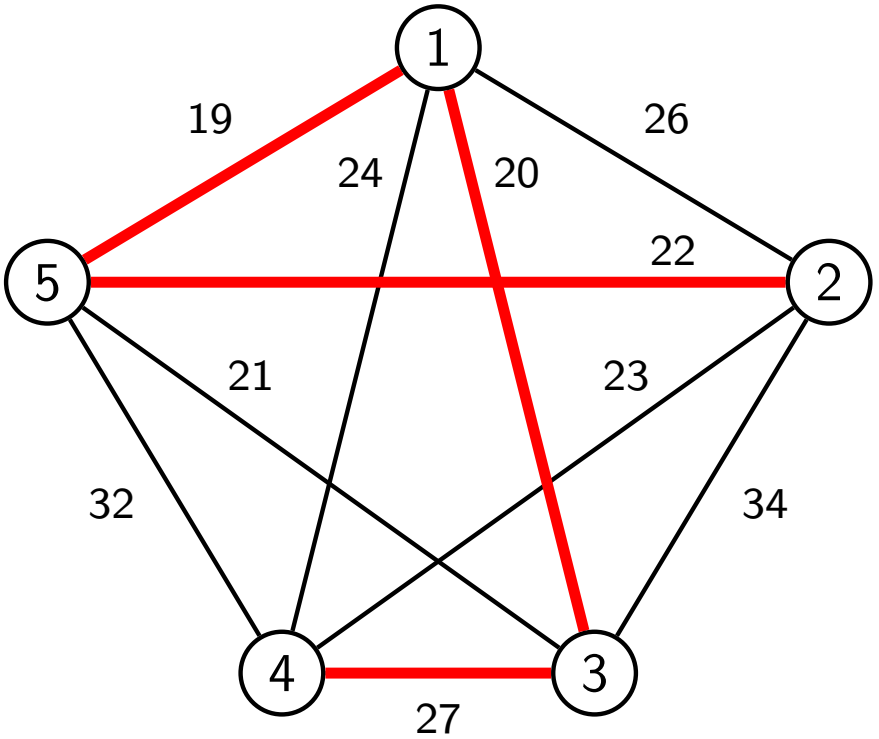
0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo 2–5–1–3

Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).
1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.
2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo 2–5–1–3–4

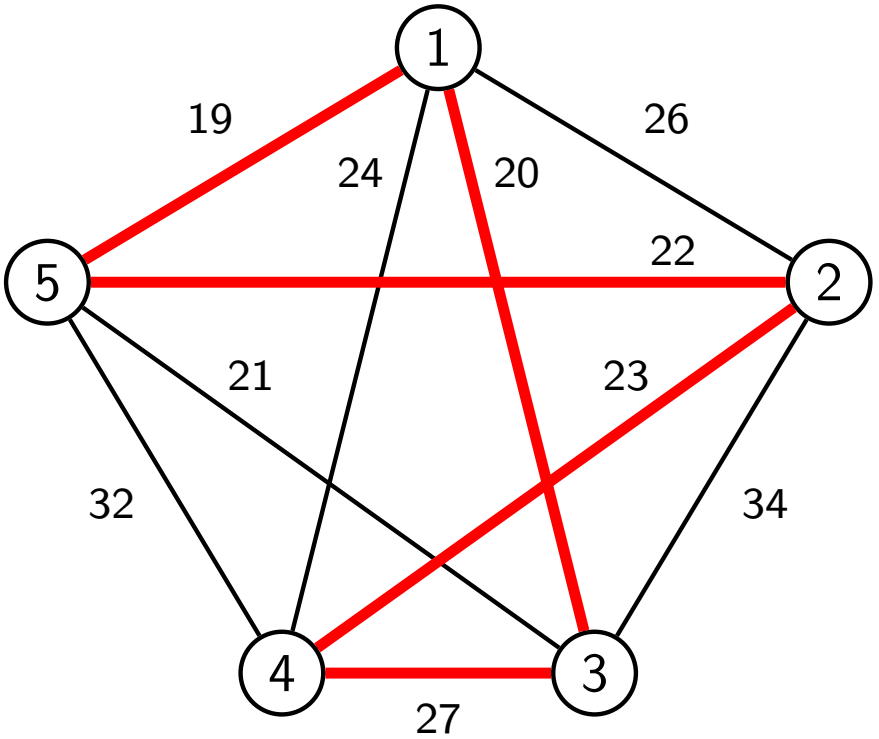
Metodo euristico: algoritmo del nodo più vicino

0. Scegli un nodo qualunque i, poni $k = i$ (nodo corrente), $U = N \setminus \{i\}$ (nodi non visitati), $C = i$ (ciclo - sequenza di nodi).

1. Se $U = \emptyset$, allora aggiungi i in fondo a C e STOP.

2. Scegli il nodo di U più vicino a k, cioè $j = \arg\min\{i \in U : \ c_{ki}\}$.

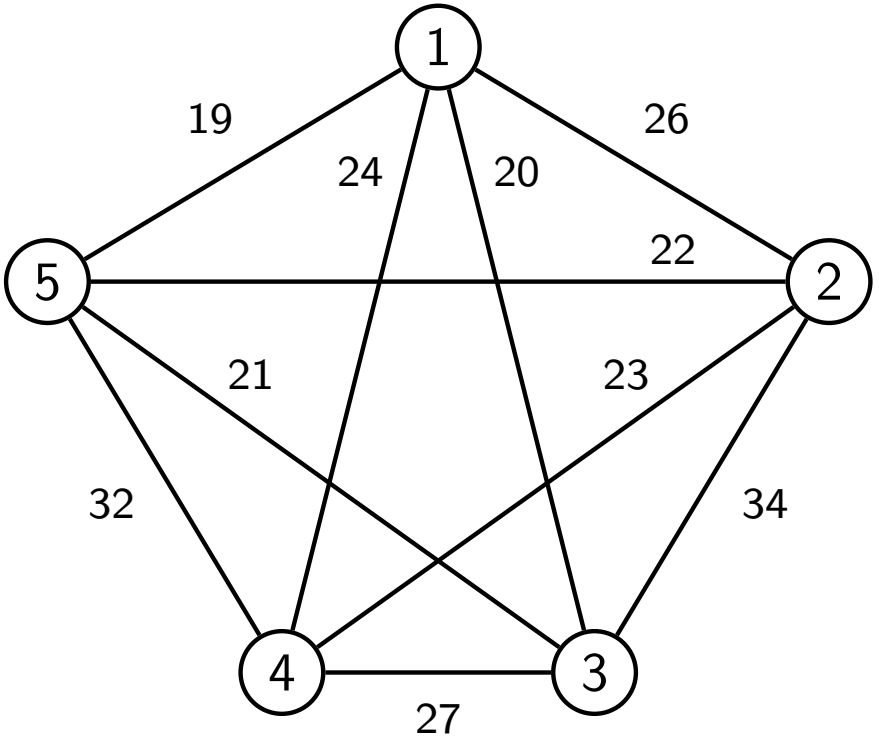
3. Aggiungi j in fondo a C, poni $U = U \setminus \{j\}$ e $k = j$. Torna al passo 1.



Partendo dal nodo 1 si ottiene il ciclo 1–5–3–4–2–1 di costo $116 = v_S(P)$.
Partendo dal nodo 2 si ottiene il ciclo 2–5–1–3–4–2 di costo $111 = v_S(P)$.

Metodo Branch and Bound

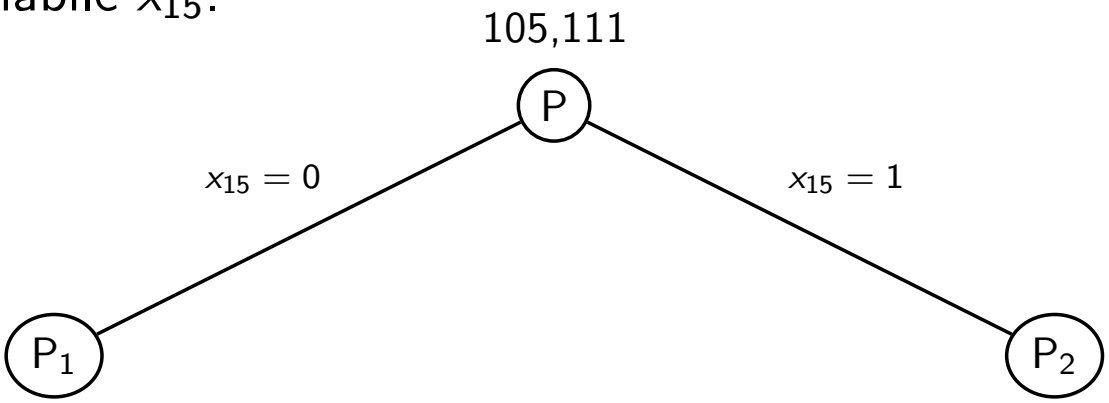
Utilizziamo il metodo Branch and Bound per trovare un ciclo hamiltoniano di costo minimo sul seguente grafo:



Bound: 3-albero di costo minimo.
Branch binario: istanziamo nell'ordine le variabili x_{15}, x_{35}, x_{13}.
Visita dell'albero decisionale: in ampiezza.

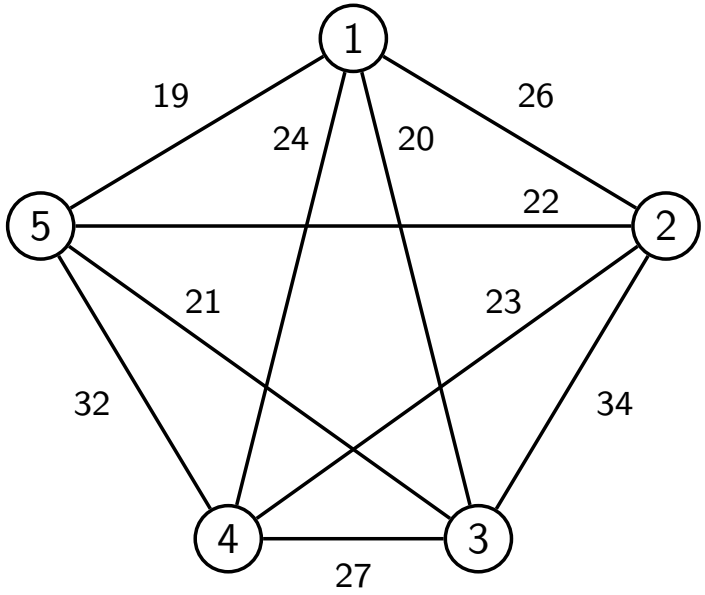
Metodo Branch and Bound

L'algoritmo del nodo più vicino fornisce $v_S(P) = 111$, mentre il 3-albero di costo minimo continuo fornisce $v_I(P) = 105$.
Istanziamo la variabile x_{15}.



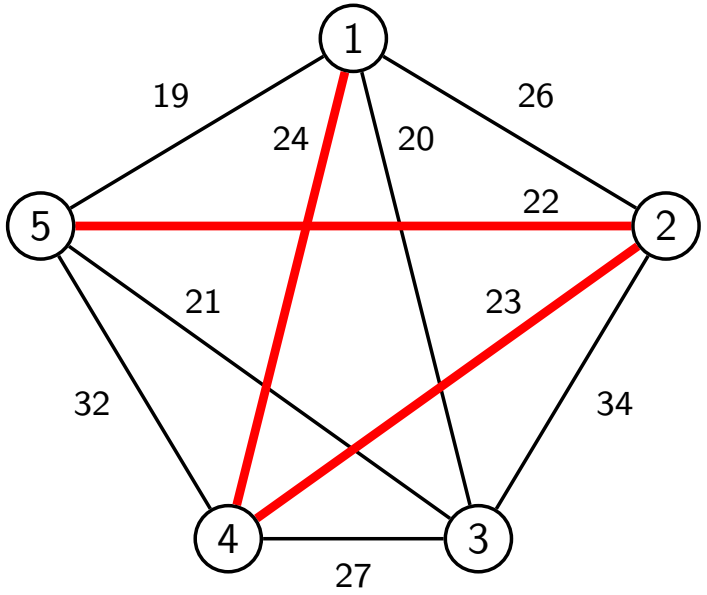
Metodo Branch and Bound

Rilassamento di P_1 ($x_{15} = 0$): il 3-albero di costo minimo contiene gli archi



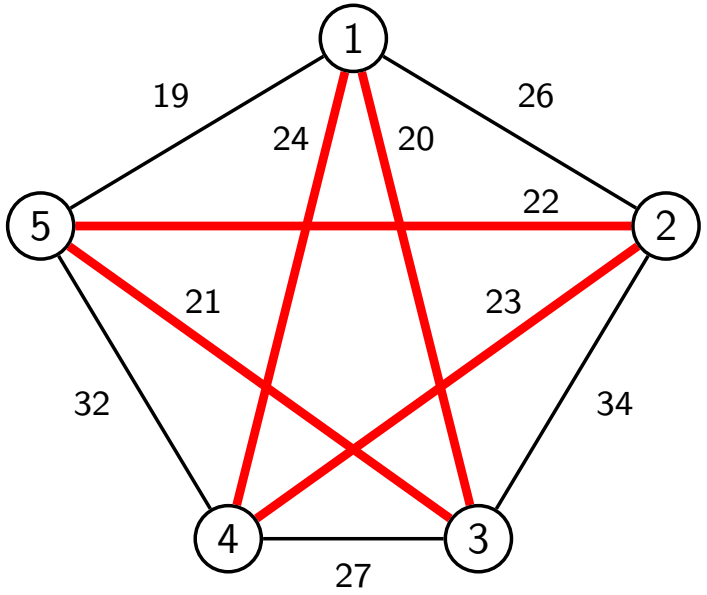
Metodo Branch and Bound

Rilassamento di P_1 ($x_{15} = 0$): il 3-albero di costo minimo contiene gli archi $\{2,5\}$, $\{2,4\}$, $\{1,4\}$ (albero di copertura sui nodi diversi da 3) e



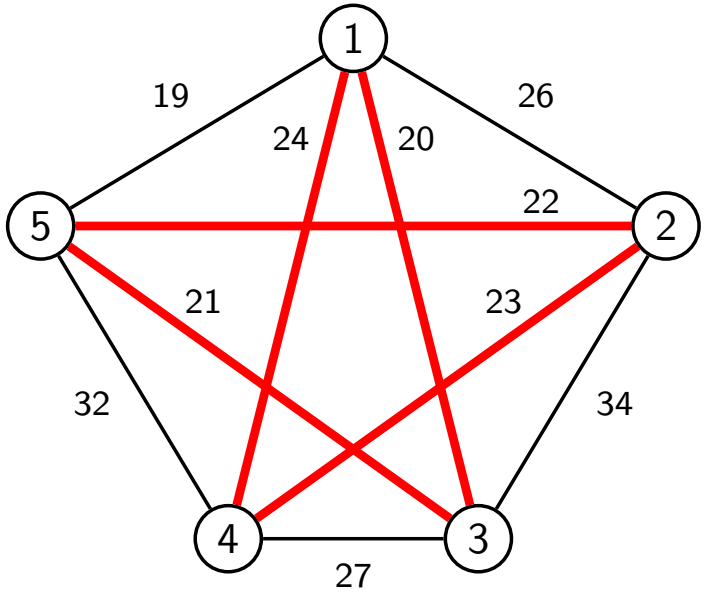
Metodo Branch and Bound

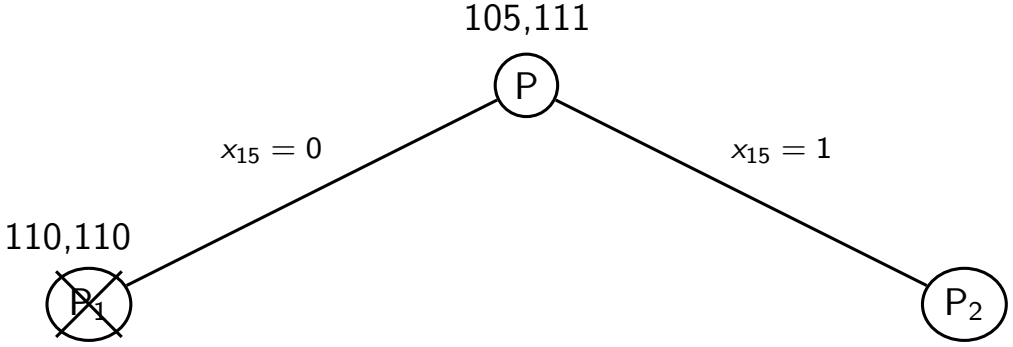
Rilassamento di P_1 ($x_{15} = 0$): il 3-albero di costo minimo contiene gli archi $\{2,5\}$, $\{2,4\}$, $\{1,4\}$ (albero di copertura sui nodi diversi da 3) e $\{1,3\}$, $\{3,5\}$ (incidenti sul nodo 3) e costa $110 = v_I(P_1)$.



Metodo Branch and Bound

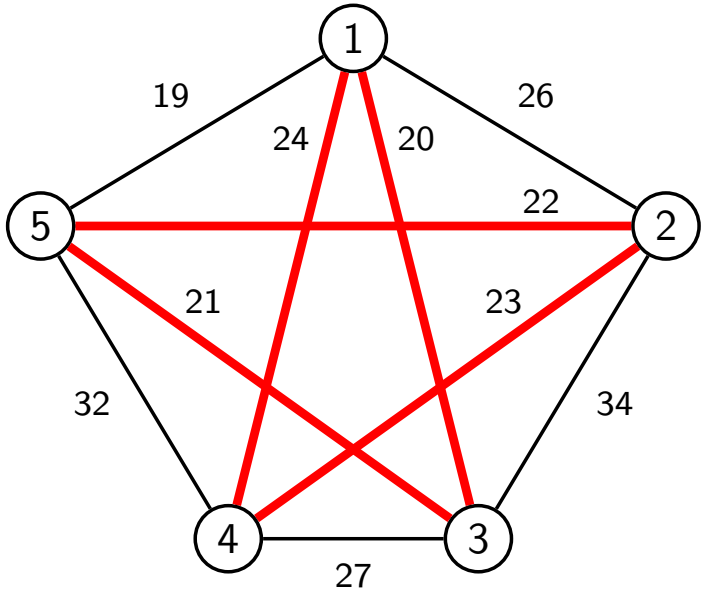
Rilassamento di P_1 ($x_{15} = 0$): il 3-albero di costo minimo contiene gli archi $\{2,5\}$, $\{2,4\}$, $\{1,4\}$ (albero di copertura sui nodi diversi da 3) e $\{1,3\}$, $\{3,5\}$ (incidenti sul nodo 3) e costa $110 = v_I(P_1)$. Questi archi formano il ciclo hamiltoniano 1-3-5-2-4-1, quindi chiudiamo P_1 e aggiorniamo $v_S(P) = 110$.

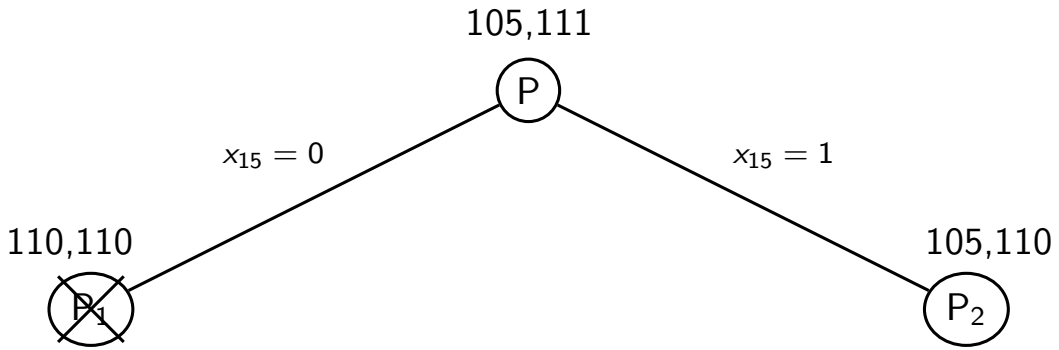




Metodo Branch and Bound

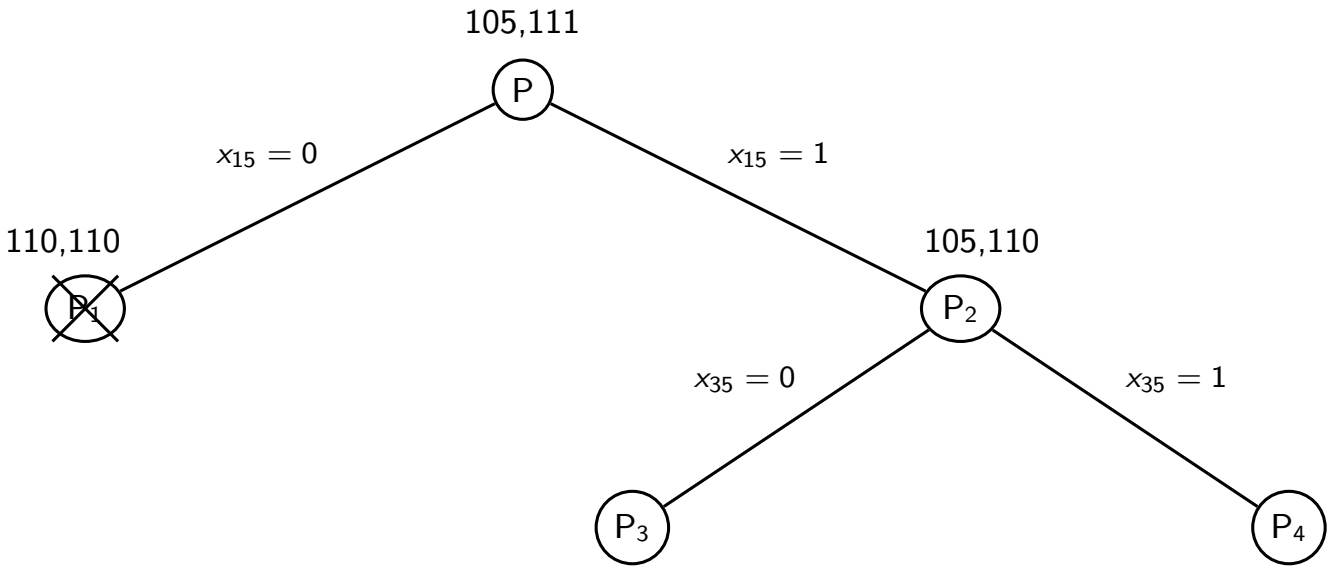
Rilassamento di P_1 ($x_{15} = 0$): il 3-albero di costo minimo contiene gli archi $\{2,5\}$, $\{2,4\}$, $\{1,4\}$ (albero di copertura sui nodi diversi da 3) e $\{1,3\}$, $\{3,5\}$ (incidenti sul nodo 3) e costa $110 = v_I(P_1)$. Questi archi formano il ciclo hamiltoniano 1-3-5-2-4-1, quindi chiudiamo P_1 e aggiorniamo $v_S(P) = 110$.



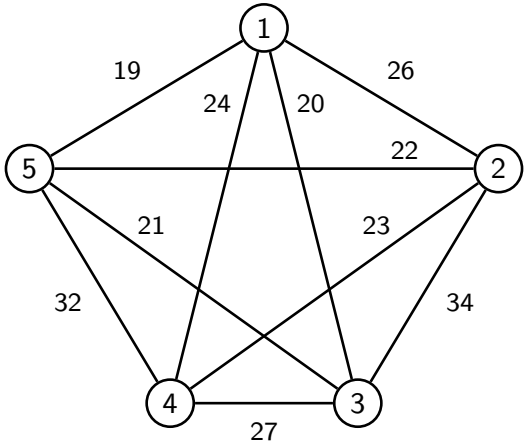


Rilassamento di P_2 ($x_{15} = 1$): il 3-albero di costo minimo coincide con quello calcolato nel nodo P perché quest'ultimo contiene l'arco (1,5), quindi $v_I(P_2) = 105 < 110 = v_S(P)$ e P_2 rimane aperto.

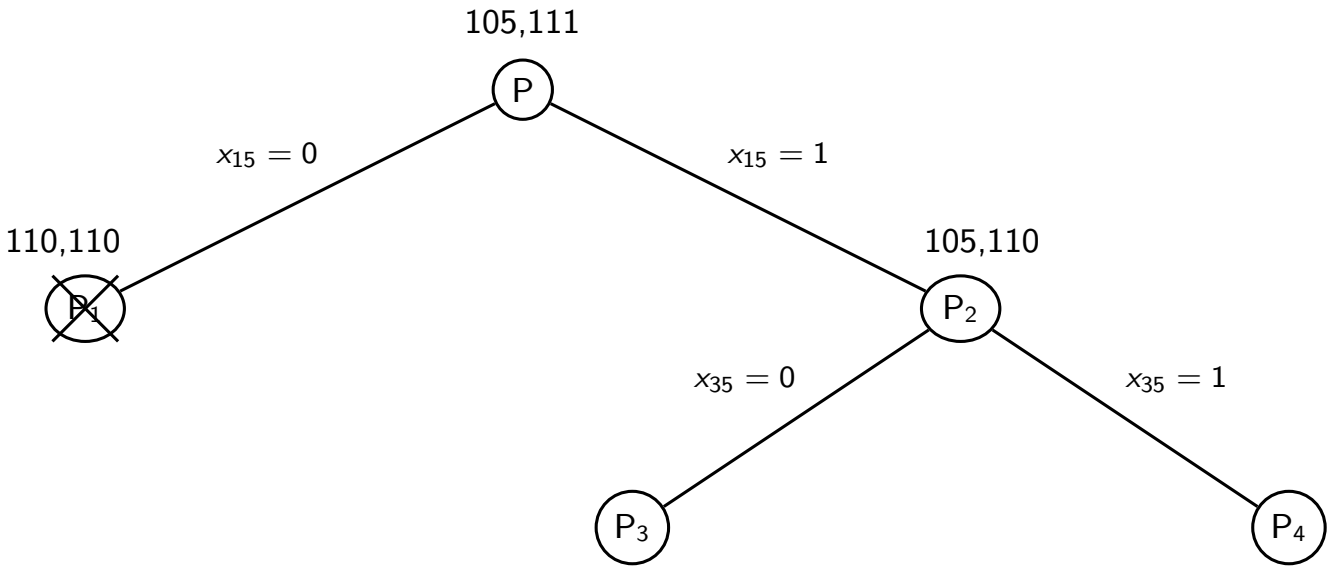
Metodo Branch and Bound



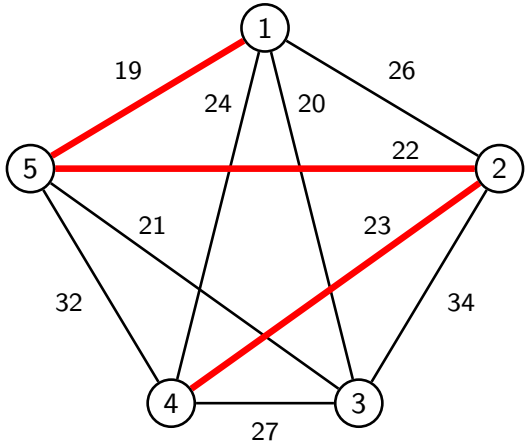
Rilassamento di P_3: il 3-albero di costo minimo contiene gli archi



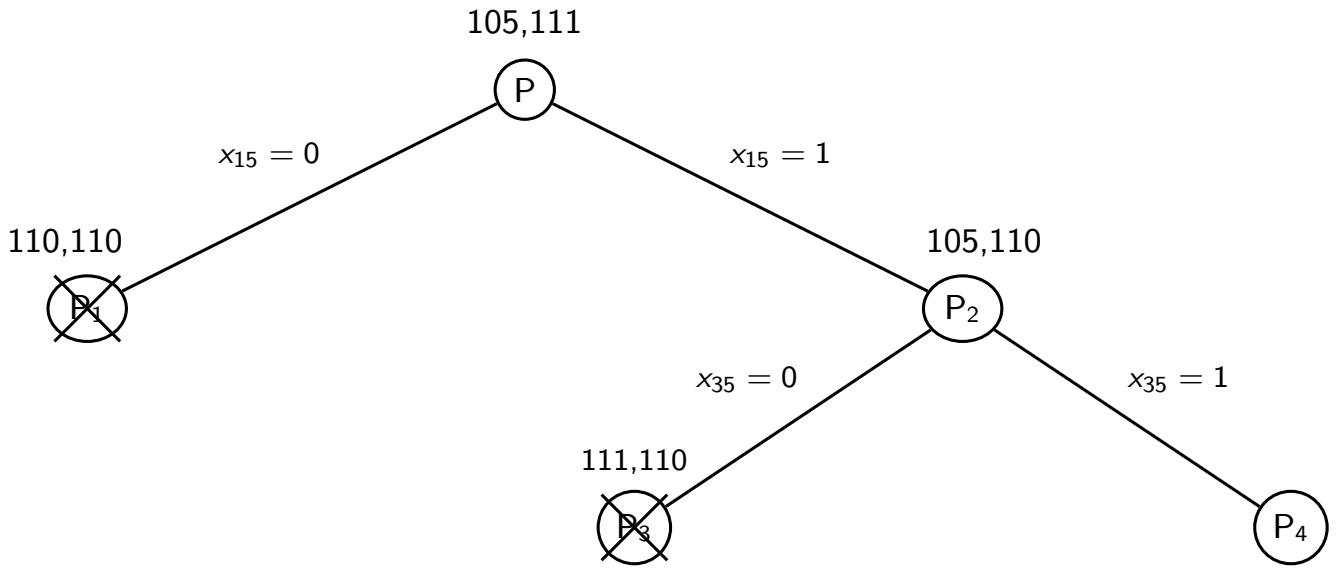
Metodo Branch and Bound



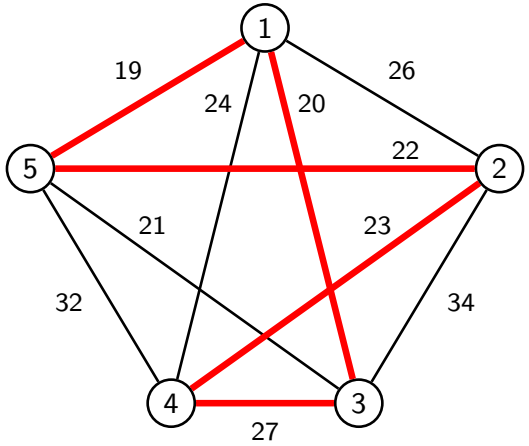
Rilassamento di P_3: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,4\}$, $\{2,5\}$ (albero di copertura sui nodi diversi da 3) e



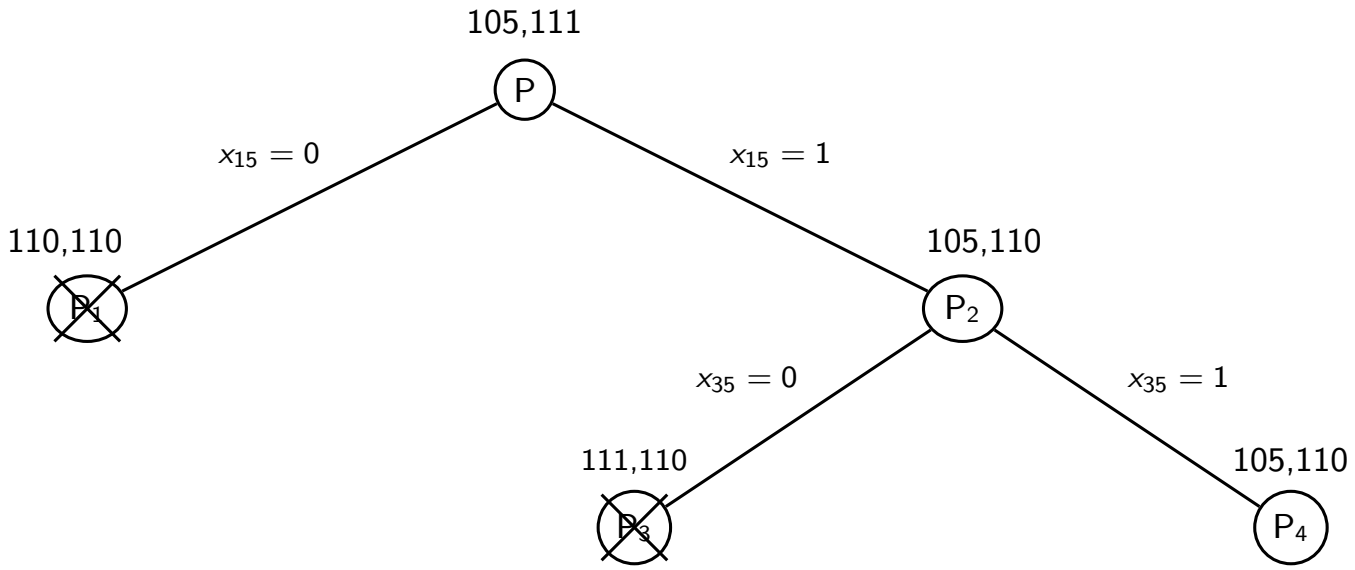
Metodo Branch and Bound



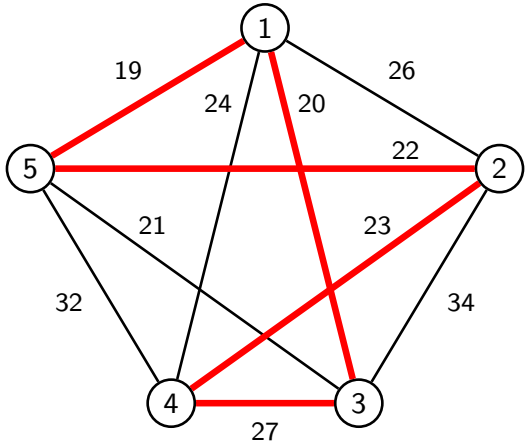
Rilassamento di P_3: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,4\}$, $\{2,5\}$ (albero di copertura sui nodi diversi da 3) e $\{1,3\}$, $\{3,4\}$ (incidenti sul nodo 3) e costa $111 = v_I(P_3) > 110 = v_S(P)$, quindi chiudiamo P_3.



Metodo Branch and Bound

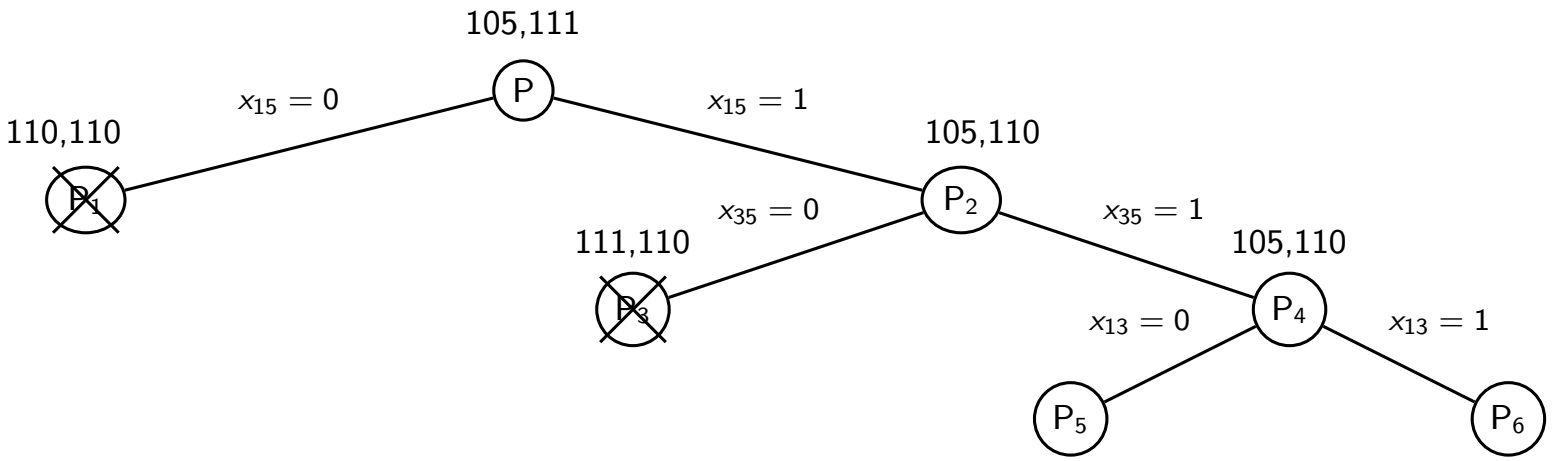


Rilassamento di P_3: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,4\}$, $\{2,5\}$ (albero di copertura sui nodi diversi da 3) e $\{1,3\}$, $\{3,4\}$ (incidenti sul nodo 3) e costa $111 = v_I(P_3) > 110 = v_S(P)$, quindi chiudiamo P_3.

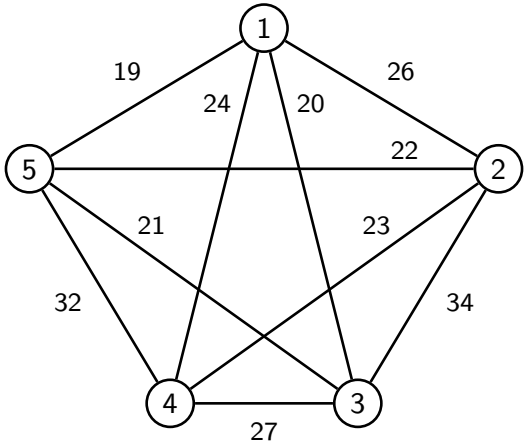


Rilassamento di P_4: il 3-albero di costo minimo coincide con quello calcolato nel nodo P perché quest'ultimo contiene gli archi (1,5) e (3,5), quindi $v_I(P_4) = 105 < 110 = v_S(P)$ e P_4 rimane aperto.

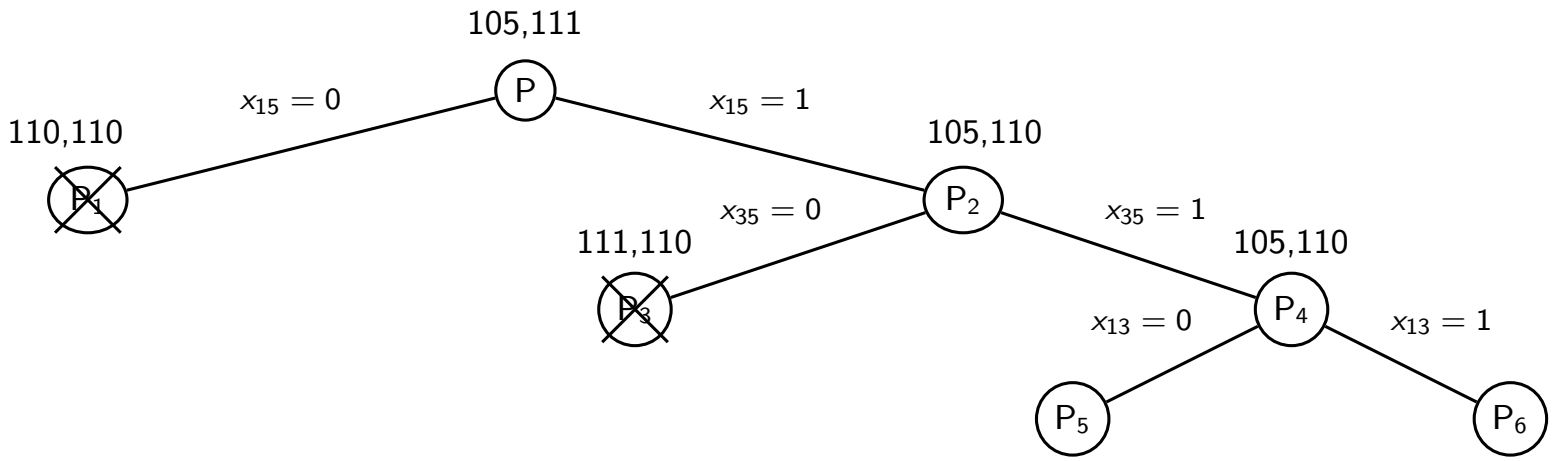
Metodo Branch and Bound



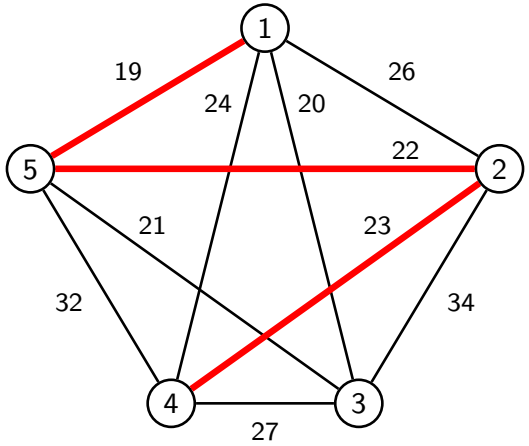
Rilassamento di P_5: il 3-albero di costo minimo contiene gli archi



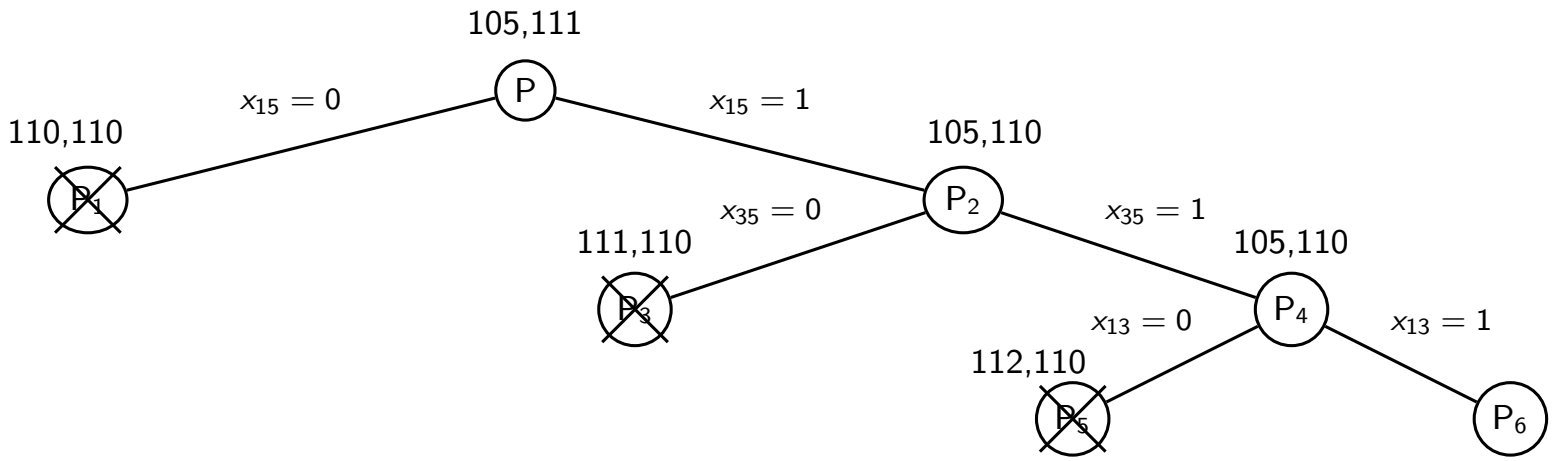
Metodo Branch and Bound



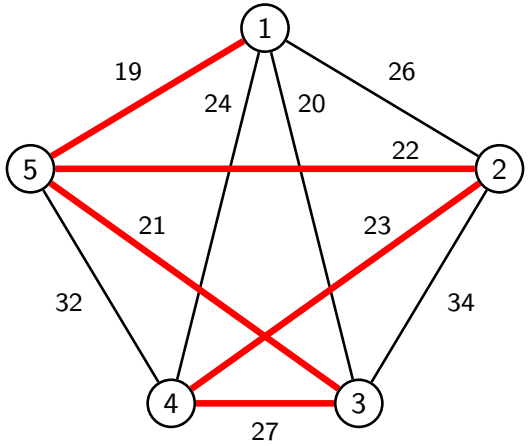
Rilassamento di P_5: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,5\}$, $\{2,4\}$ (albero di copertura sui nodi diversi da 3) e



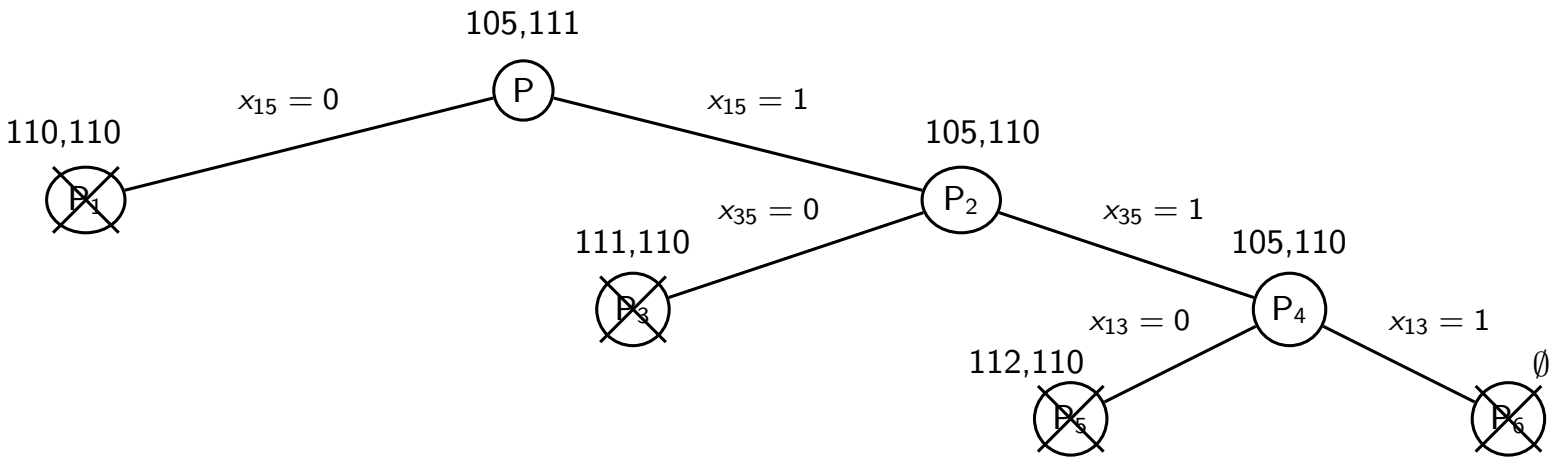
Metodo Branch and Bound



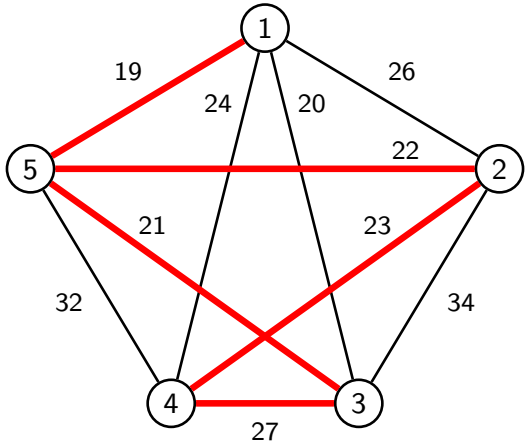
Rilassamento di P_5: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,5\}$, $\{2,4\}$ (albero di copertura sui nodi diversi da 3) e $\{3,5\}$, $\{3,4\}$ (incidenti sul nodo 3) e costa $112 = v_I(P_5) > 110 = v_S(P)$, quindi chiudiamo P_5.



Metodo Branch and Bound



Rilassamento di P_5: il 3-albero di costo minimo contiene gli archi $\{1,5\}$, $\{2,5\}$, $\{2,4\}$ (albero di copertura sui nodi diversi da 3) e $\{3,5\}$, $\{3,4\}$ (incidenti sul nodo 3) e costa $112 = v_I(P_5) > 110 = v_S(P)$, quindi chiudiamo P_5.



P_6 non contiene cicli hamiltoniani perché è presente il sottociclo 1-5-3, quindi chiudiamo anche P_6.
Pertanto il ciclo hamiltoniano di costo minimo è 1-3-5-2-4-1 di costo 110.