

Gran Compendio OLI

Meschio

2021
V1.2.3

Abstract

Guida passo a passo per alcune tipologie di esercizi con immagini e ricche spiegazioni, frammenti di altri esercizi con casi particolari, il tutto organizzato per una facile consultazione in sede di esame o per apprendere i passaggi in fase di studio.

Non è un sostituto all'esercizio e lo studio, l'esame richiede una buona dose di pratica e di comprensione teorica, l'uso esclusivo del compendio all'esame senza i punti precedenti vi porterà ad imparare una costosa ed evitabile lezione.

Contents

1	Modelli	4
1.1	Cenni di Base	4
1.2	Esercizi	6
1.2.1	Problema con Delta	6
1.2.2	Problema con massima distanza	7
1.2.3	Problema Variabile Triplo indice	8
1.2.4	Problema di Trasporto	10
1.2.5	Problema di Stoccaggio	12
1.2.6	Problema di Stoccaggio 2	13
2	Forme: Standard, Canonica, Generale	14
2.1	Standard \Rightarrow Canonica	14
2.2	Generale \Rightarrow Standard	14
2.3	Esercizio Trasformazione Gran Fritto Misto	15
3	Matrici : LP, ILP	16
3.1	Fondamenti Concettuali	16
3.1.1	Quale Simplex?	16
3.2	LP	16
3.2.1	Simplex	16
3.2.2	Simplex Duale	17

3.2.3	2Fasi	18
3.2.4	Duale del problema	20
3.3	Rappresentazione Grafica	23
3.4	ILP	25
3.4.1	Tagli di Gomory	25
3.5	Esercizi Particolari	27
3.5.1	Simpleso con variabili free	27
3.5.2	Problema PLC con simpleso e gomory in salsa teriyaki	29
3.5.3	Sensitivity Analysis	31
4	Grafi	33
4.1	GT	33
4.1.1	Dijkstra	33
4.1.2	Shortest Path Tree	35
4.2	SST	36
4.2.1	Cenni di Base	36
4.2.2	SST Prim's	37
4.2.3	SST Da Matrice trovare la soluzione ottimale	38
4.3	Max Flow	39
4.3.1	Cenni di Base	39
4.3.2	Flow Network	40
4.4	DP	42
4.4.1	DP Knapsack 0-1 Dynamic Programming	42
4.4.2	DP: SSP Bellman's-Ford	44
4.5	ILP	46
4.5.1	ILP standard B&B	46
4.5.2	ILP esercizio standard B&B	50
4.5.3	ILP esercizio B&B 0-1	52
5	GPLK	54
5.1	Dal modello al codice	54
5.1.1	Modello Easy	54
5.1.2	Caso Particolare 1 Graffa	55
5.1.3	Caso Particolare 2 Sommatoria doppio insieme	57
5.2	Dal codice al modello	58
5.2.1	Normale	58
6	Domande varie di teoria	60
6.1	PLC degenerate	60
6.2	PLC sensitivity	61
6.3	B&B	61
6.4	PLC minimization	62
6.5	cutting Plane	62
6.6	PLC dual	63
6.7	Dijkstra	63
6.8	B&B knapsack	64

6.9	Branch & cut	64
6.10	PLC	64
6.11	Soluzione Base	65
6.12	Ford-Fulkerson	65
6.13	PLI minimization & relaxation	66
6.14	B&B knapsack 0-1	66
6.15	NP problem	67

1 Modelli

Mi dispiace, qua non troverete guide per fare i modelli, sfortunatamente questi van fatti alla nausea cercando di capire di volta in volta e costruendosi il metodo, riporto solamente delle fattispecie di problemi con soluzioni di tipologie degne di nota.

1.1 Cenni di Base

- Stiamo risolvendo problemi **lineari**, questo implica che non potete e non dovete moltiplicare due variabili che avete creato in nessun modo, è un errore molto grave, ad esempio:

$$x_{ij} \in \{0, 1\} \text{ e } y_i \text{ integer } \geq 0, \text{ questa cosa è illegale: } \sum_{i \in J} x_{ij} y_i$$

È comunque possibile moltiplicare tra loro parametri del problema e variabili create ad hoc da voi.

- Il **Big M** è uno strumento che possiamo usare in certe circostanze, torna utile per impostare vincoli di verità, come ad esempio:

Supponiamo di dover produrre due prodotti A e B, il prodotto B ha una particolarità, nel caso ne facessimo anche solo 1, avremmo dei costi fissi c_B .

Ora noi impostiamo due variabili intere per rappresentare il numero di prodotti che stiamo producendo $x_A \geq 0, \text{integer}$ e $x_B \geq 0, \text{integer}$, ed una booleana per indicare se B è prodotto o no $y_B \in \{0, 1\}$, come possiamo mettere un vincolo che nel caso in cui **NON** stiamo producendo B, allora per forza di cose x_B dovrà essere ≤ 0 ?

$$x_B \leq M y_B$$

Perchè serve ? Perchè senza questo vincolo potrei avere $y_B = 0$ e $x_B = 340$, capite che qualcosa non funziona matematicamente.

Cosa stiamo facendo quindi? M rappresenta un numero molto grosso, nel momento in cui y_B varrà 1, cioè stiamo producendo il prodotto B, avremo x_B come intero positivo che sarà sempre minore di M, e il tutto rispetta il vincolo, nel caso in cui non produciamo B, y_B varrà 0 che annullerà M, rendendo l'equazione $x_B \leq 0$, quindi x_B dovrà per forza essere 0 a sua volta (x_B non può essere negativo dal vincolo messo sopra che la impone ≥ 0).

- **L'opposto di una booleana** o $(1 - x_{ij})$, questa è una chicca, serve per vedere la condizione opposta di uno stato di una booleana per un controllo inverso, vediamo un breve esempio:

Una azienda produce il prodotto X, la linea di produzione è formata da 3 differenti linee parallele identiche, 1,2,3, il prodotto X ha due lotti distinti, il lotto α e il lotto β , se produco il lotto α in una delle 3 linee, allora non posso produrre il lotto β in quella linea.

Pongo $x_{ij} \in \{0, 1\}, j \in \{1, 2, 3\}, i \in \{\alpha, \beta\}$ che rappresenta un booleano, che vale 1, nel caso in cui il lotto i è prodotto nella linea j , 0 altrimenti.

Come posso fare il controllo di verità di questa condizione con un vincolo matematico ?

$$\sum_{i \in \alpha} x_{ij} \leq (1 - x_{ki})M \quad i \in \{1, 2, 3\}, k \in \beta$$

Sto ciclando a sinistra della disuguaglianza tutti i prodotti X del lotto α , la j è vincolata fuori quindi guardiamo una linea alla volta, a destra della disuguaglianza sto dicendo che nel caso in cui x_{kj} dove k è l'insieme del lotto β vale 1, quindi lo stiamo producendo nella stessa linea produttiva i , allora avendolo posto negativo, si annulla con l'1 : $(1 - x_{ki})$, valendo 0 e di conseguenza, rompendo il vincolo.

Il Big M a destra della parentesi, serve nel caso in cui non produciamo β in quella specifica linea di produzione, e l'uno risultante dalla tonda che ne deriva, deve essere moltiplicato con qualcosa di grande per rendere vera l'equazione di sinistra.

- Δ , il delta viene usato per rappresentare la differenza di variabili, come può essere l'età media delle persone, se in un problema vedete un punto che risulta simile a: "minimizzare il valore assoluto della differenza massima tra l'età media di un gruppo e un altro"
- **Valore assoluto di un insieme** $|A|$, è semplicemente un modo per esprimere il limite superiore di un insieme che non ha una definizione numerica, ad esempio se ho un insieme A che rappresenta i nasi che ha un circo disponibile da allocare ai suoi clown, allora se pongo $x_i \geq 0$, *integer* come il numero totale di nasi che il gruppo di clown i può prelevare, posso scrivere:
 $x_i \leq |A|$

1.2 Esercizi

1.2.1 Problema con Delta

Esame 2020-06-25 Ex 1

In a summer camp there are 70 children to be allocated to 10 groups, each one with an educator that must supervise exactly 7 children.

The age of each child is $e_i, i = 1, \dots, 70$.

We want to minimize the absolute value of the maximum difference between the average age of a group and the age of each child allocated to that group (example: if the 7 children of a group have ages 8,9,10,11,11,12,12 the maximum difference for this group is $—73/7 - 8 = 2.43—$).

Write an integer linear programming model in order to decide how to allocate the children to the groups

Clearly define the used variables.

Soluzione:

$x_{ig} = 1$ se il bimbo $i = 1, \dots, 70$ è nel gruppo $g = 1 \in 10$, 0 altrimenti.

Δ massima differenza di età (valore assoluto).

$min : \Delta$

$$\Delta \geq \sum_{i=1}^{70} \frac{e_i x_{ig}}{7} - e_j x_{jg} \quad j = \{1, \dots, 70\}, g = \{1, \dots, 10\}$$

$$\Delta \geq - \sum_{i=1}^{70} \frac{e_i x_{ig}}{7} + e_j x_{jg} \quad j = \{1, \dots, 70\}, g = \{1, \dots, 10\}$$

$$\sum_{g=1}^{10} x_{ig} = 1 \quad i = \{1, \dots, 70\}$$

$$\sum_{i=1}^{70} x_{ig} = 7 \quad g = \{1, \dots, 10\}$$

$$x_{ig} \in \{0, 1\} \quad i = \{1, \dots, 70\}, g = \{1, \dots, 10\}$$

1.2.2 Problema con massima distanza

Esame 2020-07-16 Ex 1

A company that distributes medical products wants to open at least 1 out of 3 possible locations, to serve the set K of customers. d_{ik} represents the distance from location i to customer $k \in K$. We want to decide which warehouse(s) to open, and to assign each customer to exactly one warehouse. Write a MILP problem to minimize the maximum distance from a customer to the assigned warehouse. Clearly define the variables used.

Soluzione:

$y_{ik} = 1$ se il cliente $k \in K$ è assegnato al magazzino i , 0 altrimenti.

$z_i = 1$ se il magazzino nella località i è aperto, 0 altrimenti.

α massima distanza.

$$\begin{aligned} \min : & \alpha \\ \alpha \geq & d_{ik}y_{ik} && i = 1, 2, 3; k \in K \\ \sum_{i=1}^3 & z_i \geq 1 \\ \sum_{k \in K} & y_{ik} \leq |K|z_i && i = 1, 2, 3 \\ \sum_{i=1}^3 & y_{ik} = 1 && k \in K \\ y_{ik} \in & \{0, 1\} && i = 1, 2, 3; k \in K \\ z_i \in & \{0, 1\} && i = 1, 2, 3 \\ \alpha & && free \end{aligned}$$

1.2.3 Problema Variabile Triplo indice

Esame 2017-02-17 Ex 1

The national company for electric energy is going to plan the use of its power generators for a time horizon T of time periods. Let N be the set of power generators and let M be the set of customers. For each customer $j \in M$ and each time period $t \in T$ it is known the forecasted demand d_{jt} . For each power generator $i \in N$ it is known the maximum power P_i that can be produced in each time period, and the set $S(i) \subseteq M$ of the customers that may be served by generator i . A unit of energy produced by generator $i \in N$ in period $t \in T$ has a cost g_i . The energy produced by generator $i \in N$ can be transferred to customer $j \in M$ only if a *link* has been established between i and j . The installation of a link (i, j) has cost c_{ij} . To have a robust service, in case of failure of a power generator, it is required that each customer is linked to at least 2 generators.

Write a linear program to help the company to define an optimal plan to install the links among generators and customers, and to provide the required energy to each customer, while minimizing the total cost.

Improve the above model by adding the following constraints: in each time period each customer can receive energy from no more than three generators;

Soluzione:

$x_{ij} = 1$ se il generatore $i \in N$ è collegato al cliente $j \in S(i)$, 0 altrimenti.

f_{ijt} = quantità di energia che il generatore i da al cliente j , 0 altrimenti.

$\delta_{ijt} = 1$ se il generatore i fornisce energia al cliente j nel periodo t , 0 altrimenti.

$$\begin{aligned}
\min : z &= \sum_{i \in N} \sum_{j \in S(i)} c_{ij} x_{ij} + \sum_{i \in N} \sum_{j \in S(i)} \sum_{t \in T} g_{it} f_{ijt} \\
\sum_{i \in N: j \in S(i)} f_{ijt} &\geq d_{jt} && \forall j \in M; \forall t \in T; \\
\sum_{t \in T} f_{ijt} &\leq (|T| \max_{t \in T} d_{jt}) x_{ij} && \forall i \in N, \forall j \in S(i) \\
\sum_{j \in S(i)} f_{ijt} &\leq P_i && \forall i \in N, \forall t \in T \\
\sum_{i \in N: i \in S(i)} x_{ij} &\geq 2 && \forall j \in M \\
f_{ijt} &\leq d_{jt} \delta_{ijt} && \forall i \in N, \forall j \in S(i), \forall t \in T \\
\sum_{i \in N} \delta_{ijt} &\leq 3 && \forall j \in S(i), \forall t \in T \\
x_{ij} &\in \{0, 1\} && \forall i \in N, \forall j \in S(i) \\
f_{ijt} &\geq 0 && \forall i \in N, \forall j \in M, \forall t \in T \\
\delta_{ijt} &\in \{0, 1\} && \forall i \in N, \forall j \in S(i), \forall t \in T
\end{aligned}$$

1.2.4 Problema di Trasporto

Esame 2018-01-11 Ex 1

The transport operator CITYFAST is specialized in good distribution in large metropolis. The distribution into the city requires two transports. A first transport from the suppliers (outside the city) to a city distribution center, and a second transport from the distribution center to the customers. Each supplier provides only one type of good. Let I be the set of suppliers, J the set of possible distribution centers and K be the set of customers. The first transport is managed by an external company. CITYFAST pays c'_{ij} euro for each kilogram transported from supplier $i \in I$ to distribution center $j \in J$. Instead, for the second transport, CITYFAST uses its own fleet of identical ecological trucks, each with a given capacity C . The cost for a trip from the distribution center $j \in J$ to customer $k \in K$ (and return) is c''_{kj} . Each trip can deliver goods from several suppliers in order to use the truck capacity efficiently. Each trip serve a single customer. Each customer asks for d_{ik} Kg of the good provided by supplier $i \in I$ (with d_{ik} possibly zero if the good is not required). If a distribution center j is used the company has to pay a fixed una-tantum amount of s_j euro. Help the company to find an optimal transport policy by writing a Linear Program aimed to satisfy all the demands, while minimizing the total cost. (Suggestion: use a three index variable (supplier-distribution center-customer) to track the flows of goods.) Improve the previous model by adding the following constraint: given a subset of customers: $A \subseteq K$, it is required that all the customers of A are served by a single distribution center which is the same for all these customers.

Soluzione:

x_{ijk} = Kg del bene del fornitore i trasportati da un centro di distribuzione j al cliente k .

y_{jk} = numero di viaggi dal centro di distribuzione j al cliente k .

δ_j = 1 se il centro di distribuzione j è usato, 0 altrimenti.

M = un numero molto grande (costante).

z_{jk} 1 se il centro di distribuzione j serve il cliente $k \in A$, 0 altrimenti.

$$\begin{aligned}
\min : z &= \sum_{j \in J} s_j \delta_j + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} c'_{ij} x_{ijk} + \sum_{j \in J} \sum_{k \in K} c''_{jk} y_{jk} & k \in K, i \in I \\
\sum_{j \in J} x_{ijk} &= d_{ik} & k \in K, i \in I \\
\sum_{i \in I} x_{ijk} &\leq C y_{jk} & k \in K, j \in J \\
\sum_{i \in I} \sum_{k \in K} x_{ijk} &\leq M \delta_j & j \in J \\
x_{ijk} &\geq 0 & i \in I, j \in J, k \in K \\
y_{jk} &\geq 0, \text{ integer} & j \in J, k \in K \\
\delta_j &\in \{0, 1\} & j \in J \\
y_{jk} &\leq M z_{jk} & k \in A, j \in J \\
\sum_{j \in J} z_{jk} &= 1 & k \in A \\
z_{jk} &= z_{jh} & k, h \in A, j \in J
\end{aligned}$$

1.2.5 Problema di Stoccaggio

Esame 2017-07-17 Ex 3

A tile factory wants to organize the storage area of the pallets containing final products. There is a set I of pallets that have to be stored in a set L of possible locations. In each location one can store up to m pallets one over the other. Over each pallet of a special "fragile" set F one can store at most two pallets. Write a linear programming model to help the factory to find a feasible storage for all pallets (without objective function).

Let σ denote a sorting of the the pallets, i.e., $\sigma(i) < \sigma(j)$ indicates that pallet i precedes pallet j . In a second time the pallets will be picked from the storage area one after the other accordingly to the above ordering. If pallet i with $\sigma(i) < \sigma(j)$ is stored under pallet j (in the same location) before picking i one must move pallet j . Improve the linear programming model by adding an objective function which minimizes the movement of pallets.

Soluzione:

$x_{ilk} = 1$ se il pallet i è stoccato nella locazione l al livello k , 0 altrimenti.

$y_{ij} = 1$ se il pallet i è sotto al pallet j (stessa location), 0 altrimenti.

$$\begin{aligned} \min : z &= \sum_{i \in I} \sum_{j \in I: \sigma(i) < \sigma(j)} y_{ij} \\ \sum_{l \in L} \sum_{k=1}^m x_{ilk} &= 1 && i \in I \\ \sum_{i \in I} x_{ilk} &\leq 1 && l \in L, k = 1, \dots, m \\ \sum_{i \in I} x_{ilk} &\geq \sum_{i \in I} x_{il(k+1)} && l \in L, k = 1, \dots, m-1 \\ m(1 - x_{ilk}) &\geq \sum_{h=k+1}^m \sum_{j \in I: j \neq i} x_{ilh} - 2 && i \in F, l \in L, k = 1, \dots, m-3 \\ x_{ilk} \sum_{h=k+1}^m x_{jlk} - 1 &\leq y_{ij} && i, j \in I, i \neq j, l \in L, k = 1, \dots, m-1 \\ x_{ilk} &\in \{0, 1\} && i \in I, l \in L, k = 1, \dots, m \\ y_{ij} &\in \{0, 1\} && i, j \in I, i \neq j \end{aligned}$$

1.2.6 Problema di Stoccaggio 2

Esame 2020-06-08 Ex 2

A warehouse has to store n boxes in a rack with m shelves. The first m_1 shelves have length L_1 , while the remaining have length L_2 . Each box $j = 1, \dots, n$ has length l_j and a frequency of usage f_j .

Write a linear programming model that helps the warehouse to decide how to store the boxes, so that the sum of the frequencies of the boxes stored in the first m_1 shelves is maximized. (N.B. It is not known if all the boxes can be stored.)

Soluzione:

$x_{ij} = 1$ se la scatola j è stoccata nello scaffale i , 0 altrimenti.

$$\max : z = \sum_{i=1}^{m_1} \sum_{j=1}^n x_{ij} f_j$$

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

$$\sum_{j=1}^n l_j x_{ij} \leq L_1 \quad i = 1, \dots, m_1$$

$$\sum_{j=1}^n l_j x_{ij} \leq L_2 \quad i = m_1 + 1, \dots, m$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m, j = 1, \dots, n$$

2 Forme: Standard, Canonica, Generale

I problemi di LP possono essere rappresentati a sistema con 3 differenti forme, tutte equivalenti.

- **Standard**

- $\min\{c^T x : Ax = b, x_j \geq 0, j = 1, \dots, n\}$
- I vincoli sono uguaglianze "=", tutte le variabili hanno vincoli di non negatività.

- **Canonica**

- $\min\{c^T x : Ax \geq b, x_j \geq 0, j = 1, \dots, n\}$
- I vincoli sono disuguaglianze " \geq, \leq ", tutte le variabili hanno vincoli di non negatività.

- **Generale**

- $\min\{c^T x : Ax = b, A'x \geq b, x_j \geq 0, j \in J \subset \{1, \dots, n\}\}$
- I vincoli sono disuguaglianze " \geq, \leq " e uguaglianze "=", solo le variabili del sotto insieme J devono essere non-negative.

2.1 Standard \Rightarrow Canonica

Le uguaglianze, diventano disequazioni:

$$2x_1 + 5x_2 = 4 \Rightarrow \begin{cases} 2x_1 + 5x_2 \geq 4 \\ 2x_1 + 5x_2 \leq 4 \end{cases}$$

2.2 Generale \Rightarrow Standard

Le disuguaglianze diventano uguaglianze:

$$5x_1 + 3x_2 \leq 4 \quad \Rightarrow \quad 2x_1 + 5x_2 + S_1 = 8, S_1 \geq 0$$

S_1 è variabile di slack, che serve per bilanciare la disequazione, in questo altro esempio non sarebbe servita:

$$5x_1 + 3x_2 \geq 4 \quad \Rightarrow \quad 2x_1 + 5x_2 = 8$$

2.3 Esercizio Trasformazione Gran Fritto Misto

Esame 2020-06-25 Ex 3

Consider this minimization problem:

$$\begin{aligned} \min : & 8x_1 + 3x_2 + x_3 \\ & 2x_1 + 7x_2 - 5x_3 = 4 \\ & 4x_1 - 3x_2 + x_3 \geq 8 \\ & x_1 \geq 0 \\ & x_2 \leq 0 \\ & x_3 \text{ free} \end{aligned}$$

Write the corresponding canonical form.

Soluzione:

$$\begin{aligned} \min : & 8x_1 + 3x_2 + x_3 \\ & 2x_1 - 7x_2 - 5x_3^+ + 5x_3^- \geq 4 \\ & -2x_1 + 7x_2 + 5x_3^+ - 5x_3^- \geq -4 \\ & 4x_1 + 3x_2 + x_3^+ - x_3^- \geq 8 \\ & x_1, x_2, x_3^+, x_3^- \geq 0 \end{aligned}$$

Le variabili free son state sdoppiate, $x_3 \Rightarrow x_3^+, x_3^-$ e sostituite nel sistema, la x_3^+ ha mantenuto sempre il suo segno, mentre la x_3^- aveva l'opposto della sua controparte positiva.

Nel primo vincolo avevamo un'uguaglianza, abbiamo quindi dovuto porre il segno a \geq e aggiungere una nuova riga, con gli stessi termini moltiplicati per -1 e segno \leq .

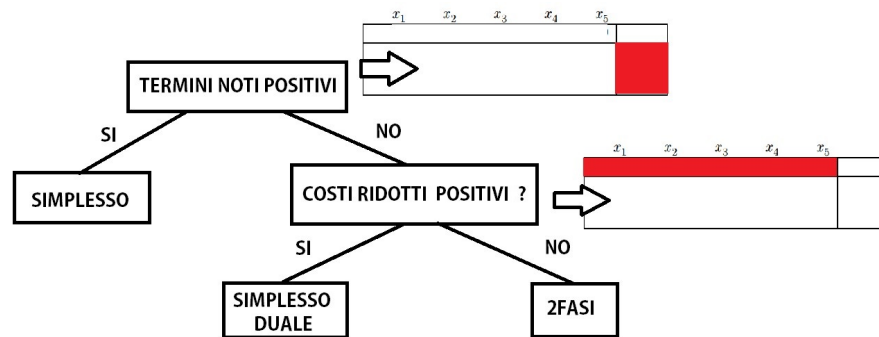
3 Matrici : LP, ILP

3.1 Fondamenti Concettuali

3.1.1 Quale Simpleso?

Quale simpleso dovrei usare se il professore non lo specifica ?

Abbiamo 3 strumenti, che non sono equivalenti (quasi), vediamo la mappa concettuale:



Questo sarebbe il flusso logico per approcciare il problema, in realtà il 2 fasi può sempre sostituire il simpleso duale, non è vero il contrario, in ogni caso il 2 fasi essendo più lungo è sconsigliato usarlo sempre. NB: la condizione di scelta del primo ramo "no" (costi ridotti negativi), non è discriminante della radice (termini noti positivi), ovvero, il simpleso standard, può essere usato se i suoi costi ridotti sono negativi, è condizione necessaria che abbia i termini noti positivi.

3.2 LP

Linear Programming, in italiano Programmazione Lineare PL

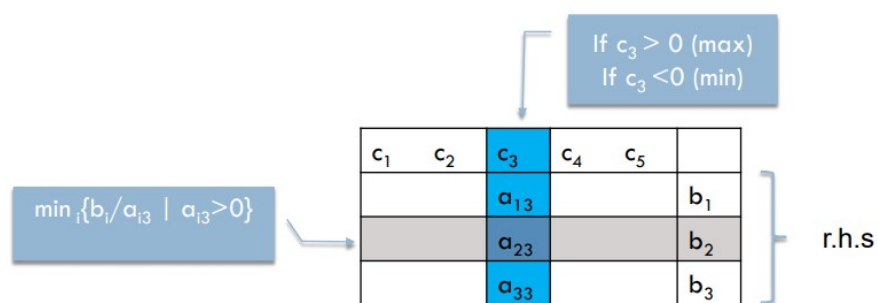
3.2.1 Simpleso

Una volta che abbiamo sviluppato il nostro problema e sistemato i vincoli, trovandoci nella condizione di poterlo sfruttare avendo i termini noti positivi una volta costruito il tableau, le iterazioni per selezionare il pivot saranno le seguenti:

- Siamo in un problema di *Massimo*: scelgo le colonne che hanno valori *positivi*, non do precedenza a nessuna colonna specifica se ne incontro più di una *positiva*, prendo sempre quella più a sinistra, con indice minore (legge di Bland).

- Siamo in un problema di *Minimo*: scelgo le colonne che hanno valori *negativi*, non do precedenza a nessuna colonna specifica se ne incontro più di una *negativa*, prendo sempre quella più a sinistra, con indice minore (legge di Bland).

Una volta designata la colonna, la riga viene scelta prendendo il valore *minore* dal rapporto dato da $\frac{b_i}{a_{ic}}$, dove i è l'indice di riga, c è l'indice di colonna e b è il termine noto della riga i che stiamo analizzando, come da esempio nell'immagine:



È importante notare che $a_{ic} > 0$, affinché il tutto funzioni. Iteriamo finché la riga della funzione obiettivo, ovvero la R_0 (o anche riga dei costi), non diventa tutta *positiva* se siamo in problema di *minimo*, o *negativa* se siamo in un problema di *massimo*.

3.2.2 Simpleso Duale

Il processo è molto simile al Simpleso standard, con una sostanziale differenza, ora non partiamo più a selezionare la colonna in base a che tipo di problema di ottimizzazione ci troviamo (*max* o *min*), ma prima partiamo a scegliere la riga, in base al fatto che il termine noto sia negativo (vale per entrambi i problemi):

0	...	0	...	0	\bar{c}_{m+1}	...	\bar{c}_h	...	\bar{c}_n	\bar{c}_0
1	...	0	...	0	$\bar{a}_{1,m+1}$...	$\bar{a}_{1,h}$...	$\bar{a}_{1,n}$	\bar{b}_1
...
0	...	1	...	0	$\bar{a}_{t,m+1}$...	$\bar{a}_{t,h}$...	$\bar{a}_{t,n}$	\bar{b}_t
...
0	...	0	...	1	$\bar{a}_{m,m+1}$...	$\bar{a}_{m,h}$...	$\bar{a}_{m,n}$	\bar{b}_n

Una volta scelta la riga, devo scegliere solo i valori di quella riga che sono *negativi*, affinché nel momento in cui farò diventare 1 quel pivot, il termine noto rispetto \bar{b}_t diventi anch'esso positivo, nel caso in cui non ci siano valori negativi nella riga, allora il duale è illimitato e ci fermiamo.

0	...	0	...	0	\bar{c}_{m+1}	...	\bar{c}_h	...	\bar{c}_n	\bar{c}_0
1	...	0	...	0	$\bar{a}_{1,m+1}$...	$\bar{a}_{1,h}$...	$\bar{a}_{1,n}$	\bar{b}_1
...
0	...	1	...	0	$\bar{a}_{t,m+1}$...	$\bar{a}_{t,h}$...	$\bar{a}_{t,n}$	\bar{b}_t
...
0	...	0	...	1	$\bar{a}_{m,m+1}$...	$\bar{a}_{m,h}$...	$\bar{a}_{m,n}$	\bar{b}_n

$\bar{b}_t < 0$

$\bar{a}_{t,h} < 0$

Nel caso avessi più di un valore negativo da cui scegliere, scelgo il pivot che ha un rapporto minore tra il valore della colonna, fratto il relativo pivot: $\frac{\bar{c}_j}{|\bar{a}_{t,j}|}$

$h = \operatorname{argmin} \left\{ \frac{\bar{c}_j}{|\bar{a}_{t,j}|} : \bar{a}_{t,j} < 0 \right\}$

0	...	0	...	0	\bar{c}_{m+1}	...	\bar{c}_h	...	\bar{c}_n	\bar{c}_0
1	...	0	...	0	$\bar{a}_{1,m+1}$...	$\bar{a}_{1,h}$...	$\bar{a}_{1,n}$	\bar{b}_1
...
0	...	1	...	0	$\bar{a}_{t,m+1}$...	$\bar{a}_{t,h}$...	$\bar{a}_{t,n}$	\bar{b}_t
...
0	...	0	...	1	$\bar{a}_{m,m+1}$...	$\bar{a}_{m,h}$...	$\bar{a}_{m,n}$	\bar{b}_n

$\bar{b}_t < 0$

Continuo ad iterare fintanto che i termini noti non sono tutti positivi.
 NB. È importante che la R_0 rimanga sempre positiva, perchè è condizione necessaria del sempliceo duale.

3.2.3 2Fasi

Il metodo delle due fasi può sostituire il sempliceo duale. Nel caso in cui noi avessimo dei termini noti negativi sarebbe sufficiente cambiare i segni dei vincoli, in modo da forzarli positivi, e bilanciare le slack negative con le variabili di surplus, il 2 fasi richiede in input un problema di *minimizzazione*, quindi nel caso in cui avessimo un problema di massimo, cambiamo i segni di quella riga per ricondurci all'input adeguato.

General Form	Standard Form
$\min z = 4x_1 + 3x_2$ s.t. $x_1 + x_2 \geq 6$ (I) $2x_1 + x_2 \geq 8$ (II) $x_1, x_2 \geq 0$	$\min z = 4x_1 + 3x_2 + 0s_1 + 0s_2$ s.t. $x_1 + x_2 - s_1 = 6$ (I) $2x_1 + x_2 - s_2 = 8$ (II) $x_1, x_2, s_1, s_2 \geq 0$

Dal nome, ci sono due fasi principali su cui lavorare, partiamo dalla prima fase, dobbiamo costruire un tableau, una volta ricavata la forma standard poniamo a 0 tutti gli elementi della R_0 , lasciando un 1 esclusivamente dove è presente una variabile di surplus. Le variabili di surplus vengono aggiunte solo se la slack di quel vincolo è negativa.

x_1	x_2	s_1	s_2	y_1	y_2	
4	3	0	0	M	M	0
1	1	-1	0	1	0	6
2	1	0	-1	0	1	8

→

x_1	x_2	s_1	s_2	y_1	y_2	
0	0	0	0	1	1	0
1	1	-1	0	1	0	6
2	1	0	-1	0	1	8

y_1
 y_2

Procediamo facendo sparire gli 1 delle rispettive variabili di surplus dalla R_0 , calcoliamo quindi $R'_0 = R_0 - 1 * R_1 - 1 * R_2$ (Si usano i costi ridotti, non i termini noti, e in questo caso sono sempre valorizzati a 1):

x_1	x_2	s_1	s_2	y_1	y_2	
0	0	0	0	1	1	0
1	1	-1	0	1	0	6
2	1	0	-1	0	1	8

→

x_1	x_2	s_1	s_2	y_1	y_2	
-3	-2	1	1	0	0	-14
1	1	-1	0	1	0	6
2	1	0	-1	0	1	8

$-w$ | RO'
 y_1 | $R1$
 y_2 | $R2$

Adesso procediamo iterando con il semplice standard, quindi, trovandoci in un problema di minimizzazione, dobbiamo far sparire le colonne che nella R_0 hanno valori *negativi*.

x_1	x_2	s_1	s_2	y_1	y_2	
-3	-2	1	1	0	0	-14
1	1	-1	0	1	0	6
2	1	0	-1	0	1	8

→

x_1	x_2	s_1	s_2	y_1	y_2	
0	-1/2	1	-1/2	0	3/2	-2
0	1/2	-1	1/2	1	-1/2	2
1	1/2	0	-1/2	0	1/2	4

$-w$ | RO''
 y_1 | $R1'$
 x_2 | $R2'$

x_1	x_2	s_1	s_2	y_1	y_2	
0	0	0	0	1	1	0
0	1	-2	1	2	-1	4
1	0	1	-1	-1	1	2

→

x_1	x_2	s_1	s_2	y_1	y_2	
0	-1/2	1	-1/2	0	3/2	-2
0	1/2	-1	1/2	1	-1/2	2
1	1/2	0	-1/2	0	1/2	4

$-w$ | RO'''
 x_2 | $R1''$
 x_1 | $R2''$

Iniziamo ora la Seconda Fase, prendiamo la matrice precedente che ora si trova la R_0 piena di 0 ad esclusione delle colonne delle variabili di surplus, rimuoviamo quindi queste colonne e sostituiamo la R_0 con le variabili della funzione obbiettivo del problema originale. NB. Se il problema era di massimizzazione originariamente, dobbiamo riportarlo, nel caso fosse di minimizzazione possiamo lasciarlo così com'è.

x_1	x_2	s_1	s_2	y_1	y_2	
0	0	0	0	1	1	0
0	1	-2	1	2	-1	4
1	0	1	-1	-1	1	2

→

x_1	x_2	s_1	s_2			
4	3	0	0	0	0	-z
0	1	-2	1	4	0	x_2
1	0	1	-1	2	0	x_1

$-z$ | RO'''
 x_2 | $R1''$
 x_1 | $R2''$

Procediamo infine ad avere gli 0 nelle rispettive colonne della nostra base, in modo da sviluppare la soluzione, quindi tramite operazioni matriciali sistemiamo la R_0 .

$$R_0''' = R_0'' - 3 * R_1'' - 4 * R_2'' \text{ (Si usano i costi ridotti, non i termini noti)}$$

x_1	x_2	s_1	s_2	
4	3	0	0	0
0	1	-2	1	4
1	0	1	-1	2

-z	R0'''
x_2	R1''
x_1	R2''

➔

x_1	x_2	s_1	s_2	
0	0	2	1	-20
0	1	-2	1	4
1	0	1	-1	2

-z	R0''''
x_2	R1''
x_1	R2''

3.2.4 Duale del problema

Come trasformiamo il nostro sistema in un problema duale ?

Per prima cosa dobbiamo scegliere una direzione per i segni, tutti devono andare nella stessa direzione (non va considerata la riga finale delle $x_i \geq 0$), possiamo seguire questa tabella per indicazioni più rigorose:

Canonic Form	
Primal	Dual
$\min c^T x$	$\max u^T b$
$Ax \geq b$	$u^T A \leq c^T$
$x \geq 0$	$u^T \geq 0$

Standard Form	
Primal	Dual
$\min c^T x$	$\max u^T b$
$Ax = b$	$u^T A \leq c^T$
$x \geq 0$	$u^T \text{ free}$

- se i vincoli del primale sono: $a_i^T x \geq b_i$, nel duale arriveremo dal basso della regione ammissibile, le slack di conseguenza saranno **non negative**. Nel duale staremo massimizzando, per cui le nostre corrispettive variabili (moltiplicative) dovranno essere **non negative**.
- se i vincoli del primale sono $a_i^T x \leq b_i$, le slack dei vincoli dovranno essere **non positive**. Nel duale staremo massimizzando, per cui le nostre corrispettive variabili (moltiplicative) dovranno essere **non positive**.
- se i vincoli del primale sono $a_i^T x = b_i$, le slack dei vincoli potranno avere qualsiasi valore. Le corrispondenti variabili (moltiplicative) potranno avere qualsiasi valore.

Esempio:

$$\begin{array}{ll}
\min : 2x_1 + 3x_2 + x_3 & \min : 2x_1 + 3x_2 + x_3 \\
- x_1 + 3x_2 - 2x_3 \geq 8 & \Rightarrow - x_1 + 3x_2 - 2x_3 \geq 8 \\
x_2 - x_3 \leq 2 & - x_2 + x_3 \geq -2 \\
x_1, x_2, x_3 \geq 0 & x_1, x_2, x_3 \geq 0
\end{array}$$

Possiamo sempre ricondurci alla forma canonica o standard prima di scrivere il duale:

Min	Max
$a_i^T x \geq b_i$	$u^T \geq 0$
$a_i^T x \leq b_i$	$u^T \leq 0$
$a_i^T x = b_i$	u^T free
$x_j \geq 0$	$u^T A_j \leq c_j$
$x_j \leq 0$	$u^T A_j \geq c_j$
x_j free	$u^T A_j = c_j$

Ordiniamo il sistema per una maggiore chiarezza:

$$\begin{array}{|c|c|c|}
\hline
2x_1 & + 3x_2 & + x_3 \\
\hline
- x_1 & + 3x_2 & - 2x_3 \\
+ 0 & - x_2 & + x_3 \\
\hline
\end{array}
\begin{array}{l}
\geq 8 \\
\geq -2
\end{array}$$

Possiamo dividerlo in sotto vettori colonna, prendendo il vettore colonna delle x_1 , partiamo dal vettore colonna dei vincoli, evidenziata di rosso, ribaltiamo di 90 gradi a sinistra e sostituiamo ogni x con u_i dove i rappresenta la riga del vettore colonna, quindi avremo:

$$-u_1 + 0u_2 \Rightarrow -u_1$$

mettiamo ora la disequazione, che rappresenta il verso opposto a quello scelto precedentemente, quindi avevamo impostato il sistema tutto a: \geq , avremo:

$$-u_1 \leq$$

A destra della disequazione inseriamo il vettore colonna arancione, ovvero il valore della rispettiva x_1 della funzione obiettivo:

$$-u_1 \leq 2$$

Reiteriamo per ogni colonna, il vettore colonna azzurro, ovvero il vettore dei termini noti, sar  la nostra nuova funzione obiettivo, usiamo lo stesso procedimento di prima, quindi avremo $8u_1 - 2u_2$, il problema verr  anche invertito

quindi dal *minimo* di partenza, ora avremo un *massimo*:

$$\begin{aligned}
 \max : & 8u_1 - 2u_2 \\
 & -u_1 \leq 2 \\
 & 3u_1 - u_2 \leq 3 \\
 & -2u_1 + u_2 \leq 1 \\
 & u_1, u_2 \geq 0
 \end{aligned}$$

$$\begin{array}{l}
 \boxed{8u_1 - 2u_2} \\
 \boxed{-u_1} \leq \boxed{2} \quad X1 \\
 \boxed{3u_1 - u_2} \leq \boxed{3} \quad X2 \\
 \boxed{-2u_1 + u_2} \leq \boxed{1} \quad X3
 \end{array}$$

Osservando l'ultima immagine per chiarezza col risultato finale che si riconduce al sistema iniziale.

Se volessi risolvere il sistema invece, partendo dal sistema già orientato nel verso giusto e il suo problema duale con soluzione $x = (15, 0, 20, 0, 0)$ $z_P = -30$:

$$\begin{array}{ll}
 \min : 2x_1 + 4x_2 - 3x_3 & \max : 5u_1 - 10u_2 \\
 x_1 - 3x_2 - x_3 \geq -5 & u_1 - u_2 \leq 2 \\
 -2x_1 - 2x_2 + x_3 \geq -10 & -3u_1 - 2u_2 \leq 4 \\
 x_1, x_2, x_3 \geq 0 & -u_1 + u_2 \leq -3 \\
 & u_1, u_2 \geq 0
 \end{array}$$

$$\left\{ \begin{array}{l}
 (x_1 - 3x_2 - x_3 + 5)u_1 = 0 \\
 (-2x_1 - 2x_2 - x_3 + 10)u_2 = 0 \\
 (u_1 - 2u_2 - 2)x_1 = 0 \\
 (-3u_1 - 2u_2 - 4)x_2 = 0 \\
 (-u_1 + u_2 + 3)x_3 = 0
 \end{array} \right. \Rightarrow \left\{ \begin{array}{l}
 (0)u_1 = 0 \\
 (0)u_2 = 0 \\
 u_1 - 2u_2 = 2 \\
 (-3u_1 - 2u_2 - 2)0 = 0 \\
 (-u_1 + u_2) = -3
 \end{array} \right.$$

$$u = (4, 1)z_D = -30$$

- Moltiplico ogni vincolo del primale per la sua variabile corrispondente del duale e lo pongo uguale a zero, prima porto a sinistra il termine noto.
- Moltiplico ogni vincolo del duale per la sua variabile corrispondente del primale e lo pongo uguale a zero, prima porto a sinistra il termine noto.
- Metto tutto a sistema e semplifico, ad esempio sappiamo i valori delle variabili della funzione obiettivo del duale, inoltre i vincoli che ci portano alla soluzione ottima saranno uguale a zero e i restanti diversi da 0.
- Ottengo un sistema che posso risolvere, e dovrebbero essere presenti solo variabili del problema primale, risolvo il sistema e ottengo i valori delle variabili della funzione obiettivo del problema primale

Nell'esempio del sistema sopra, stringi stringi abbiamo fatto:

Poniamo a 0 le x dentro le parentesi, abbiamo un sistema di n equazioni in m incognite, poniamo uguali a 0 a scelta $n - m$, noi scegliamo quindi di annullare fuori dalle parentesi tra x_1, x_2, x_3 , $x_2 = 0$ per semplificarci i calcoli (si veda la 4 riga del sistema risolto sopra), ci rimane in questo caso un sistema a due equazioni:

$$u_1 - 2u_2 = 2$$

$$(u_1 + u_2) = -1$$

Lo risolviamo $u_1 = 4, u_2 = 1$ avremo per cui sostituendo le relative u nella funzione obiettivo del duale $5u_1 - 10u_2, z = -30$.

3.3 Rappresentazione Grafica

Anche tu sei a digiuno di grafici ed hai un incontrollato appetito di disegnarne altri ?

Molto semplicemente partendo da un sistema, si calcola la derivata prima della funzione obiettivo per avere il nostro bel gradiente. Ogni vincolo è una retta nel grafico e il segno è l'area accettabile dove guardare, un esempio semplice di un problema duale:

$$\begin{aligned} \max : & 8u_1 - 2u_2 \\ & -u_1 \leq 2 \\ & 3u_1 - u_2 \leq 3 \\ & -2u_1 + u_2 \leq 1 \\ & u_1, u_2 \geq 0 \end{aligned}$$

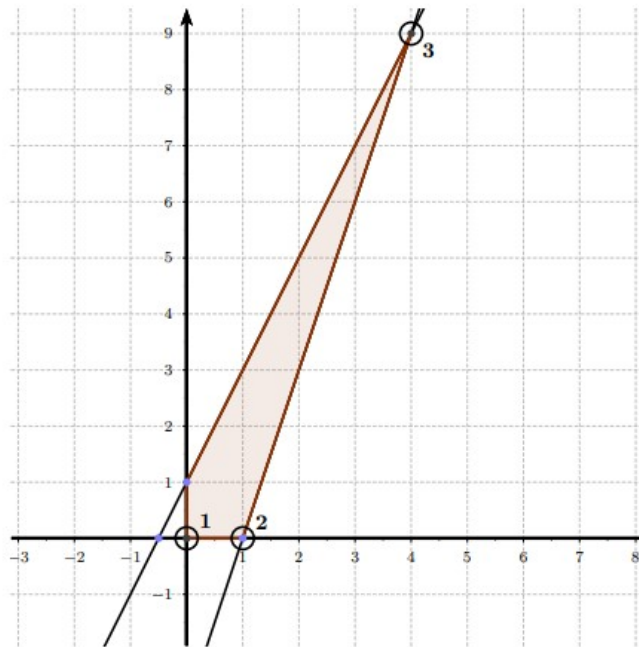
Possiamo interpretare per facilità gli u_1 come x e gli u_2 come y , avremo un gradiente : $\nabla = (8, -2)$.

prendiamo il primo vincolo: $-u_1 \leq 2 \Rightarrow -x \leq 2$

Quindi ponendo $x = 0$ non salta fuori nulla.

Il secondo vincolo $3x - y \leq 3$, ponendo $x = 0$ ricavo il primo punto $y = -3$ con \geq avendo cambiato il segno, come secondo punto per tracciare la retta scelgo $x = 1$, avrò $y = 0$ sempre con \geq , traccio la retta nel grafico.

Ultimo vincolo $-2x + y \leq 1$, pongo $x = 0$ avrò $y = 1$ con l'area accettabile sotto di essa ≤ 1 , secondo punto pongo $x = -\frac{1}{2}$ avrò $y = 0$, il grafico risultante qui sotto, ha l'area interna ammissibile, che si trova nel primo quadrante del grafico, avendo come vincolo ulteriore $u_1, u_2 \geq 0$, dal gradiente che cade sul punto 8 in questo caso, la soluzione ottimale è lo spigolo in alto a destra con coordinate $(4, 9)$ che corrisponde alla soluzione $z = 30$ andando a sostituire le coordinate nella funzione obiettivo di max del problema.



3.4 ILP

Integer Linear Programming, in italiano Programmazione Lineare Intera PLI

3.4.1 Tagli di Gomory

I tagli di gomory si usano quando il nostro problema da PL Lineare è PLI, ovvero Lineare Intero, il principio è che, non avendo numeri con virgole, posso approssimare il risultato all'intero più vicino.

Partendo dal tableau già risolto e con risultato ottimale e avendo termini fratti nella R_0 , posso sfruttare Gomory per trovare una soluzione PLI, la formula da usare è:

$$\sum_{j \in F} (\bar{a}_{tj} - \lfloor \bar{a}_{tj} \rfloor) x_j \geq \bar{b}_t - \lfloor \bar{b}_t \rfloor$$

NB. Le sbarre non sono valore assoluto, ma significano che approssimo difetto.

Dovrò aggiungere una riga e una colonna alla fine del mio tableau, sfruttando la relativa formula, vediamo un esempio:

x_1	x_2	S_1	S_2	
0	0	1/4	1/4	3/2
1	0	1/6	-1/6	1
0	1	1/4	1/4	3/2

- Prendiamo una riga che abbia un termine noto frazionario, in questo caso R_2
- Scriviamo : $x_2 + \frac{1}{4}S_1 + \frac{1}{4}S_2 = \frac{3}{2}$
- usiamo al formula di Gomory: $(\frac{1}{4} - \lfloor \frac{1}{4} \rfloor)S_1 + (\frac{1}{4} - \lfloor \frac{1}{4} \rfloor)S_2 \geq (\frac{3}{2} - \lfloor \frac{3}{2} \rfloor)$
- effettuiamo gli arrotondamenti: $(\frac{1}{4} - 0)S_1 + (\frac{1}{4} - 0)S_2 \geq (\frac{3}{2} - 1)$
- arriviamo a: $\frac{1}{4}S_1 + \frac{1}{4}S_2 \geq \frac{1}{2}$

Possiamo ora inserire la riga nel tableau, aggiungendo anche una colonna di surplus per bilanciare:

$$\frac{1}{4}S_1 + \frac{1}{4}S_2 - S_3 = \frac{1}{2}$$


NB. la Slack S_3 è negativa perchè il segno era \geq , fosse stato \leq sarebbe stata positiva.

Moltiplichiamo per -1 la riga per avere una base accettabile:

x_1	x_2	s_1	s_2	s_3	
0	0	1/4	1/4	0	3/2
1	0	1/6	-1/6	0	1
0	1	1/4	1/4	0	3/2
0	0	-1/4	-1/4	1	-1/2

La seguente soluzione non è accettabile per il costo negativo dei termini noti ($-\frac{1}{2}$), dalla sezione del Compendio 2.1.1 sappiamo che dobbiamo usare il semplice Duale, iteriamo di conseguenza:

x_1	x_2	s_1	s_2	s_3	
0	0	1/4	1/4	0	3/2
1	0	1/6	-1/6	0	1
0	1	1/4	1/4	0	3/2
0	0	-1/4	-1/4	1	-1/2



x_1	x_2	s_1	s_2	s_3	
0	0	0	0	1	1
1	0	0	-1/3	2/3	2/3
0	1	0	0	1	1
0	0	1	1	-4	2


Ora la base è accettabile, abbiamo però un nuovo valore frazionario nei termini noti, R_1 , possiamo iterare di nuovo con gomory seguendo il procedimento precedente:

$(-\frac{1}{3} - \lfloor -\frac{1}{3} \rfloor)S_2 + (\frac{2}{3} - \lfloor \frac{2}{3} \rfloor)S_3 \geq (\frac{2}{3} - \lfloor \frac{2}{3} \rfloor) \Rightarrow (\frac{1}{3} + 1)S_2 + (\frac{2}{3} - 0)S_3 \geq (\frac{2}{3} - 0)$
 NB. L'arrotondamento di S_2 è stato di +1, i termini negativi, per difetto, si avvicinano a $-\infty$, di conseguenza $-\frac{1}{3} \Rightarrow -0,33 \Rightarrow -1$!

Avremo: $\frac{2}{3}S_2 + \frac{2}{3}S_3 - S_4 = \frac{2}{3}$

Aggiungiamo la riga nel tableau moltiplicandola prima per -1 come prima:

x_1	x_2	s_1	s_2	s_3	s_4	
0	0	0	0	1	0	1
1	0	0	-1/3	2/3	0	2/3
0	1	0	0	1	0	1
0	0	1	1	-4	0	2
0	0	0	2/3	2/3	-1	2/3



x_1	x_2	s_1	s_2	s_3	s_4	
0	0	0	0	1	0	1
1	0	0	-1/3	2/3	0	2/3
0	1	0	0	1	0	1
0	0	1	1	-4	0	2
0	0	0	-2/3	-2/3	1	-2/3

E risolviamo con il semplice duale vista la situazione, qua il risultato finale:

x_1	x_2	s_1	s_2	s_3	s_4	
0	0	0	0	1	0	1
1	0	0	0	1	-1/2	1
0	1	0	0	1	0	1
0	0	1	0	-3	3/2	1
0	0	0	1	1	-3/2	1

La soluzione infine è: $(x|s)^* = (1, 1, 1, 1, 0, 0)$, $z^* = -1$

3.5 Esercizi Particolari

3.5.1 Simplexso con variabili free

Esame 2018-07-19 Ex 2

Consider the following PLC problem. Solve it with the simplex method, using the Bland's rule. Verify the solution by solving the problem with the graphic method.

$$\begin{aligned} \max : & x_1 - 2x_2 \\ & x_1 - x_2 \leq 1 \\ & -2x_1 + x_2 \leq 1 \\ & x_1, x_2 \text{ free} \end{aligned}$$

Soluzione:

Prima di partire notiamo che le x sono free, questo comporta una particolarità, dobbiamo vincolarle ≥ 0 , scriveremo :

$$\begin{aligned} \max : & x_1^+ - x_1^- - 2x_2^+ + 2x_2^- \\ & x_1^+ - x_1^- - x_2^+ + x_2^- \leq 1 \\ & -2x_1^+ + 2x_1^- + x_2^+ - x_2^- \leq 1 \\ & x_1^+, x_1^-, x_2^+, x_2^- \geq 0 \end{aligned}$$

costruiamo il relativo tableau e risolviamolo con il simplexso standard:

x_1	x_2	x_3	x_4	x_5	x_6		
1	-1	-2	2	0	0	0	$-z$
①	-1	-1	1	1	0	1	x_5
-2	2	1	-1	0	1	1	x_6

x_1	x_2	x_3	x_4	x_5	x_6		
0	0	-1	1	-1	0	-1	$-z$
1	-1	-1	①	1	0	1	x_1
0	0	-1	1	2	1	3	x_6

x_1	x_2	x_3	x_4	x_5	x_6		
-1	1	0	0	-2	0	-2	$-z$
1	-1	-1	1	1	0	1	x_4
-1	①	0	0	1	1	2	x_6

x_1	x_2	x_3	x_4	x_5	x_6		
0	0	0	0	-3	-1	-4	$-z$
0	0	-1	1	2	1	3	x_4
-1	1	0	0	1	1	2	x_2

NB. Perché nella prima matrice prendo come pivot 1 invece di -2 ?
 Ricordando dalla sezione del Compendio 2.1.2, il valore del pivot deve essere > 0 , -2 non soddisfa questo requisito.

Arrivati alla fine, come soluzione avremo le colonne x_4 che rappresenta x_2^- e la colonna x_2 che rappresenta x_1^- , avremo perciò:

$$x_1^- = 2 \Rightarrow x_1 = -2$$

$$x_2^- = 3 \Rightarrow x_2 = -3$$

$$z = 4$$

cambiamo semplicemente i segni essendo che sono le x^-

3.5.2 Problema PLC con simplesso e gomory in salsa teriyaki

In questo esercizio ci sta la particolarità che il sistema viene già impostato a = senza disequazioni, non si sa perchè, ma sembra funzionare, si usa il due fasi e si aggiungono alla cazzo di cane le colonne di surplus perchè sì, come punto aggiuntivo ci sta anche da usare i tagli di Gomory:

Esame 2017-02-17 Ex 2

Consider the following PLC problem. Solve it with the simplex method.

$$\begin{aligned} \min : & x_1 + 2x_2 + 1x_3 \\ & 2x_1 + 3x_2 - 2x_3 = 20 \\ & x_1 - 2x_2 + 3x_3 = 12 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Add the integrality constraints for the variables x and apply the cutting plane method with Gomory's cuts to look for an integer solution. Always choose the first row available to generate a cut. Stop the algorithm after adding at most two cuts

FASE I

x_1	x_2	x_3	x_1^a	x_2^a		
-3	-1	-1	0	0	-32	$-\xi$
(2)	3	-2	1	0	20	x_1^a
1	-2	3	0	1	12	x_2^a

x_1	x_2	x_3	x_1^a	x_2^a		
0	$\frac{7}{2}$	-4	$\frac{3}{2}$	0	-2	$-\xi$
1	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	10	x_1
0	$-\frac{7}{2}$	(4)	$-\frac{1}{2}$	1	2	x_2^a

x_1	x_2	x_3	x_1^a	x_2^a		
0	0	0	1	1	0	$-\xi$
1	$\frac{3}{8}$	0	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{21}{2}$	x_1
0	$-\frac{7}{8}$	1	$-\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	x_3

FASE II

x_1	x_2	x_3		
0	$\frac{9}{4}$	0	-11	$-z$
1	$\frac{3}{8}$	0	$\frac{21}{2}$	x_1
0	$-\frac{7}{8}$	1	$\frac{1}{2}$	x_3

$x = (\frac{21}{2}, 0, \frac{1}{2})$, $z_P = 11$. Gomory's cut: $\frac{5}{8}x_2 \geq \frac{1}{2}$

x_1	x_2	x_3	x_4		
0	$\frac{9}{4}$	0	0	-11	$-z$
1	$\frac{3}{8}$	0	0	$\frac{21}{2}$	x_1
0	$-\frac{7}{8}$	1	0	$\frac{1}{2}$	x_3
0	(5)	0	1	$-\frac{1}{2}$	x_4

x_1	x_2	x_3	x_4		
0	0	0	$\frac{18}{5}$	$-\frac{64}{5}$	$-z$
1	0	0	1	10	x_1
0	0	1	$-\frac{7}{5}$	$\frac{6}{5}$	x_3
0	1	0	$-\frac{8}{5}$	$\frac{4}{5}$	x_2

$x = (10, \frac{4}{5}, \frac{6}{5}, 0)$, $z_P = \frac{64}{5}$ Gomory's cut: $\frac{3}{5}x_2 \geq \frac{1}{5}$

x_1	x_2	x_3	x_4	x_5		
0	0	0	$\frac{18}{5}$	0	$-\frac{64}{5}$	$-z$
1	0	0	1	0	10	x_1
0	0	1	$-\frac{7}{5}$	0	$\frac{6}{5}$	x_3
0	1	0	$-\frac{8}{5}$	0	$\frac{4}{5}$	x_2
0	0	0	$-\frac{3}{5}$	1	$-\frac{1}{5}$	x_5

x_1	x_2	x_3	x_4	x_5		
0	0	0	0	6	14	$-z$
1	0	0	0	$\frac{2}{3}$	$\frac{29}{3}$	x_1
0	0	1	0	$-\frac{1}{3}$	$\frac{5}{3}$	x_3
0	1	0	0	$-\frac{1}{3}$	$\frac{4}{3}$	x_2
0	0	0	1	$-\frac{1}{3}$	$\frac{1}{3}$	x_4

$$x = \left(\frac{29}{3}, \frac{5}{3}, \frac{8}{3}, \frac{1}{3}, 0\right), z_P = \frac{64}{5}$$

3.5.3 Sensitivity Analysis

Esame 2020-02-16 Ex 2

Consider the following LP problem:

$$\begin{aligned} \max : & -5x_1 + 5x_2 + 13x_3 \\ & -x_1 + x_2 + 3x_3 + x_4 = 20 \\ & 12x_1 + 4x_2 + 10x_3 + x_5 = 90 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

The optimal solution is $x = (0, 20, 0, 0, 10)$ with value 100

Given that $B^{-1} = \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}$, answer the following questions. Provide motivations to support your arguments:

- what happens to the optimal base if b_1 , the right hand side of the first constraint, becomes 24?
- what happens to the optimal base if c_3 , the coefficient of variable x_3 becomes 10?

Soluzione:

Dalle slide del prof sulla sensitivity analysis pagina 7:

- We are considering in general terms the variation of a right hand side:
- $b \rightarrow b + \Delta b$
- The new basic solution changes value:
- $x_B = B^{-1}(b + \Delta b)$
- The solution is a basic feasible solution if:
- $x_B = B^{-1}(b + \Delta b) \geq 0$
- This is what we need to check in order to know if the right hand side variations let the basis feasible:
- $B^{-1}b + B^{-1}\Delta b \geq 0$, this is a new system where Δb are the new variables.
- $\begin{bmatrix} 20 \\ 90 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix} * \begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix} \geq 0 \Rightarrow \begin{cases} 1\Delta b_1 + 0\Delta b_2 \geq -20 \\ -4\Delta b_1 + 1\Delta b_2 \geq -90 \end{cases}$
- dove: $\begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix}$ sono la differenza dei termini noti precedenti, quindi aumentando 20 a 24: $\begin{bmatrix} 4 \\ 0 \end{bmatrix}$
- We study what happens if only one term changes: if $\Delta b_2 = 0$ and only b_1 changes:
- $\begin{cases} 1\Delta b_1 \geq -20 \\ -4\Delta b_1 \geq -90 \end{cases} \Rightarrow -20 \leq \Delta b_1 \leq \frac{90}{4}$ because originally $b_1 = -20$.
- Fixed the other right hand sides, if b_1 lays between -20 and 22.5 the basis does not change, outside of this range the basis changes, so the new basis $b_1 = 24$ is outside this range, and the new solution will be different.

Per la seconda domanda, x_3 non è nelle soluzioni ottimali, dal testo : $x = (0, 20, 0, 0, 10)$, se poi l'abbassiamo di valore da 13 a 10 diventa pure meno appetibile per l'ottimizzazione, la soluzione quindi non cambia.

4 Grafi

4.1 GT

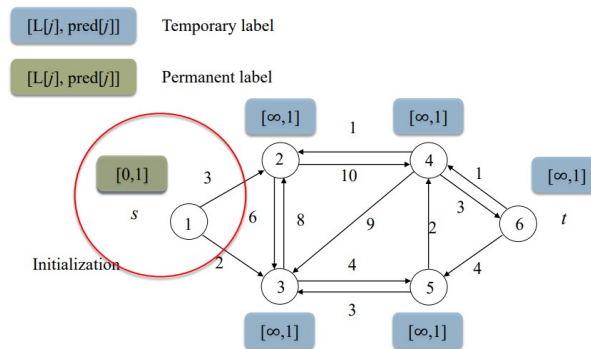
Graph Theory

4.1.1 Dijkstra

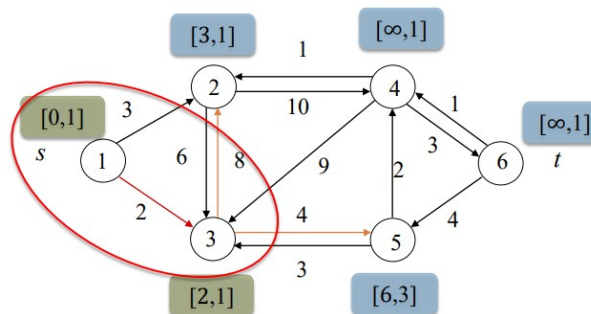
Operiamo sempre con grafi dai costi positivi.

Il principio è abbastanza semplice, ogni nodo ha due indici $L[j]$ che viene pre-impostato a 0 per il nodo di partenza ed a ∞ e rappresenta il costo della strada più economica da $s \Rightarrow j$ mentre $pred[j]$ rappresenta il nodo precedente da cui siamo arrivati che ha la strada più economica.

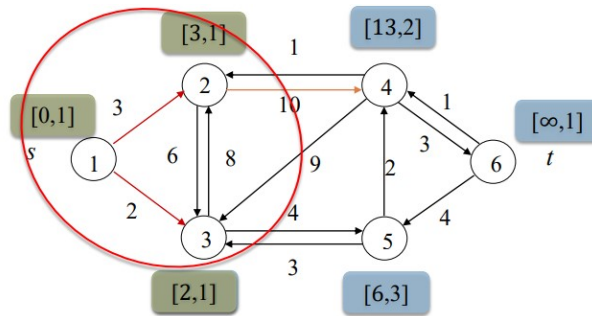
Avremo delle etichette per ogni nodo, che hanno questi due indici e passano da temporanei a permanenti, l'esplorazione da nodo a nodo procede iterativamente selezionando sempre l'arco dal costo minore di tutti i nodi che ho raggiunto in quella fase:



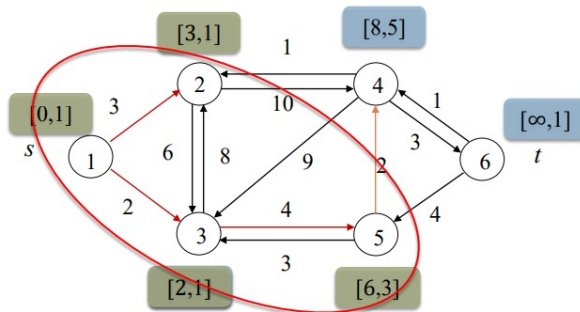
Etichettiamo temporaneamente i nodi che sono raggiunti da S , ci spostiamo al nodo che costa meno da S (nodo 3 in questo caso) e approdiamo al nodo 3, essendo quello che costa meno tra i nodi già raggiunti, rendendo permanente l'etichetta perchè è oggettivamente la strada più economica, aggiungiamo anche altre etichette temporanee ai nodi che ora sono accessibili dal nodo 3 appena esplorato, ovvero il 5.



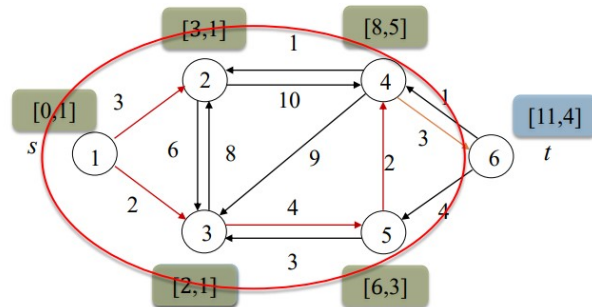
Rendiamo permanente anche l'etichetta del nodo 2 non avendo una strada più economica di quella trovata precedentemente ($1 \Rightarrow 2$). Aggiungiamo un'altra etichetta temporanea al nodo ora visibile dopo l'esplorazione del 2, ovvero il 4.

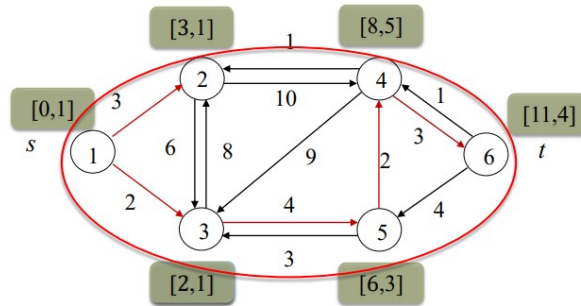


Mi sposto ora sul nodo 5 essendo quello che è raggiunto da un arco che ha costo minore tra quelli disponibili e raggiunti fino ad ora, rendo l'etichetta permanente e aggiorno anche l'etichetta temporanea del nodo 4, avendo trovato ora una strada più economica di quella precedente passante per l'attuale nodo 5.



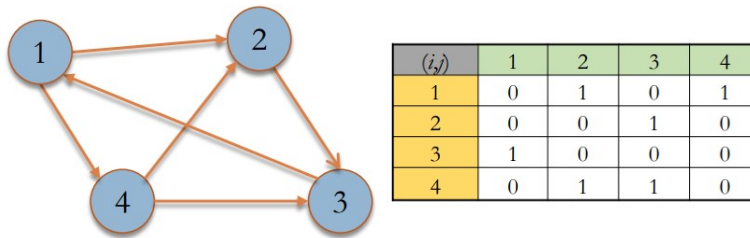
Ripetiamo il ragionamento per gli ultimi due veloci passaggi:





4.1.2 Shortest Path Tree

Un grafo orientato che possiamo rappresentare tramite matrice di adiacenza, può avere anche costi negativi.



Posso usare anche una lista di predecessori per rappresentarlo:

$$S_1 = \{2, 4\}$$

$$S_2 = \{3\}$$

$$S_3 = \{1\}$$

$$S_4 = \{2, 3\}$$

In forma Tabulare con due matrici, una per i costi ed una per il nodo precedente, in questo caso sotto, C_{ij} è matrice di adiacenza, mentre $L_{[j]}$ matrice dei pesi e $pred_{[j]}$ matrice dei nodi precedenti sincronizzata coi relativi costi:

$$C_{ij} =$$

	1	2	3	4	5	6	7	8
1		3					5	1
2			4				3	
3		4		1			5	
4			1		3	2	10	
5				3		2		5
6					2			1
7		3	5					0
8					5	1	0	

S	L[i]							Pred[i]						
	2	3	4	5	6	7	8	2	3	4	5	6	7	8
{1}	3	inf	inf	inf	inf	5	1	1	1	1	1	1	1	1
{1,8}	3	inf	inf	6	2	1	1	1	1	1	8	8	8	1
{1,8,7}	3	6	inf	6	2	1	1	1	7	1	8	8	8	1
{1,8,7,6}	3	6	inf	4	2	1	1	1	7	1	6	8	8	1
{1,8,7,6,2}	3	6	inf	4	2	1	1	1	7	1	6	8	8	1
{1,8,7,6,2,5}	3	6	7	4	2	1	1	1	7	5	6	8	8	1
{1,8,7,6,2,5,3}	3	6	7	4	2	1	1	1	7	5	6	8	8	1
{1,8,7,6,2,5,3,4}	3	6	7	4	2	1	1	1	7	5	6	8	8	1

4.2 SST

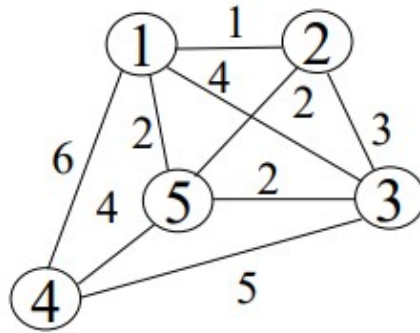
Shortest Spanning Tree

4.2.1 Cenni di Base

- **Path**(strada): Una sequenza di archi che collega due o più nodi, due nodi sono connessi se esiste almeno una path tra loro, una **simple path** è invece una strada senza ripetizione di archi
- **Connected Graph**: se in un grafo tutti i nodi sono raggiungibili da una strada si dice che è connesso.
- **Directed Graph**(grafo orientato): grafo che ha gli archi con frecce, ovvero indicando la direzione, la strada non è bidirezionale.
- **Directed path**: una sequenza consecutiva di archi che vanno dal nodo $A \Rightarrow B$.
- **Cycl Circuit**(ciclo): Se una strada di un grafo non orientato torna ad un nodo già visitato crea un ciclo, nel caso di un grafo orientato si parla invece di circuito.
- **Complete graph**: un grafo nel quale tutti i suoi nodi sono collegati tra di loro, ovvero da ogni nodo è possibile raggiungere con un solo arco gli altri nodi.

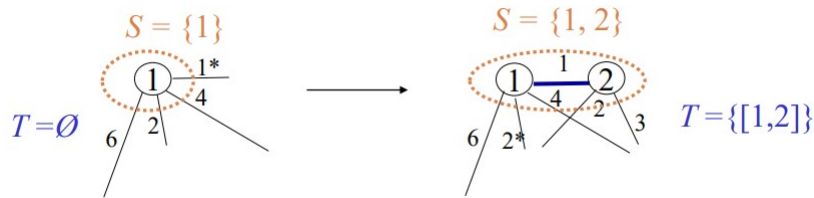
4.2.2 SST Prim's

L'algoritmo di prim's è utile per ricavare un SST da un grafo non orientato:

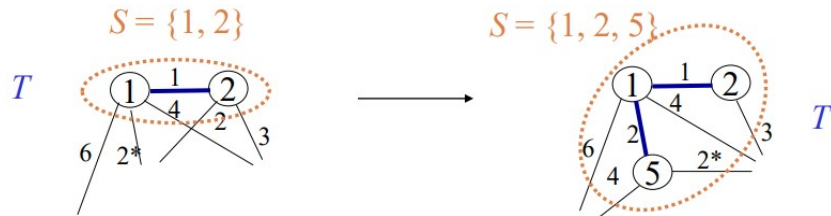


Partendo dal nodo 1 che sarà usata come radice del nostro SST, costruiamo la soluzione $S = \{1\}$.

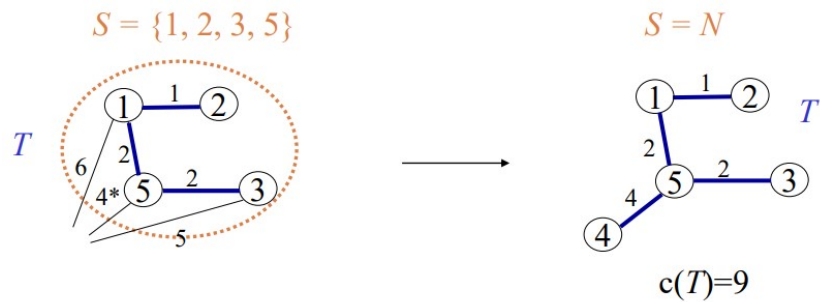
Marco con un * l'arco che costa meno, in questo caso quello che porta al nodo 2, e lo collego segnandomi la soluzione in $S = \{1, 2\}$, segno l'arco successivo tra i nodi che ho raggiunto, in questo caso l'arco dal costo 2, che si collega al nodo 5.



Collego i nodi $1 \Rightarrow 5$ con l'arco dal costo 2 che è il minore di costo tra quelli disponibili e aggiorno la soluzione $S = \{1, 2, 5\}$, marco già l'arco successivo che ha costo 2.



Collego i nodi $5 \Rightarrow 3$, aggiorno la soluzione $S = \{1, 2, 5, 3\}$, marco il nodo con costo 4, ed infine lo collego nell'ultimo passaggio $5 \Rightarrow 4$, abbiamo collegato tutti i nodi, ci fermiamo e calcoliamo il costo totale che è $c(T) = 9$.



4.2.3 SST Da Matrice trovare la soluzione ottimale

Esame 2020-06-25 Ex 2

Given the following cost matrix representing an undirected graph with 7 nodes, find the Shortes Spanning Tree. Provide the edges in the optimal solution and its value

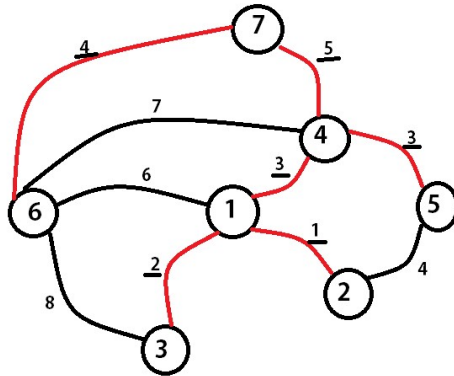
c_{ij}	1	2	3	4	5	6	7
1	-	1	2	3	-	6	-
2	-	-	-	-	4	-	-
3	-	-	-	-	-	8	-
4	-	-	-	-	3	7	5
5	-	-	-	-	-	-	-
6	-	-	-	-	-	-	4
7	-	-	-	-	-	-	-

Soluzione:

Si può disegnare velocemente partendo dalla matrice di adiacenza, partendo dal nodo 1, selezioni rami con il costo più basso e tiro i collegamenti, di volta in volta, la soluzione dovrà essere quella che sommando tutti i rami (senza mai passare dallo stesso nodo 2 volte) abbia il costo minore in assoluto

Sol: $\{(1,2), (1,3), (1,4), (4,5), (4,7), (6,7)\}$

Cost = 18



dal disegno, gli archi rossi son quelli selezionati per il SST, sono stati evidenziati anche i relativi pesi nel caso di stampa non a colori

4.3 Max Flow

Problemi di flusso, dato un grafo diretto e connesso con nodi collegati tra loro da archi, che hanno due valori, a sinistra quanto liquido sta passando (flusso), a destra quanti liquidi massimo l'arco supporta (capacità), dobbiamo trovare la strada ottimale per arrivare a destinazione.

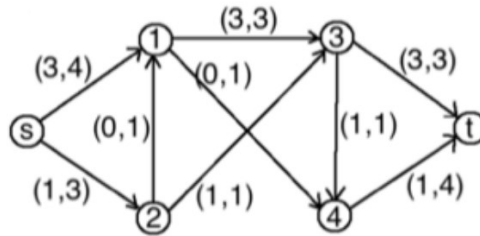
4.3.1 Cenni di Base

- **Capacità**(Capacity): Quanto liquido può passare massimo in quell'arco
- **Flusso**(Flow): quanto liquido sta passando effettivamente in quell'arco
- **Flusso Accettabile**(Feasible Flow): Il flusso che scorre dal nodo $s \Rightarrow t$ rispetta i vincoli di capacità dei suoi archi
- **Arco Saturato** (Saturated Arc): Arco nel quale scorre un flusso pari alla sua capacità
- **Arco vuoto**(Empty Arc): Arco nel quale non scorre un flusso.
- **Flusso di Conservazione** (Residual Flow): Il flusso che scorre tra più nodi deve rispettare i limiti di ogni vincolo di capacità (non sono molto sicuro su questo punto)
- **Taglio** (Cut): selezionando due nodi collegati tra loro, guardo la somma di capacità e/o flusso degli archi uscenti

4.3.2 Flow Network

Esame 2020-07-16 Ex 4

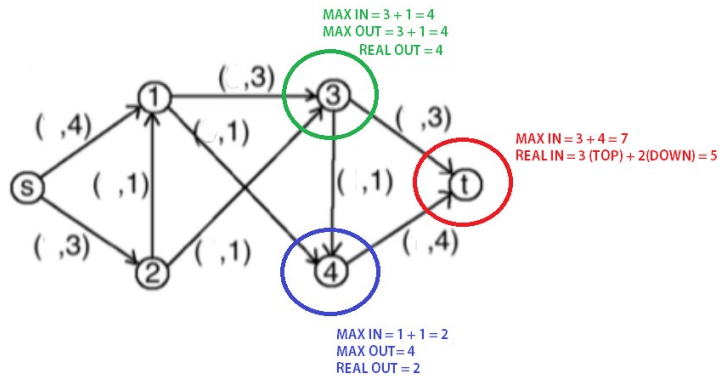
Consider the following network $G = (V, A)$, where for each arc the current flow and the capacity are indicated as (f_{ij}, c_{ij})



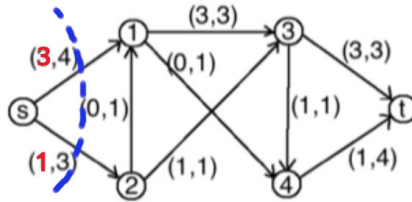
- Is the current flow feasible ? Motivate the answer.
- Provide the value of the optimal max flow that can be carried from s to t on the given network.
- Provide the arcs of the corresponding minimum capacity $s - t$ cut.

Soluzione:

- la soluzione è accettabile, perchè tutti i flussi rispettano le capacità massime degli archi, e il flusso di conservazione, ovvero, nel nodo 4, abbiamo un output di (1,4), se per assurdo lo avessimo avuto di (4,4), per quanto avesse rispettato il vincolo di capacità, avrebbe rotto il vincolo di conservazione, perchè il nodo 4 avrebbe avuto archi entranti per un massimo di flusso dal valore 2, di conseguenza in archi uscenti il nodo 4, può avere massimo 2 di capacità reale, malgrado il suo ramo possa supportare fino a 4.



- il flusso massimo che posso avere in uscita sul nodo t , è 5.
 osservando l'immagine sopra, nel quale ho rimosso i flussi passanti e lasciato solo i flussi di capacità, notiamo che malgrado il nodo t abbia un upperbound massimo di 7, in realtà i suoi archi entranti che arrivano dal nodo 3 e 4, possono sostenere realisticamente e rispettivamente una capacità reale di uscita di 4 per il nodo 3, e di 2 per il nodo 4, dobbiamo anche notare che un nodo uscente di 3 finisce giusto nel nodo 4, quindi abbiamo nel best case scenario che il nodo 3 e 4 possono approvvigionare rispettivamente 3 dall'arco superiore e 2 dall'arco inferiore.
- Ricordando il teorema del Max-flow min-cut: "Il valore massimo di flusso tra s e t equivale sempre al peso del taglio minimo $s - t$ ", ovvero, devo effettuare un taglio che includa il numero minimo di archi, e che separi il nodo s da t , il flusso che scorre in quel taglio, è il flusso massimo che il network può supportare, da immagine qua sotto, vediamo l'esempio separando vicino al nodo s con la linea tratteggiata blu, tale esempio può funzionare anche vicino al nodo t , considerando che in questa immagine anche questo taglio avrebbe due archi, ed effettivamente sommando il flusso del taglio (i numeri evidenziati in rosso) il valore di flusso massimo è 4!:



Questo teorema posso usarlo anche per risolvere il punto 1, ammesso il problema sia ben posto rispettando tutti i vincoli

4.4 DP

Dynamic Programming, usato con Knapsack e Shortest Path (Bellman-Ford)

4.4.1 DP Knapsack 0-1 Dynamic Programming

Esame 2021-02-16 Ex 3

Use Dynamic Programming to solve the following Knapsack Problem: weight $w_j = (4, 2, 3)$, profit $p_j = (2, 1, 2)$, and capacity $C = 5$. Report all the iterations giving the states and their values. Report the optimal solution.

Soluzione:

per il punto 1, dobbiamo costruire 2 matrici, le colonne rappresenteranno il peso dello zaino, mentre le righe rappresentano gli oggetti, di conseguenza avremo matrici 4×6 , conto una riga e colonna extra per il valore 0.

Una matrice rappresenterà la funzione (f) ed una l'insieme di soluzioni (j).

Partendo dalla matrice f, riempiamo colonna riga 0, di 0, lì non possiamo prendere niente.

Successivamente iteriamo per riga, partendo dalla 1, che rappresenta l'oggetto 1, ovvero $w_1 = 4$ e $p_1 = 2$, procedendo per colonne, lo possiamo inserire solo nella 4, ovvero, quando il suo peso può entrare nello zaino,svolgiamo l'algoritmo valutando se $p_j + f^{j-1}(s - w_j) > f^{j-1}(s)$ quindi, vediamo se in colonna 0 (NB. 4-4), sommandoci il profitto attuale di 2, sia un valore maggiore che quello nella riga precedente nella stessa posizione di colonna, quindi 0, se così non è, prendiamo dalla posizione $f^{j-1}(s - w_j)$ e aggiungiamo 2 come nell'immagine .

tengo aggiornata nella seconda matrice J, gli oggetti che sto prendendo, "E" rappresenta insieme vuoto.

	0	1	2	3	4	5		J^0	E	E	E	E	E	E
z^0	0	0	0	0	2	0		J^0	E	E	E	E	E	E
z^1	0	0	0	0	2	2		J^1	E	E	E	E	1	1
z^2	0							J^2	E	E				
z^3	0							J^3	E	E				

Procedendo con la riga 2, e l'oggetto $w_2 = 2$ e $p_2 = 1$, nella colonna 2 posso inserirlo, arrivati alla colonna 4, reiterando l'algoritmo con il procedimento precedente, valuto che nella colonna 2 (NB. 4-2), della riga precedente e sommandoci il profitto di questa riga, ovvero 1, ottengo un valore più basso che prendendolo nella stessa colonna della riga precedente, proseguo quindi ricopiando quest'ultimo valore, ripeto stesso ragionamento per colonna 5.

	0	1	2	3	4	5		J^0	E	E	E	E	E	E
z^0	0	0	0	0	0	0		J^0	E	E	E	E	E	E
z^1	0	0	2	0	2	2		J^1	E	E	E	E	1	1
z^2	0	0	1	1	2	2		J^2	E	E	2	2	1	1
z^3	0	0						J^3	E	E				

Nell'ultima riga seguo i ragionamenti precedenti, l'oggetto $w_3 = 3$ e $p_3 = 2$, può essere inserito solamente a partire dalla 3 colonna, quindi nella colonna 2 ricopio il profitto della riga superiore, arrivati alla colonna 3, confronto che il

profitto della riga superiore è inferiore a quello che avrei prendendo la colonna 0 e sommandoci il profitto di quest'ultimo oggetto, scrivo quindi 2, uguale per la colonna 4, ma posso prendere indistintamente l'oggetto 3 o 1, infine come nell'immagine qua sotto, alla colonna 5, vedo che prendendo la colonna 2 (NB. 5-3), e sommandoci il profitto attuale, ottengo un valore ancora maggiore, scrivo quindi 3.

	0	1	2	3	4	5			0	1	2	3	4	5
z^0	0	0	0	0	0	0	J^0	E	E	E	E	E	E	E
z^1	0	0	0	0	2	2	J^1	E	E	E	E	E	1	1
z^2	0	0	1	1	2	2	J^2	E	E	2	2	E	1	1
z^3	0	0	1	2	2	3	J^3	E	E	2	3	3/1	2,3	

Per la seconda parte dell'esercizio dobbiamo usare l'altro algoritmo di Dynamic Programming dello knapsack.

Calcoliamo il massimo profitto che sarà il nostro upperbound, sommando tutti i profitti: $P = \sum_j p_j = 5$, le colonne qua rappresenteranno i profitti, le righe son le fasi, o gli oggetti come precedentemente.

Imposto nella colonna 0, tutti i valori a 0, e nella riga 0 esclusa la colonna 0, i valori ad ∞ (che nelle tabelle verrà rappresentato come M).

Per la colonna 1 ricopio ∞ dalla riga superiore, arrivati alla colonna 2, posso inserire il profitto del 1 oggetto $w_1 = 4$ e $p_1 = 2$, quindi scrivo 4, e riempio ad infinito il resto, aggiorno di conseguenza la seconda matrice J per indicare cosa sto prendendo come nr di oggetti.

	0	1	2	3	4	5			0	1	2	3	4	5
f^0	0	M	M	M	M	M	J^0	E	E	E	E	E	E	E
f^1	0	M	4	M	M	M	J^1	E	E	1	E	E	E	E
f^2	0						J^2	E						
f^3	0						J^3	E						

Proseguo con la 2 riga e l'oggetto $w_2 = 2$ e $p_2 = 1$, avendo profitto 1, posso inserirlo dalla prima colonna, essendo che nella posizione della riga precedente ho un ∞ , e che l'algoritmo, simile all'es precedente, confrontando $w_j + f^{j-1}(s - p_j) < f^{j-1}(s)$, ho $0+2 = 2$.

Nella colonna 2, mi trovo ad avere nella posizione della riga precedente un valore più basso (4) rispetto ad ∞ nella posizione (1,0) (ricavata dalla riga precedente e dalla colonna 2-1).

proseguo alla colonna 3, ricavo il 6 come valore, dalla colonna 2 (NB. 2-1) e sommandoci il peso attuale (2), scelgo il valore più basso tra le due opzioni: 6 o ∞ e riempio ad ∞ seguendo lo stesso iter il resto delle colonne.

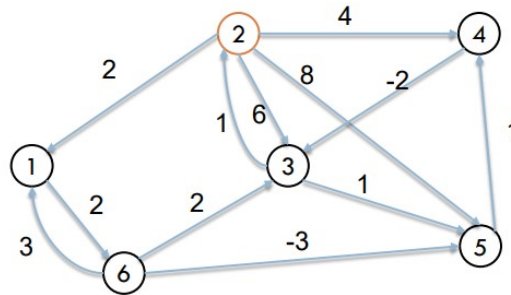
	0	1	2	3	4	5			0	1	2	3	4	5
f^0	0	M	M	M	M	M	J^0	E	E	E	E	E	E	E
f^1	0	M	4	M	M	M	J^1	E	E	1	E	E	E	E
f^2	0	2	4	6	M	M	J^2	E	2	1	1,2	E	E	E
f^3	0						J^3	E						

Per l'ultima riga re-itero l'algoritmo seguendo le stesse procedure, una volta riempita la matrice, seleziono il profitto che raggiunge l'upperbound o ci va più

vicino senza sfolarlo, in questo caso il 5, nella posizione (3,3).

	0	1	2	3	4	5		0	1	2	3	4	5
f^0	0	M	M	M	M	M	J^0	E	E	E	E	E	E
f^1	0	M	4	M	M	M	J^1	E	E	1	E	E	E
f^2	0	2	4	6	M	M	J^2	E	2	1	1,2	E	E
f^3	0	2	3	5	7	9	J^3	E	2	3	2,3	1,3	1,2,3

4.4.2 DP: SSP Bellman's-Ford



Parto dal nodo 2, voglio sapere quale è il percorso meno costoso per arrivare ad ogni nodo!

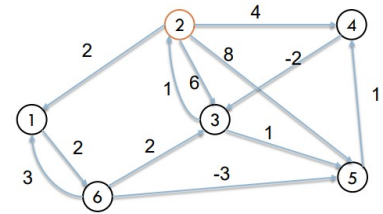
Costruisco due matrici, la matrice f che terrà traccia dei costi, e la matrice $pred$ che terrà traccia dei nodi precedenti in base al costo.

Le colonne rappresentano i nodi di arrivo, le righe sono le iterazioni ne indico $n - 1$ dove n sono i nodi, li posso anche vedere come quanti archi prendo per volta per vedere i costi degli spostamenti, quindi avremo 5 righe + 1 che è quella nr 0, riempiamo nella riga 0, quella dello stato di partenza, tutto ad infinito, tranne il nodo 2, che ha un costo di 0, e la relativa colonna.

Iter.	$f(j)$						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1		0						2				
2		0						2				
3		0						2				
4		0						2				
5		0						2				

Analizziamo la fase 1, o riga 1, stiamo guardando i costi per arrivare al nodo 1, nella prima colonna, scrivo 2 nella matrice di f , ovvero il costo per arrivare dal nodo 2 al nodo 1, tengo aggiornata la matrice delle posizioni segnando 2, che è il nodo precedente da cui sono arrivato, lo stesso approccio si applica anche alle altre colonne, ad eccezione della 6, il nodo 2 non ha collegamenti diretti, quindi il costo è ignoto, mettiamo ∞ .

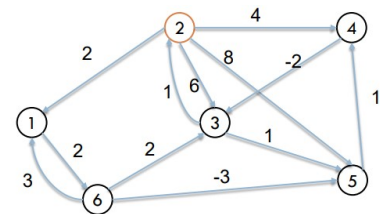
Iter.	$f(j)$						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1	2	0	6	4	8	inf	2	2	2	2	2	-
2		0						2				
3		0						2				
4		0						2				
5		0						2				



Passiamo ora alla fase 2, nel nodo 1 posso arrivarci da 2 o da 6, 6 dalla riga precedente ha costo ∞ , quindi riporto il costo di 2 e il percorso precedente, nella colonna 3, vedo che posso arrivare dal nodo 4, che ha costo -2, sommato al costo precedente che si trova in posizione (1,4) ottengo $4 - 2 = 2$, inserisco quindi 2, e segno il nodo 4 nella matrice prec, 6 non viene analizzato perchè ha ancora costo ∞ .

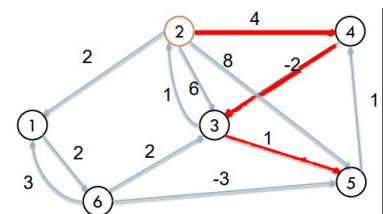
Colonna 5, posso arrivarci dal nodo 6 che ignoriamo, dal 3 e dal 2 che abbiamo già segnato, vedo che dal nodo 3 ho un costo inferiore di quello precedente $6 + 1 = 7$, aggiorno di conseguenza, arrivare a 6 ora è possibile, e posso solo da 1, il costo totale sarà 4, aggiorno le matrici.

Iter.	$f(j)$						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1	2	0	6	4	8	inf	2	2	2	2	2	-
2	2	0	2	4	7	4	2	2	4	2	3	1
3		0						2				
4		0						2				
5		0						2				



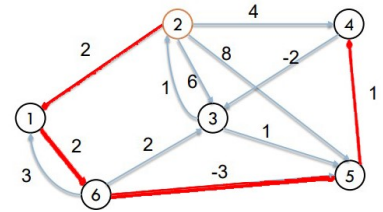
Per la riga 3, vedo che posso aggiornare solo la colonna 5, arrivandoci dal percorso $2 \Rightarrow 4 \Rightarrow 3 \Rightarrow 5$ a costo 1.

Iter.	$f(j)$						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1	2	0	6	4	8	inf	2	2	2	2	2	-
2	2	0	2	4	7	4	2	2	4	2	3	1
3	2	0	2	4	1	4	2	2	4	2	6	1
4		0						2				
5		0						2				



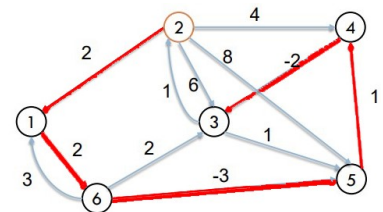
Per la riga 4, guardando i vari costi non riusciamo ad ottenere miglioramenti eccetto per la colonna 4, con un nuovo percorso, usando 4 archi: $2 \Rightarrow 1 \Rightarrow 6 \Rightarrow 5 \Rightarrow 4$ a costo 2!

Iter.	f(j)						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1	2	0	6	4	8	inf	2	2	2	2	2	-
2	2	0	2	4	7	4	2	2	4	2	3	1
3	2	0	2	4	1	4	2	2	4	2	6	1
4	2	0	2	2	1	4	2	2	4	5	6	1
5		0						2				



Infine l'ultima riga, per la colonna 3, usando 5 archi arrivo a costo 0 con questo percorso: $2 \Rightarrow 1 \Rightarrow 6 \Rightarrow 5 \Rightarrow 4 \Rightarrow 3$
il resto non trovo di meglio e rimane invariato

Iter.	f(j)						pred					
	1	2	3	4	5	6	1	2	3	4	5	6
0	inf	0	inf	inf	inf	inf	-	2	-	-	-	-
1	2	0	6	4	8	inf	2	2	2	2	2	-
2	2	0	2	4	7	4	2	2	4	2	3	1
3	2	0	2	4	1	4	2	2	4	2	6	1
4	2	0	2	2	1	4	2	2	4	5	6	1
5	2	0	0	2	1	4	2	2	4	5	6	1



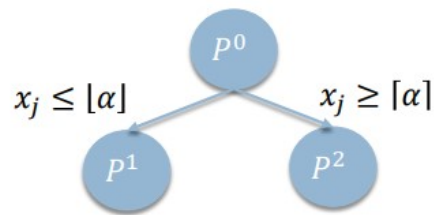
Abbiamo trovato quindi il meno costoso per ogni nodo nell'ultima riga.

4.5 ILP

Integer Linear Programming, Branch & Bound usato con standard e knapsack.

4.5.1 ILP standard B&B

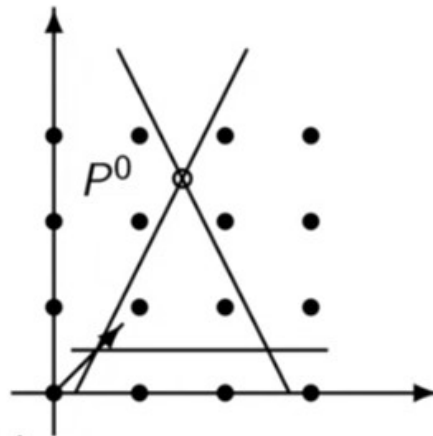
Per questo tipo di problemi usiamo il B& B su sistemi per trovare la soluzione ottimale, è un approccio dicotomico, ogni nodo avrà due sotto problemi, selezionando una variabile frazionaria $x_j = \alpha$, ci sposteremo a sinistra impostando $x_j \leq \lfloor \alpha \rfloor$ e a destra $x_j \geq \lceil \alpha \rceil$



Prendiamo come esempio questo sistema:

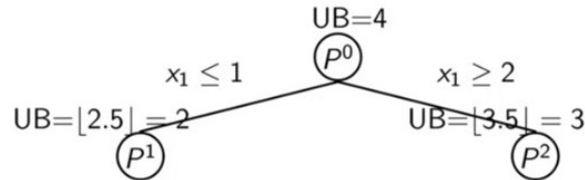
$$\begin{aligned}
& \max : x_1 + x_2 \\
& 4x_1 - 2x_2 \geq 1 \\
& 4x_1 + 2x_2 \leq 11 \\
& 2x_2 \geq 1 \\
& x_1, x_2 \text{ integer}
\end{aligned}$$

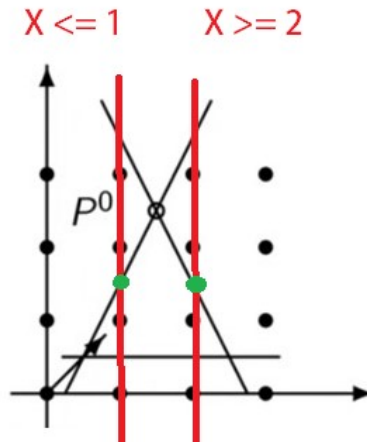
Avendo variabili intere e rappresentando graficamente in problema arriviamo a questa soluzione:



All'interno del triangolo abbiamo la nostra zona ammissibile, i pallini rappresentano gli interi, ovvero dove cercheremo di avvicinarci e per raggiungere la nostra soluzione ottimale intera, facendo il gradiente, abbiamo la nostra freccia con coordinate $(1, 1)$, la nostra soluzione ottimale di conseguenza è la P^0 con coordinata $x_1 = \frac{3}{2}, x_2 = \frac{5}{2}$, soluzione che come abbiamo appena detto non è intera, di conseguenza, dovremmo avvicinarci al pallino nel grafico più vicino alla nostra soluzione ottimale intera.

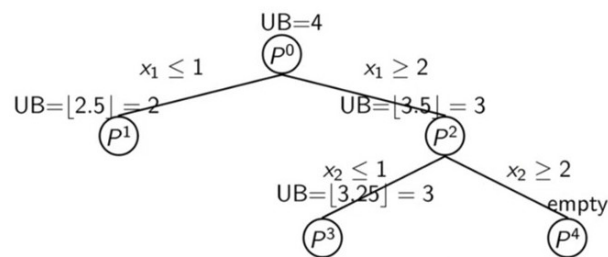
La nostra soluzione con queste coordinate è $z = x_1 + x_2 \Rightarrow \frac{3}{2} + \frac{5}{2} = 4$, sarà il nostro Upper Bound (UB) di partenza del nostro albero (P^0), ora usando gli arrotondamenti di α citati prima, nel ramo sinistro arrotonderemo per difetto, in quello di destra per eccesso x_1 , per cui:





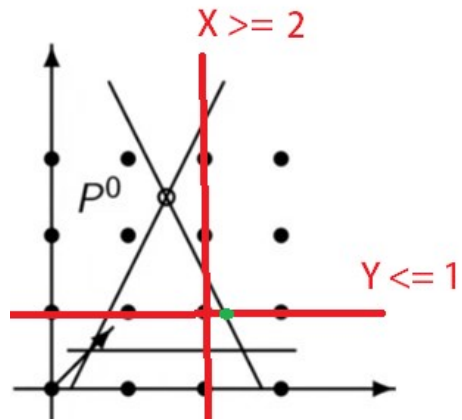
Graficamente risulta così, dividiamo in sezioni per interi, a sinistra abbiamo $x_1 \leq 1$ ed a destra abbiamo $x_1 \geq 2$, considerato il nostro gradiente, le soluzioni ottimali delle due condizioni sono rispettivamente $z = 2,5$ ricavato da $x_1 = 1, x_2 = 1,5$, mentre per la parte di destra abbiamo $z = 3,5$ ricavato da $x_1 = 2, x_2 = 1,5$, ovviamente le z son state ricavate sostituendo gli x_1, x_2 nella funzione obbiettivo di questo problema, che ricordo è: $max : x_1 + x_2$. Vengono arrotondati per difetto le soluzioni, per le relative P^1, P^2 del nostro albero, non avendo ancora soluzioni intere continuiamo.

Non possiamo più spostarci a sx perchè andremmo sullo zero e fuori dalla zona ammissibile del problema, di conseguenza esploriamo il ramo destro:

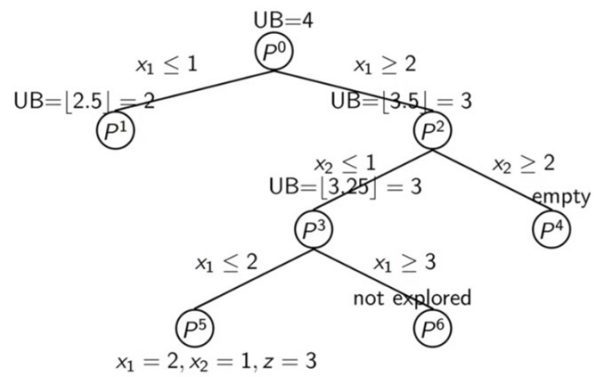


Stiamo ora valutando x_2 che è anch'essa frazionaria.

P^4 esce fuori dalla zona ammissibile del problema e lo vediamo graficamente, quindi scriveremo *empty*, P^3 invece è nella regione ammissibile, graficamente risulta :



Avremo ancora una soluzione frazionaria, corrispondente ad $x_1 = 2,25, x_2 = 1$ per cui $z = 3,25$, arrotondando come upperbound di P^3 avremo 3, possiamo proseguire visto che la soluzione non è ancora intera.



P^6 esce dalla zona accettabile, mentre P^5 questa volta è soluzione intera con coordinate $x_1 = 2, x_2 = 1$ e relativa $z = 3$, possiamo interromperci.

4.5.2 ILP esercizio standard B&B

Esame 2018-07-19 Ex 3

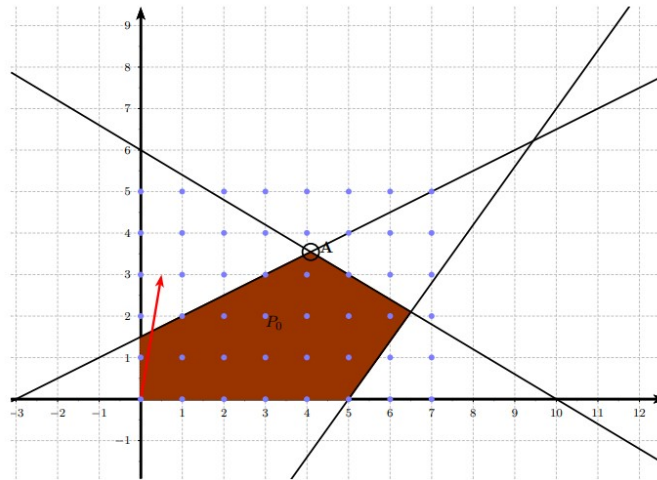
Consider the following PLI model:

$$\begin{aligned} \max & : x_1 + 4x_2 \\ & 7x_1 - x_2 \leq 1 \\ & -2x_1 + x_2 \leq 1 \\ & 2x_2 \geq 1 \\ & x_1, x_2 \text{ free} \end{aligned}$$

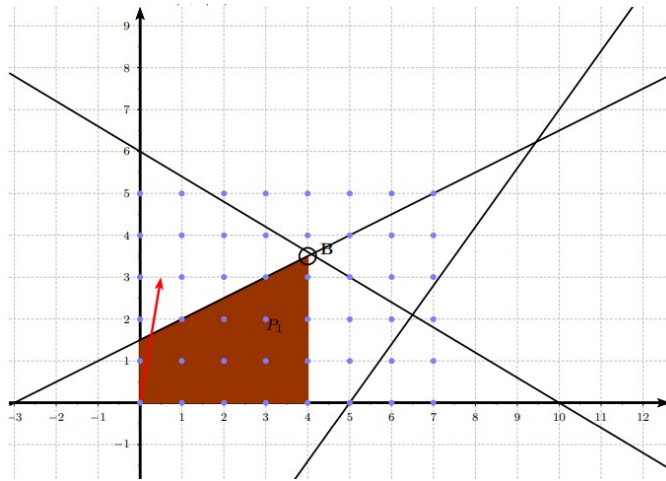
Solve it with the standard Branch-and-Bound algorithm. Solve each relaxed subproblems graphically. Perform the first branching on variable x_1 .

Soluzione:

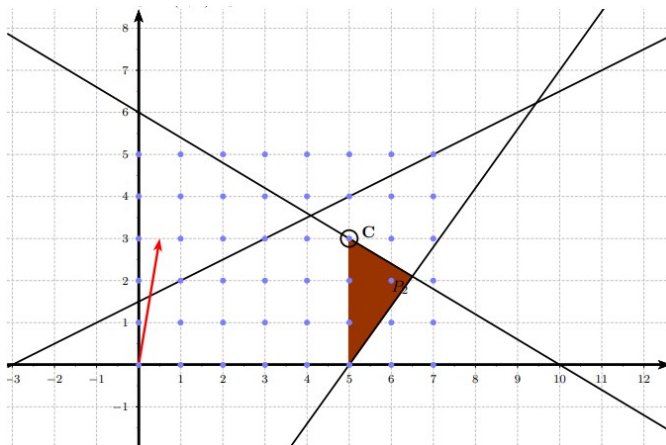
Problem P^0 : $x_A = (\frac{45}{11}, \frac{39}{11})$ $z_A = [\frac{201}{11}] = 18$



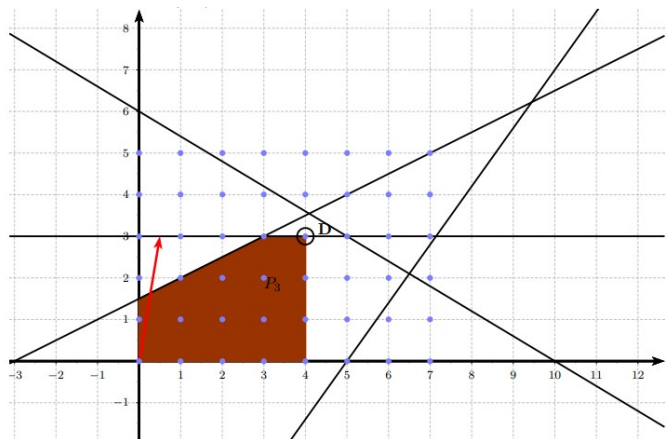
Problem P^1 : $x_B = (4, \frac{7}{2})$ $z_B = 18$

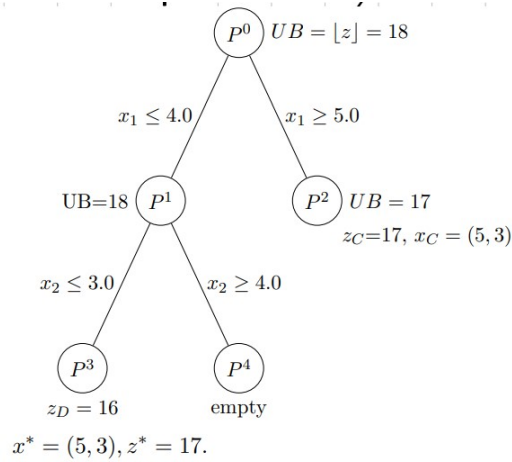


Problem $P^2 : x_C = (4, 3) \quad z_C = 17$



Problem $P^3 : x_D = (4, 3) \quad z_D = 16$





4.5.3 ILP esercizio B&B 0-1

Esame 2020-06-08 Ex 2

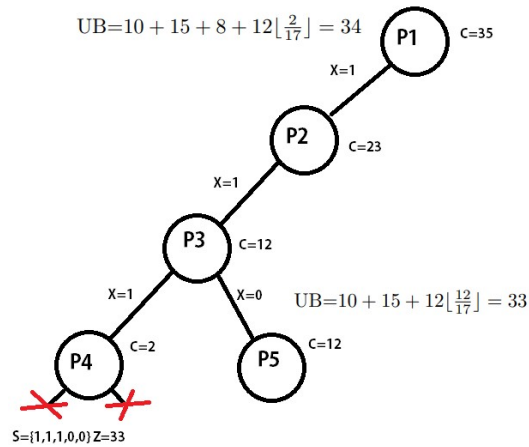
Consider a 0-1 knapsack problem with 5 items with profits $p_j = (10, 15, 8, 12, 11)$, weights $w_j = (8, 15, 10, 17, 20)$ and a knapsack of size 35. Consider the branch-and-bound used to solve the problem and the first subproblem generated by a branching of the type $x_j = 0$. Write the upper bound of this subproblem, and say if the exploration must proceed to lower levels: justify your choice.

Soluzione:

Per prima cosa dobbiamo ordinare gli oggetti per $\frac{\text{valore}}{\text{peso}}$ in ordine decrescente, facendo la frazione $\frac{p_j}{w_j}$, una volta ordinati, in questo esercizio è già stato fatto dal prof, calcoliamo l'Upper bound del nodo p1 con Dantzig's bound

$$UB = \sum_{j=1}^{s-1} p_j + p_j x_s \left(\sum_{j=1}^{s-1} p_j + \lfloor p_j x_s \rfloor \text{ if integer } p_j \right)$$

Dalla figura seguente vediamo che arrivati a P4, ci fermiamo che non possiamo più inserire altri oggetti, e abbiamo la nostra prima soluzione $P4 = \{1,1,1,0,0\}$ con valore 33, risalendo a P3 ci spostiamo a dx non prendendo l'ultimo oggetto (il terzo per la precisione, valore 8 peso 10), come ogni volta che non prendiamo un oggetto, ricalcoliamo UB, in questo caso abbiamo sempre valore 33, e non ha senso esplorare ulteriormente questo ramo.



La sequenza delle operazioni è la seguente:

- Ordino gli oggetti in ordine decrescente di rapporto profitto/peso.
- Scrivo il primo nodo e calcolo U_0 ed ipotizzo C_0 come massimo valore ammissibile, cioè C
- Inizio l'esplorazione dell'albero:
 - se vado a sinistra eredito U_x (Upper Bound) e ricalcolo \bar{c}_x
 - se vado a destra eredito \bar{c}_x (Capacità residua) e ricalcolo U_x
- Per il calcolo di U_x sommo i p_j di tutti gli oggetti per i quali la somma dei w_j risulta minore di C (Capacità del Knapsack). Se ho della capacità residua \bar{c} allora la moltiplico per il valore p_j/w_j e prendo il lower bound $\lfloor \alpha \rfloor$.
- Se arrivo in fondo ad un ramo allora ho una soluzione intera, cioè senza parti di oggetti, e quindi calcolo Z (come valore) e X (come elenco di oggetti).
- Ad ogni nodo, se U_x è minore o uguale alla migliore Z trovata fino a quel momento, allora non proseguo nella esplorazione di quel ramo perchè non troverò un risultato migliore.
- Ad ogni nodo, se andando a sinistra la w_j dell'oggetto è inferiore alla capacità residua \bar{c} allora mi fermo e non inserisco l'oggetto.
- Se arrivo in fondo ad un ramo oppure mi sono fermato per poca capacità residua \bar{c} , allora risalgo l'albero fino a che non ho la possibilità di andare a destra.
- Ho concluso il problema quando tutti i percorsi sono stati esplorati, ed ho trovato la Z^* e la X^* .

5 GPLK

Questa sezione si limita a raccogliere alcuni problemi in GPLK particolari senza spiegazioni.

5.1 Dal modello al codice

5.1.1 Modello Easy

Esame 2016-07-19 Ex 3

Write a GLPK or XPRESS model corresponding to the following mathematical model:

$$\begin{aligned} \max z &= \sum_{i=1}^n \sum_{j=1}^m p_{ij} x_{ij} \\ \sum_{i=1}^n q_i x_{ij} &\leq c & j = 1, \dots, m & \quad (1) \\ \sum_{i \in S} \sum_{j=1}^m x_{ij} &\leq n m y_S & i = 1, \dots, n & \quad j = 1, \dots, m & (2) \\ \sum_{i \notin S} \sum_{j=1}^m x_{ij} &\leq n m y_S & i = 1, \dots, n & \quad j = 1, \dots, m & (3) \\ x_{ij} &\in \{0, 1\} & i = 1, \dots, n & \quad j = 1, \dots, m & (4) \\ y_S &\in \{0, 1\} & & & (5) \end{aligned}$$

Soluzione:

```
1 param n, integer, > 0;
2 param m, integer, > 0;
3 set I := 1..n;
4 set J := 1..m;
5 set S;
6
7 param c, integer, > 0;
8 param p{i in I, j in J}, >= 0;
9 param q{i in I}, >= 0;
10
11 var x{i in I, j in J}, binary;
12 var ys, binary;
13
14 maximize z: sum{i in I, j in J} p[i,j]*x[i,j];
```

```

15
16 s.t. cap{j in J}: sum{i in I} q[i]*x[i,j] <= c;
17 yS1: sum{i in S, j in J} x[i,j] <= n*m*ys;
18 yS2: sum{i in I, j in J: i not in S} x[i,j] <= n*m*
    ys;
19 solve;

```

5.1.2 Caso Particolare 1 Graffa

Esame 2017-02-01 Ex 3

Write a GLPK or XPRESS model corresponding to the following mathematical model:

$$\max : z = \sum_{j \in V} \sum_{i \in U} p_{ij} x_{ij} - \sum_{i \in V \setminus \{0\}} c_i y_i$$

$$\sum_{i \in V \setminus \{n+1\}} x_{0j} \leq 1 \quad (1)$$

$$\sum_{i \in U} x_{ij} \leq 1 \quad j \in V \setminus \{n+1\} \quad (2)$$

$$\sum_{j \in V \setminus \{n+1\}} x_{ij} - \sum_{j \in U \setminus \{0\}} x_{ij} = \begin{cases} 0 & i \in V \setminus \{0, n+1\} \\ M & i = 0 \\ -M & i = n+1 \end{cases} \quad (3)$$

$$\sum_{j \in V \setminus \{n+1\}} f_{ij} - \sum_{j \in U \setminus \{0\}} f_{ij} = 0 \quad i \in V \setminus \{0, n+1\} \quad (4)$$

$$f_{ij} \leq q_{ij} x_{ij} \quad i, j \in V \quad (5)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in U, j \in V \quad (6)$$

$$y_j \in \{0, 1\} \quad \forall j \in U \quad (7)$$

$$f_{ij} \geq 0 \quad \forall i \in U, j \in V \quad (8)$$

Soluzione:

```

1 param n, integer, > 0;
2 param m, integer, > 0;
3 set V := 0..n+1;
4 set U := 0..n+1;
5 set V0 := 1..n-1;
6 set Vn1 := 0..n;
7 set V0n1 := 1..n;
8 set U0 := 1..n+1;
9 param M;

```

```

10
11 param p{i in U, j in V}, >= 0;
12 param q{i in U, j in V}, >= 0;
13 param c{i in Vn1}, >= 0;
14
15 var x{i in U, j in V}, binary;
16 var y{ j in V0}, binary;
17 var f{i in U, j in V}, >= 0;
18
19
20 maximize z: sum{i in U, j in V } p[i,j]*x[i,j] -
    sum{i in V0} c[i]*y[i];
21
22 s.t. from0: sum{j in Vn1} x[0,j] <= 1;
23 inj{j in Vn1}: sum{i in U} x[i,j] <= 1;
24 balance1x{i in V0n1}: sum{j in Vn1} x[i,j] - sum{j in
    U0} x[j,i] = 0;
25 balance2x: sum{j in Vn1} x[0,j] - sum{j in U0} x[j,0]
    = M;
26 balance3x: sum{j in Vn1} x[n+1,j] - sum{j in U0}
    x[j,n+1] = -M;
27 balance1f{i in V0n1}: sum{j in Vn1} f[i,j] - sum{j in
    U0} f[j,i] = 0;
28 fx{i in V, j in V}: f[i,j] - q[i,j]*x[i,j] <= 0;
29 solve;

```


5.1.3 Caso Particolare 2 Sommatoria doppio insieme

Esame 2017-06-29 Ex 3

Write a GLPK or XPRESS model corresponding to the following mathematical model:

$$\min : z = \sum_{j=1}^n c_{ij} x_{0j}$$

$$\sum_{j=1}^{n+1} x_{0j} - \sum_{j=0}^n x_{j,n+1} = 0 \quad (1)$$

$$\sum_{j=0}^{n+1} x_{ij} - \sum_{j=0}^{n+1} x_{ji} = 0 \quad i = 1, \dots, n \quad (2)$$

$$\sum_{i=0, i \notin S_k}^{n+1} x_{ij} \sum_{j \in S_k} x_{ij} \leq 1 \quad k = 1, \dots, p \quad (3)$$

$$\sum_{i \in S_k} \sum_{j \in S_k} x_{ij} \geq 1 \quad k = 1, \dots, p \quad (4)$$

$$x_{ij} \in \{0, 1\} \quad i, j = 0, \dots, n+1, i \neq j \quad (5)$$

Soluzione:

```

1 param n, integer, > 0;
2 param p, integer, > 0;
3
4 set I := 0..n+1;
5 set Ip := 1..n;
6 set P := 1..p;
7 set S{k in P};
8
9 param q{i in I}, >= 0;
10 param c{i in I, j in I}, >= 0;
11
12 var x{i in I, j in I}, >= 0, binary;
13
14
15 minimize z: sum{j in I} c[0,j]* x[0,j];
16
17 s.t.
18 V1: sum{j in I: j != 0} x[0,j] - sum{j in I: j !=
      n+1} x[j,n+1] = 0;

```

```

19 V2{i in Ip}: sum{j in I} x[i,j] - sum{j in I} x[j,i]
    = q[i];
20 V3{k in P}: sum{i in I diff S[k], j in S[k]} x[i,j]
    <= 1;
21 V4{k in P}: sum{i in S[k], j in S[k]} x[i,j] >= 1;
22 solve;

```

5.2 Dal codice al modello

5.2.1 Normale

Esame 2017-02-17 Ex 3

Consider the following GLPK program and write the corresponding mathematical model.

```

1 param n, integer, > 0;
2 param m, integer, > 0;
3 set V := 0..n;
4 set U := 0..n;
5 set V1 := 1..n;
6 set U1 := 1..n;
7 set UU{i in U};
8
9 param c{i in U, j in V}, >= 0;
10 param s{i in U1}, >= 0;
11 param q{i in U, j in V}, >= 0;
12
13 var x{i in U, j in V}, binary;
14 var y{ j in U1}, binary;
15 var f{i in U, j in V}, >= 0;
16
17 maximize z: sum{i in U, j in V } c[i,j]*x[i,j] -
    sum{i in U1} s[i]*y[i];
18 s.t. T00: sum{j in V} x[j,0] <= 1;
19 balance{i in U1}: sum{j in V1} x[i,j] - sum{j in U1}
    x[j,i] = 0;
20 FandX{i in V, j in V}: f[i,j] - q[i,j]*x[i,j] <= 0;
21 XandY{i in U1}: sum{j in V} x[i,j] <= sum{j in V}
    q[i,j]*y[i];
22 ContrUsubset{i in U}: sum{j in UU[i]} x[i,j] <= 1;
23 solve;
    Soluzione:

```

$$\max : z = \sum_{i=0}^n \sum_{j=0}^n c_{ij} x_{ij} + \sum_{i=0}^n s_i y_i$$

$$\sum_{j=0}^n x_{j0} \leq 1 \quad (1)$$

$$\sum_{j=0}^n x_{ij} - \sum_{j=1}^n x_{ji} = 0 \quad i = 1, \dots, n \quad (2)$$

$$f_{ij} - q_{ij} x_{ij} \leq 0 \quad i, j = 0, \dots, n \quad (3)$$

$$\sum_{j=0}^n x_{ij} \leq \sum_{j=0}^n q_{ij} y_i \quad i = 1, \dots, n \quad (4)$$

$$\sum_{j \in UU_i} x_{ij} \leq 1 \quad i = 0, \dots, n \quad (5)$$

$$x_{ij} \in \{0, 1\} \quad i, j = 0, \dots, n \quad (6)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, n \quad (7)$$

$$f_{ijt} \geq 0 \quad i, j = 0, \dots, n \quad (8)$$

6 Domande varie di teoria

Il conteggio si svolge così all'esame, avrete 5 domande, ogni domanda pu darvi massimo 1 punto a giusta o togliervi massimo 1 punto se sbagliate tutto, ogni domanda pu avere pi di una scelta giusta, se sbagliate una scelta all'interno di quella domanda, l'errore rimane confinato lì ed andr a sottrarsi al monte punti, ad esempio: se la domanda X, ha 3 scelte giuste, quindi un totale di 3 punti interni a quella domanda, io ne segno 2 giuste ed una sbagliata, ottengo $2 - 1 = 1$ punto, che sui 3 giusti vale 0.33 punti reali per l'esame!

nel caso avessi preso tutte e 3 le scelte giuste della domanda X quindi facendo internamente a quella domanda 3 punti, avrei ottenuto 1 punto pieno per il mio esame.

Il prof arrotonda per eccesso, ovviamente se in una domanda sbaglio tutto posso andare in negativo fino ad un massimo di -1 punto per domanda nel conto del voto finale dell'esame, rispondete se siete sicuri.

6.1 PLC degenerate

A basic solution of a PLC problem is called degenerate when:

- A — there are more variables than constraints;
- B — the number of basic variables with zero value is equal to the number of constraints;
- C — the number of basic variables with zero value is equal to the difference of the number of columns and rows ($n - m$);
- D — the number of basic variables with non-zero value is equal to the number of rows in the constrains matrix;
- E — the basic matrix is invertible.

Soluzione:

Nessuna di queste giusta.

6.2 PLC sensitivity

Given a PLC problem the sensitivity analysis of a right hand side value b :

- A — defines an upper bound on the optimal solution value;
- B — defines the minimum and maximum value of b which does not change the optimal basis;
- C — defines the minimum and maximum value of b which does not change the value of the optimal solution;
- D — defines the range of values of b which does not change the reduced costs;
- E — defines the range of values of b which does not change the dual variables;

Soluzione:

(B)

6.3 B&B

The branch-and-bound method:

- A — is used to find the computational complexity of the problem;
- B — can be applied to minimization problems in standard form;
- C — finds the optimal solution of an NP-complete problem;
- D — always use as bounding procedure the solution of a PLC problem;
- E — none of the above answers.

Soluzione:

(C)

6.4 PLC minimization

Given a PLC problem P in minimization form, with 2 constraints with sign ' \geq ' and 1 constraint with sign '=', and 5 variables, consider the dual problem D:

- A — D has 3 variables and 5 constraints;
- B — D has 5 variables and 3 constraints;
- C — the first variable of D must be non-negative;
- D — the third variable of D is unrestricted in sign;
- E — the second variable of D must be non-positive.

Soluzione:

(A,C,D)

6.5 cutting Plane

The cutting plane method:

- A — is used to solve linear programming problems with continuous variables;
- B — terminates in a polynomial number of iterations;
- C — is used to solve NP-complete (difficult) problems;
- D — at each iteration uses the simplex algorithm to solve the current sub-problem;
- E — uses the lowest-first strategy to explore the subproblems.

Soluzione:

(C,D)

6.6 PLC dual

Given a primal PLC problem in minimization form and its dual:

- A — if the primal has finite solution, then the dual has finite solution;
- B — a feasible dual solution may have a value smaller than that of the optimal primal solution;
- C — if the primal is unbounded, then the dual is unbounded;
- D — any feasible solution of the primal has a value greater or equal to any feasible solution of the dual;
- E — if the primal is empty, then the dual is empty.

Soluzione:

(A,B,D)

6.7 Dijkstra

The Dijkstra algorithm :

- A — cannot be used in a graph with negative cost arcs;
- B — cannot be used in a graph with negative circuits;
- C — can be used to find the shortest path in a directed graph, from a source node to a sink node;
- D — can be used to find the minimum cost spanning tree;
- E — cannot be applied to acyclic graphs.

Soluzione:

(A,B,C)

6.8 B&B knapsack

The branch-and-bound method for 0-1 knapsack problems:

- A — uses a decision tree with an exponential number of levels;
- B — uses a decision tree with a polynomial number of levels;
- C — uses the lowest-first strategy to explore the decision tree;
- D — computes an upper bound at each node of the decision tree;
- E — uses a decision tree with an exponential number of nodes.

Soluzione:

(B,E)

6.9 Branch & cut

The branch-and-cut method:

- A — is used to solve linear programming problems with continuous variables;
- B — always uses the lowest-first strategy to explore the subproblems.;
- C — is used to solve NP-complete (difficult) problems;
- D — terminates in a polynomial number of iterations;
- E — at each iteration uses the simplex algorithm to solve the current sub-problem.

Soluzione:

(C,E)

6.10 PLC

Given a PLC problem with n variables and m constraints, a basic matrix is:

- A — a collection of m constraints;
- B — none of the above answers.;
- C — a square $m \times m$ matrix with value 1 on the main diagonal;
- D — a collection of $n - m$ columns of the constraint matrix;
- E — a square submatrix of the constraint matrix, that can be inverted.

Soluzione:

(B)

6.11 Soluzione Base

Consider the two problems, $P : \min\{c^T x : Ax = b, x \geq 0\}$, $D : \max\{u^T : u^T A \leq c^T\}$, then:

- A — a pair of solution x and u with x feasible for P and u feasible for D are optimal if $\{Ax - b\}u^T = 0$;
- B — a pair of solution x and u with x feasible for P and u feasible for D are optimal if $c^T x \geq u^T b$;
- C — a feasible solution of P gives a lower bound on the optimal value of D ;
- D — a solution u is optimal if $u^T A \leq c^T$ and $u^T \geq 0$;
- E — a feasible solution of D gives a lower bound on the optimal value of P .

Soluzione:

(E)

6.12 Ford-Fulkerson

The Ford-Fulkerson algorithm for the max-flow problem:

- A — at each iteration computes the shortest path from the source to the sink;
- B — terminate when an $s - t$ cut of the graph is saturated;
- C — uses saturated arcs in their opposite direction;
- D — minimizes the total cost of the flow;
- E — uses non-empty, non-saturated arcs in both directions.

Soluzione:

(B,C)

6.13 PLI minimization & relaxation

Given a PLI problem IP in minimization form, consider its continuous relaxation and let z^* be the optimal value of the relaxation:

- A — the least integer greater than or equal to z^* is a lower bound on the optimal IP value if the coefficients of the objective function are integer;
- B — z^* is a lower bound on the optimal IP value;
- C — the least integer greater than or equal to z^* is an upper bound on the optimal IP value;
- D — z^* is an upper bound on the optimal IP value;
- E — the least integer greater than or equal to z^* is a lower bound on the optimal IP value if the coefficients of the objective function are rational numbers;

Soluzione:

(A,B)

6.14 B&B knapsack 0-1

The branch-and-bound method for 0-1 knapsack problem:

- A — stop the search when level n is reached;
- B — computes the upper bound only in the son nodes generated by a constraint of the kind $x = 1$;
- C — computes the upper bound only in the son nodes generated by a constraint of the kind $x = 0$;
- D — use the highest first strategy to explore the tree;
- E — has $O(2^n)$ tree levels;

Soluzione:

(C,E)

6.15 NP problem

a generic NP-complete problem:

- A — can be solved exactly using the simplex algorithm with the addition of Gomory cuts;
- B — can be solved exactly using the simplex algorithm with a non-polynomial number of iteration;
- C — cannot be solved exactly in polynomial time;
- D — can be solved exactly using a binary decision-tree with n levels, where n is the number of variables;
- E — can be solved exactly using Dijkstra algorithm;

Soluzione:

(C,D)