gd

Gabriele Dragotto
gabriele.dragotto@studenti.polito.it

Ricerca operativa

PL - Programmazione lineare

RICERCA OPERATIVA

*La ricerca operativa è la branca della **matematica applicata** in cui problemi decisionali complessi vengono analizzati e risolti mediante **modelli matematici e metodi quantitativi avanzati.***
*L'obiettivo è quello di fornire un **supporto alla presa di decisioni.***

DEFINIZIONE

*La programmazione lineare è quella branca della ricerca operativa che si occupa di studiar**e algoritmi di risoluzione per problemi di ottimizzazione lineari.***

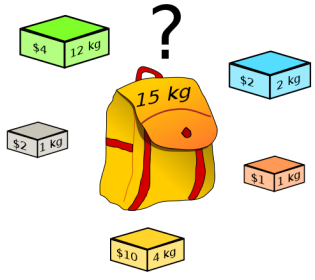
$$\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$

Un problema generico di programmazione lineare consta delle seguenti componenti:

1. **NV VARIABILI DECISIONALI/CONTROLLO**
   $X_1, X_2, \ldots, X_{NV}$
   Contenute in un **vettore colonna X** delle variabili.
   Possono essere **reali non negative, intere non negative o binarie (0,1)**
2. **UN SISTEMA DI VINCOLI**
3. **FUNZIONE OBIETTIVO**
   $$F.O. = \sum_{i=1}^{NV} c_i \cdot x_i = \underline{c}^T \underline{x}$$
   Dove le variabili decisionali vengono moltiplicate per **un rispettivo moltiplicatore c**

KNAPSACK PROBLEM

*Given a **set of items, each with a weight and a value**, determine the **number of each** item to include in a collection so that the total weight is less than or equal to a **given limit** and the total value is as large as possible*



$$\max \sum_{i \in N} u_i x_i$$

$$\sum_{i \in N} w_i x_i \leqslant W$$

$$x_i \in \{0,1\} \quad \forall i \in N$$

POLITOPO CONVESSO

In uno spazio euclideo un insieme convesso è un insieme nel quale, per ogni coppia di punti, il ***segmento che li congiunge è interamente contenuto nell'insieme.***

TEOREMA DI ESISTENZA DI UN ESTREMO

Si dimostra che per ogni politopo convesso, esiste un elemento tale che:

$$\|x_0\| < \|x\| \quad \forall x \in A \setminus \{x_0\}$$

TEOREMI UTILI

Teorema 10.1.1 (Vertici di un poliedro) *Sia dato un poliedro $S = \{x \in R^n : \ Ax \geq b\}$. Un punto $\bar{x} \in S$ è un vertice se e solo se esistono n righe a_i^T della matrice A corrispondenti ai vincoli attivi in $\bar{x}$ linearmente indipendenti, cioè se e solo se risulta*

$$rango\{A_{I(\bar{x})}\} = n$$

dove $A_{I(\bar{x})}$ la matrice $|I(\bar{x})| \times n$ costituita dalle righe di A con indice in $I(\bar{x})$

Corollario 10.1.2 *Sia dato un poliedro $S = \{x \in \mathbb{R}^n : \ Ax \geq b\}$. Se la matrice A ha un numero di righe linearmente indipendenti minore di n, allora S non ha vertici.*
In particolare se $m < n$ allora S non ha vertici.

2.1.2 Sistemi di equazioni lineari

- *Sistemi di equazioni in forma matriciale*: un sistema di m equazioni in n incognite può essere messo in forma matriciale:
  $Ax = b$, con $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$.
- *Teorema di Rouché-Capelli*:
  $Ax = b$ ammette soluzioni $\iff \rho(A) = \rho(A|b) = r$ (∞^{n-r} soluzioni).
- *Operazioni elementari su matrici*:
  - scambiare la riga i con la riga j;
  - moltiplicare la riga i per uno scalare non nullo;
  - sostituire alla riga i, la riga i più α volte la riga j ($\alpha \in \mathbb{R}$).

  Le operazioni elementari sulla matrice aumentata $[A|b]$ non alterano l'insieme delle soluzioni ammissibili del sistema $Ax = b$.
- *Metodo di Gauss-Jordan* per la soluzione di sistemi $Ax = b$: eseguire delle operazioni elementari sulla matrice aumentata in modo da ottenere in A una sottomatrice identità di dimensioni pari a $\rho(A) = \rho(A|b)$.

INESISTENZA SOLUZIONE

1. **PROBLEMA IMPOSSIBILE**
   Avviene quando **i vincoli** non permettono la realizzazione della soluzione.

2. **PROBLEMA ILLIMITATO**
   Il problema risulta illimitato nella **direzione del gradiente della funzione obiettivo.**

MODELLAZIONE LOGICA

CONDIZIONE ONE-TO-MANY 1-N

Si verifica nel caso una data variabile x_i possa assumere un set limitato di k valori quali $p_1, p_2, \ldots, p_k$.
Introduco **K VARIABILI** y_i

$$\sum_{i=1}^{k} y_i = 1$$

$$x_j = \sum_{i=1}^{k} p_i y_i$$

BIG-M PROBLEM

Si verifica nel caso una data variabile x_i essere limitata **superiormente o** inferiormente da dei valori K, M
Introduco **una variabile fittizia binaria y_i (0,1) e la vincolo**

$$x_j \geqslant K y_j.$$

$$x_j \leqslant M y_j$$

IF-THEN-ELSE

Si verifica nel caso una data variabile x_i essere incompatibile con un'altra variabilex_j
Introduco **due variabili fittizie binarie (0,1) e le vincolo**

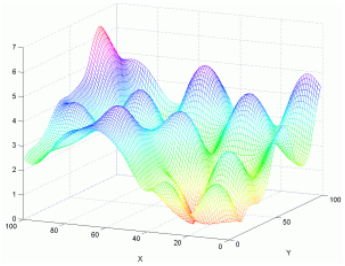
$$x_i \leqslant M y_i$$

$$x_j \leqslant M y_j$$

$$y_i + y_j \leqslant 1.$$

P. NON LINEARE

La programmazione nonlineare implica che i vincoli del problema non siano rappresentati da semplici disequazioni di primo grado, ma da vincoli matematicamente di ordine superiore.



Metodo del simplesso

SIMPLESSO

In matematica, il simplesso n-dimensionale è il ***politopo n-dimensionale col minor numero di vertici.***

Il simplesso di dimensione zero è un **singolo punto**, il simplesso bidimensionale un **triangolo** e quello tridimensionale un **tetraedro**.

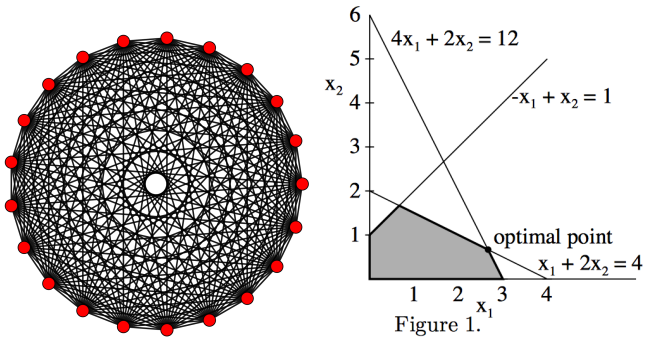


Figure 1.

FORMA STANDARD

La forma standard prevede un modello di ***minimo*** *a* ***variabili positive*** *con* ***vincoli di uguaglianza***

$$\min \bar{c}^T \bar{x}$$
$$A\bar{x} = \bar{b}$$
$$\bar{x} \geq \bar{0}$$

CONVERSONE PROBLEMI IN FORMA STANDARD

1. **FUNZIONE OBIETTIVO**
   Nel caso questa sia di massimo, viene **cambiato il segno.**
   $\max \bar{c}^T \bar{x} = \min(-\bar{c}^T \bar{x})$

2. **VINCOLI**
   In base al segno della disuguaglianza **aggiungo o sottraggo uno slack**.
   Nel caso la variabile non sia vincolata viene rappresentata come **differenza di variabili non negative**

$\bar{a^i}\,\bar{x} \leq b_i$	$\bar{a^i}\,\bar{x} + y_i = b_i$
$\bar{a^i}\,\bar{x} \geq b_i$	$\bar{a^i}\,\bar{x} - y_i = b_i$
$LIBERE$	$x_i = u_i - v_i$

EX

$$\begin{array}{rrrrrrr} \max & 130x_1 & + & 100x_2 & & & \\ & 1,5x_1 & + & x_2 & \leq & 27 \\ & x_1 & + & x_2 & \leq & 21 \\ & 0,3x_1 & + & 0,5x_2 & \leq & 9 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$

$$\begin{array}{rrrrrrrrrrrrr} \min z = & -130x_1 & - & 100x_2 & & & & & & & & = & 27 \\ & 1,5x_1 & + & x_2 & + & s_1 & & & & & & = & 21 \\ & x_1 & + & x_2 & & & + & s_2 & & & & = & 9 \\ & 0,3x_1 & + & 0,5x_2 & & & & & + & s_3 & & = & 0 \\ & x_1 & , & x_2 & , & s_1 & , & s_2 & , & s_3 & \geq & & \end{array}$$

SOLUZIONI DI BASE

Sia data una generica matrice $A \in \Re^{m \times n}$ di ***rango massimo.***
Una base B di A è una sottomatrice quadrata di A di rango massimo *o, in altri termimi, una matrice $B \in \Re^{m \times m}$ ottenuta scegliendo* ***m colonne linearmente indipendenti*** *della matrice A*

$$A = [B|F] \quad B \in \mathbb{R}^{m \times m}, det(B) \neq 0 \qquad x = \begin{bmatrix} x_B \\ x_F \end{bmatrix}, x_B \in \mathbb{R}^m, x_F \in \mathbb{R}^{n-m}$$

$$\begin{aligned} &\min c_B^T x_B + c_N^T x_N \\ &Bx_B + Nx_N = b \\ &x_B \geq 0, \\ &x_N \geq 0 \end{aligned}$$

SOLUZIONI AMMISSIBILI
Sono punti contenuti nel **politopo convesso n-dimensionale**

SOLUZIONI DI BASE DEGENERI
Una delle variabili in base **risulta nulla.**
Se più soluzioni ottime sono degeneri, il simplesso looppa.

SOLUZIONI AMMISSIBILI OTTIME
Sono i **vertici del politopo convesso n-dimensionale**, ottenute annullando le variabili fuori base tale che:

$$x = \begin{bmatrix} x_B \\ x_F \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$$

TEOREMA FONDAMENTALE PL

Dato un problema di programmazione lineare, ***se esiste una soluzione ottima****, allora* ***esiste una soluzione ammissibile di base ottima.***

1. **LIMITE DELLE SOLUZIONI**

   Mentre il numero di **soluzioni ammissibili è illimitato** $\infty^{(n-r)}$, il numero di soluzioni **ammissibili di base è limitato superiormente dal numero delle possibili combinazioni** di **m colonne** scelte tra le **n colonne di A**.

METODO DEL SIMPLESSO

*Il metodo del simplesso è un metodo iterativo che permette di **esplorare in modo efficiente l'insieme delle soluzioni ammissibili** di base, a partire da una soluzione ammissibile di base data.*

L'algoritmo garantisce ad ogni passaggio:

1. **SOLUZIONI AMMISSIBILI**
2. **UNA SOLUZIONE MIGLIORE**

COSTI RIDOTTI

Si definiscono costi ridotti le quantità

$$r_j = c_j - z_j \forall j = 1...n$$

c_j VARIAZIONE DELLA FUNZIONE OBIETTIVO

Se le altre variabili fuori base fossero **nulle** e la variabile x_j **aumentasse di un unità.**

TABLEAU IN FORMA MATRICIALE

Sfruttando la sottomatrice delle colonne in base e quella fuori base posso scrivere

$$\begin{aligned} &min \quad \bar{c}^T \bar{x} \\ &A\bar{x} = \bar{b} \\ &\bar{x} \geq 0 \end{aligned} \qquad A=[B,D] \qquad \bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{x}_D \end{pmatrix}$$

DA CUI DISCENDE

$$B\bar{x}_B + D\bar{x}_D = \bar{b}$$
$$z = \bar{c}_B^T \bar{x}_B + \bar{c}_D^T \bar{x}_D$$

$$\bar{x}_B = B^{-1}\bar{b} - B^{-1}D\bar{x}_D$$
$$\bar{r}_D^T = \bar{c}_D^T - \bar{\lambda}^T D$$
$$\bar{\lambda}^T = \bar{c}_B^T B^{-1}$$

PROBLEMA ILLIMITATO

Se tutti i **coefficienti della colonna corrispondente alla variabile entrante sono negativi** o nulli, allora il problema è illimitato.

REGOLA DI BLAND

*Tra le variabili candidate ad entrare/uscire dalla base scegliere sempre quella con **indice minore***

ALGORITMO DEL SIMPLESSO REVISIONATO

1. CONSIDERO I COSTI RIDOTTI

$$\overline{r_D}^T = \overline{c_D}^T - \overline{\lambda}^T D$$

Se $\overline{r_D} \geq 0$ **STOP: OTTIMO RAGGIUNTO**

2. DETERMINO LA VARIABILE DA FAR ENTRARE IN BASE

$$r_q: \ r_q < r_i \ \ \forall i = 1 \ldots n$$

Scelgo la variabile con **costo ridotto minore** di tutte le altre.
Calcolo la rappresentazione del vettore della q-esima colonna in termini della base corrente

$$\overline{y_q} = B^{-1} \overline{a_q}$$

3. DETERMINO LA VARIABILE DA FAR USCIRE IN BASE

$$\theta = min\left\{ \frac{y_{i0}}{y_{iq}} \right\} \ \ con \ \ y_{iq} > 0$$

Scelgo la variabile con **rapporto non negativo minore**
(Divido il termine noto della riga per il valore nella colonna della variabile che entra in base)

4. AGGIORNAMENTO E RICALCOLO COSTI

Aggiungo alla base il vettore che entra e faccio uscire l'altro, tenendo presente che essi si devono **unicamente scambiare in posizione sia in B che D.**
Ricalcolo **i costi ridotti** e ritorno allo step 1.
Calcolo la matrice invertita B^{-1} e la soluzione di base corrispondente $\overline{x_B} = B^{-1}\overline{b}$

DUALITA'

*The duality principle is the principle that **optimization problems may be viewed from either of two perspectives**, the primal problem or the dual problem.*

$$\begin{array}{ccc} n_{vincoli} & = & n_{variabili} \\ min\ \bar{c}^T\bar{x} & & max\ \bar{\lambda}^T\bar{b} \\ A\bar{x} \geq \bar{b} & & \bar{\lambda}^T A \leq \bar{c}^T \\ \bar{x} \geq 0 & & \bar{\lambda} \geq 0 \end{array}$$

A *Matrice* $m\,x\,n$
$\bar{c}$ *Vettore colona* **dim=n**
$\bar{b}$ *Vettore colonna* **dim=m**
$\bar{x}$ *Vettore colonna* **dim=n**
$\bar{\lambda}$ *Vettore colonna* **dim=m**

VARIABILI LIBERI NEL DUALE

Unicamente se vi è un vincolo di uguaglianza

	PRIMALE	**DUALE**	
	$\min c^T x$	$\max b^T u$	
VINCOLI	$= b_i, \quad i \in I$ $\geq b_i, \quad i \in J$	$u_i, \quad i \in I$, libere $u_i, \quad i \in J, u_i \geq 0$	VARIABILI
VARIABILI	$x_j \geq 0, \quad j \in M$ $x_j, \quad j \in N$ libere	$\leq c_j, \quad j \in M$ $= c_j, \quad j \in N$	VINCOLI

Esempio 5.1.1 *Si consideri il seguente problema di Programmazione Lineare*

$$\begin{cases} \min 2x_1 + 3x_2 + 4x_3 + x_4 \\ x_1 - 5x_3 + 2x_4 \geq 7 \\ 2x_1 + 4x_2 - 6x_3 \geq 9. \end{cases}$$

Il problema duale associato è

$$\begin{cases} \max 7u_1 + 9u_2 \\ u_1 + 2u_2 = 2 \\ 4u_2 = 3 \\ -5u_1 - 6u_2 = 4 \\ 2u_1 = 1 \\ u_1 \geq 0, u_2 \geq 0. \end{cases}$$

TEOREMA DUALITA' DEBOLE

Per ogni soluzione ammissibile $\bar{x}$ del problema primale ed ogni soluzione ammissibile $\bar{\lambda}$ del problema duale si ha che

$$c^T\bar{x} \geq b^T\bar{\lambda}$$

AUTOINDUZIONE DELLA LIMITAZIONE DELLA SOLUZIONE

DUALITA' DEBOLE COROLL1

Per ogni soluzione ammissibile $\bar{x}$ del problema primale ed ogni soluzione ammissibile $\bar{\lambda}$ del problema duale tali che

$$c^T \bar{x} = b^T \bar{\lambda}$$

Allora le ***soluzioni sono corrispettivamente ottime per i due problemi***

DUALITA' DEBOLE COROLL2

Se un problema è illimitato nella direzione del gradiente della funzione obiettivo, allora anche ***l'altro problema è illimitato.***

TEOREMA DELLA DUALITA' FORTE

Sia dato un problema primale avente soluzione $\bar{x}$ ottima. Allora vale la seguente proposizione che lega alla soluzione del problema duale $\bar{\lambda}$

$$\bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{0} \end{pmatrix} \quad A=[B,D] \quad \bar{c} = \begin{pmatrix} \bar{c}_B \\ \bar{c}_D \end{pmatrix}$$

$$\bar{\lambda}^T = \bar{c}_B^T B^{-1}$$

CONDIZIONI DI COMPLEME.

Siano dati un problema primale e uno duale aventi come soluzioni, rispettivamente, $\bar{x}$ e $\bar{\lambda}$.Esse sono soluzioni solo se valgono le seguenti condizioni:

1. **VINCOLO DUALE SATURO (AMMIX)**
   $$\forall i \; : \; x_i > 0 \; \Rightarrow \; \bar{\lambda}^T \bar{a}_i = c_i$$
2. **COMPLEMENTARIETA' DUALE**
   $$\forall i \; : \; \bar{\lambda}^T \bar{a}_i < c_i \; \Rightarrow \; x_i = 0$$
3. **VINCOLO PRIMALE SATURO (AMMIX)**
   $$\forall j \; : \; \lambda_j > 0 \; \Rightarrow \; \bar{a}_j \bar{x} = b_j$$
4. **COMPLEMENTARIETA' PRIMALE**
   $$\forall j \; : \; \bar{a}_j \bar{x} > b_j \; \Rightarrow \; \lambda_j = 0$$

Per ogni **variabile del problema primale non nulla all'ottimo**, il corrispondente **vincolo del problema duale è soddisfatto all'uguaglianza** e **simmetricamente** per ogni variabile del problema duale non nulla all'ottimo, il corrispondente vincolo del problema primale è soddisfatto all'uguaglianza.

CONDIZIONI DI COMPLEME. λ LIBERA $\forall j$

Siano dati un problema primale e uno duale aventi come soluzioni, rispettivamente, $\overline{x}$ e $\overline{\lambda}$.Esse sono soluzioni solo se valgono le seguenti condizioni:

ANALOGO SENZA (3) (4)

1. **VINCOLO DUALE SATURO**

$$\forall i \ : \ x_i > 0 \ \Rightarrow \ \overline{\lambda}^T \overline{a}_i = c_i$$

2. **VINCOLO 2**

$$\forall i \ : \ \overline{\lambda}^T \overline{a}_i < c_i \ \Rightarrow \ x_i = 0$$

EX

Esempio 5.1.18 *Si consideri il problema di Programmazione Lineare*

$$\begin{cases} \min 3x_1 + 2x_2 + x_3 + 4x_4 \\ x_1 - 3x_3 + 2x_4 \geq 5 \\ 2x_1 + x_2 - x_3 \geq 3 \end{cases}.$$

Il problema duale associato è

$$\begin{cases} \max 5u_1 + 3u_2 \\ u_1 + 2u_2 = 3 \\ u_2 = 2 \\ -3u_1 - u_2 = 1 \\ 2u_1 = 4 \\ u_1 \geq 0, u_2 \geq 0 \end{cases}$$

Poiché il duale ha solo vincoli di uguaglianza, le condizioni di complementarità si riducono a $u^T(Ax - b) = 0$ *che in questo caso sono*

$$\begin{aligned} u_1(x_1 - 3x_3 + 2x_4 - 5) &= 0 \\ u_2(2x_1 + x_2 - x_3 - 3) &= 0. \end{aligned}$$

INTERPRETAZ. DUALITA'

Ad esempio, ad un modello per la pianficazione della produzione, i vincoli di un problema (primale) possono rappresentare una **limitazione dovuta alla limitata disponibilità delle risorse**; Un aumento di queste ultime può consentire un **aumento della produzione** quindi **anche del profitto**,
Uno dei possibili usi della dualità è quello di **rendere esplicito l'effetto dei cambiamenti nei vincoli sul valore della funzione obiettivo**. Questo perchè, come vedremo, le variabili duali possono essere anche interpretate come i **cosiddetti prezzi ombra** in quanto misurano i "costi" impliciti associati ai vincoli.

ANALISI DI SENSIBILITA'

*Per analisi di sensibilità si intende la **risposta delle soluzioni del problema alla perturbazione** di un vettore dei coefficienti di costo o dei termini noti.*

PERTURBAZIONE DEI TERMINI NOTI

*In questo caso si hanno **problemi di ammissibilità della soluzione***

$$\overline{b_{new}} = \overline{b} + \overline{\theta}$$
$$\overline{x_{new}} = B^{-1}\,\overline{b_{new}}$$

Verifico che: $x_{new_i} > 0 \ \forall i$

$$\Delta z = \overline{\lambda}^{T} \Delta \overline{b}$$
$$\lambda_i = \frac{\Delta z}{\Delta b_i}$$

$\lambda_i > 0$ **AUMENTO** b_i
$\lambda_i < 0$ **DIMINUISCO** b_i

1. **PERTURBAZIONE FISSA** $\overline{b}$
2. **PERTURBAZIONE PARAMETRICA** $\overline{b}$
3. **INTERVALLO DI STABILITA** $\overline{b}$
   Impongo che le soluzioni siano sempre **positive**

PERTURBAZIONE DEI COSTI

*In questo caso **la soluzione continuerà ad essere ammissibile** ma **può non essere ottima***

$$\overline{c_{new}} = \overline{c} + \overline{\theta}$$
$$\overline{r_{D_{new}}}^{T} = \overline{c_{D_{new}}}^{T} - \overline{c_{B_{new}}}^{T} B^{-1} D \geq 0$$

$$\overline{r_D}^{T} + k\,\overline{\theta_D}^{T} - k\,\overline{\theta_B}^{T} B^{-1} D \geq 0$$

1. **PERTURBAZIONE FISSA** $\overline{c}$
2. **PERTURBAZIONE PARAMETRICA** $\overline{c}$
3. **INTERVALLO DI STABILITA** $\overline{c}$

METODO DEL SIMPLESSO DUALE

*Il metodo parte da una **soluzione ammissibile duale** e cerca iterativamente una **soluzione ammissibile primale***

LAVORO SUL TABLEAU

CRITERIO DI ENTRATA

$$h = \arg\min\{\frac{\bar{c}_j}{|\bar{a}_{tj}|} : j \in \{1, \ldots, n\}, \bar{a}_{tj} < 0\}$$

$$\min 3x_1 + 4x_2 + 5x_3$$
s.t.
$$2x_1 + 2x_2 + x_3 \geq 6$$
$$x_1 + 2x_2 + 3x_3 \geq 5$$
$$x_1, x_2, x_3 \geq 0$$

$$\min 3x_1 + 4x_2 + 5x_3$$
s.t.
$$2x_1 + 2x_2 + x_3 - x_4 = 6$$
$$x_1 + 2x_2 + 3x_3 - x_5 = 5$$
$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

3	4	5	0	0	0
2	2	1	- 1	0	6
2	1	3	0	- 1	5

CAMBIO SEGNO ALLE RIGHE

3	4	5	0	0	0
- 2	- 2	- 1	1	0	- 6
- 2	- 1	- 3	0	1	- 5

scegliamo x_4 (riga $t = 1$) come var. uscente

DETERMINO ELEMENTO USCENTE

$$\min\{\frac{\bar{c}_1}{\bar{a}_{11}} = \frac{3}{2}, \quad \frac{\bar{c}_2}{\bar{a}_{12}} = 2, \quad \frac{\bar{c}_3}{\bar{a}_{13}} = 5\}$$

$PIVOT(1,1)$

0	1	7/2	3/2	0	-9
1	1	1/2	-1/2	0	3
0	- 1	- 5/2	-1/2	1	-2

l'unica riga con $\bar{b}_t < 0$ è $t = 2$, cioè, x_5 è la variabile entrante. Ripetendo il ragionamento precedente si individua l'elemento di pivot $(2, 2)$:

$PIVOT(2,2)$

0	0	1	1	1	-11
1	0	-2	-1	1	1
0	1	5/2	1/2	-1	2

soluzione $(1, 2, 0, 0, 0)$ ammissibile primale $\implies$ ottima

Complessità computazionale

DEFINIZIONE

Computational complexity theory is a branch of the theory of computation in theoretical computer science that focuses on classifying computational problems *according to their inherent difficulty, and relating those classes to each other.*

1. **PROBLEMA**
   Generica questione da risolvere, caratterizzata da diversi **parametri descritti** e da alcune **proprietà della soluzione.**

2. **ISTANZA**
   L'istanza di un problema corrisponde alla **specificazione dei valori dei parametri di quest'ultimo.**

3. **ALGORITMO**
   Procedura **a step finita** che porta alla soluzione di un problema

EFFICIENZA DI UN ALGORITMO

Con complessità di un algoritmo o efficienza di un algoritmo ci si riferisce alle risorse di calcolo richieste.
RISORSE TEMPO VELOCITA' ACCURATEZZA

SOLVIBILITA' DI UN PROBLEMA
Un problema è risolubile unicamente se è possibile risolverlo con **una macchina di Turing.**

TEOREMA CURCH-TURING
Qualsiasi **algoritmo è modellabile** con una macchina di Turing.

STABILITA' NUMERICA
Un algoritmo si dice numericamente stabile tanto più è **difficile perturbare il risultato di questo.**

MISURAZIONE RISORSE

*Per quel che riguarda la misurazione della risorsa tempo (spazio), data una **macchina di Turing M**, si dice che M opera in **tempo f(n)** se dato un input x **di lunghezza n**, la macchina M produce il **risultato in f(n) passi***

La funzione di complessità deve avere le seguenti caratteristiche, realmente non realizzabili:

1. **MONOTONA CRESCENTE**
2. **CALCOLABILE IN TEMPO E SPAZIO LIMITATO**
   Limitatamente alla funzione stessa

Dato che questo è impossibile, viene introdotta la nozione di **O-GRANDE**

$$f(n)=O(g(n)) \quad \equiv \quad \exists(n_0,c):\ c,n_0\geq 0\ \ \forall n>n_0\ \ f(n)\leq cg(n)$$

NOZIONI DI GRANDEZZA

1. OMEGA — CASO MIGLIORE

$$g(x)=\Omega(f(x)) \quad \equiv \quad \exists(n_0,c):\ c,n_0\geq 0\ \ \forall n>n_0\ \ g(n)\geq cf(n)$$

Ovvero **g(n) cresce non più lentamente di f(n)**; questa notazione è utile per valutare il caso ottimo di un algoritmo: se un algoritmo è **Ω(f(n)) significa che nel caso migliore richiede f(n) passi per essere risolto.**

2. THETA — CASO PEGGIORE

$$g(x)=\Theta(f(x)) \quad \equiv \quad g(n)\in O(f(n))\ \ e\ \ g(n)\in\Theta(f(n))$$

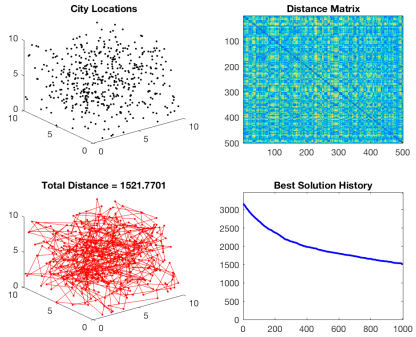
Ovvero **g(n) cresce al più al pari f(n)**; questo significa che non ci sono particolari variazioni tra i due algoritmi.

TSP PROBLEM

*Given a **list of cities and the distances** between each pair of cities, what is the **shortest possible route** that visits each city exactly once and returns to the origin city?*

$$x_{ij}=\begin{cases} 1 \text{ city } i \text{ to city } j \\ 0 \text{ otherwise} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n} c_{ij}x_{ij} \quad c:\ distance \\ & 0\leq x_{ij}\leq 1 \qquad i,j=1,,n; \\ & u_i\in Z \qquad i=1,,n; \\ & \sum_{i=1,i\neq j}^{n} x_{ij}=1 \qquad j=1,,n; \\ & \sum_{j=1,j\neq i}^{n} x_{ij}=1 \qquad i=1,,n; \\ & u_i-u_j+nx_{ij}\leq n-1 \qquad 2\leq i\neq j\leq n. \end{aligned}$$



LIMITI SUPERIORI E INFERIORI

*Al fine di classificare un algoritmo, è necessario porre dei limiti **superiori ed inferiori** alle risorse temporali dello stesso*

1. BEST CASE

La complessità associata alla risoluzione di un **input ben formattato di lunghezza n**

2. WORST CASE

La complessità associata alla risoluzione di un **input mal formattato di lunghezza n**

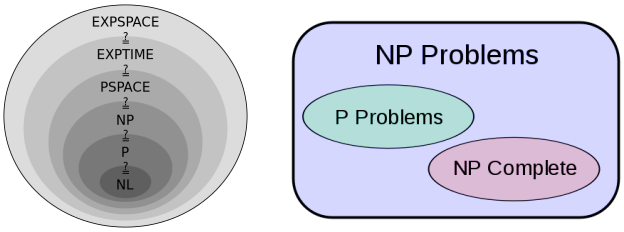
QUICK SORT

Algoritmo di ordinamento di una sequenza di numeri identificati da un indice.

WORST: $O(n^2)$ **BEST:**$\omega(n^2)$

CLASSE DI COMPLESSITA'

Una classe di complessità è un insieme di problemi di complessità analoga.



LE CLASSI

1. CLASSE L

L $= \textbf{SPACE}(log(n))$

Macchina di turing deterministica che opera in spazio

2. CLASSE NL

NL $= \textbf{NSPACE}(log(n))$

Macchina di turing **non deterministica** che opera in spazio

3. CLASSE P

P $= \cup_{k>0} \textbf{TIME}(n^k)$

Per risolvere i problemi appartenenti alle classi fin qui elencate sono noti algoritmi che **terminano in tempo polinomiale rispetto alla dimensione dei dati.**

EX: quicksort, cammino minimo, flusso minimo

4. CLASSE NP

NP $= \cup_{k>0} \textbf{NTIME}(n^k)$

Per questi problemi sono noti algoritmi che terminano in un numero di passi **polinomiale rispetto alla dimensione dei dati** nel caso si possa utilizzare un numero indeterminato di macchine in parallelo, o nel caso si utilizzi una **macchina di Turing non deterministica**

$$P \subseteq NP$$

$$P ?= NP \quad V \quad P \subset ?NP$$

NP-HARD

*Un problema si dice di classe NP-HARD se è **almeno difficile quanto il più difficile** problema in NP*

$$\forall \quad \Pi' \in NP \quad \Rightarrow \quad \Pi' \leq \log \Pi$$

Da questa definizione si ricava che i problemi **NP-difficili sono non meno difficili dei problemi NP-completi**, che a loro volta sono per definizione i più difficili delle classi P/NP.

NP-COMPLETE

*i problemi NP-completi sono **i più difficili problemi nella classe NP** ,nel senso che, se si trovasse un algoritmo in grado di risolvere "velocemente" un qualsiasi problema NP-completo, allora si potrebbe **usarlo per risolvere "velocemente" ogni problema in NP.***

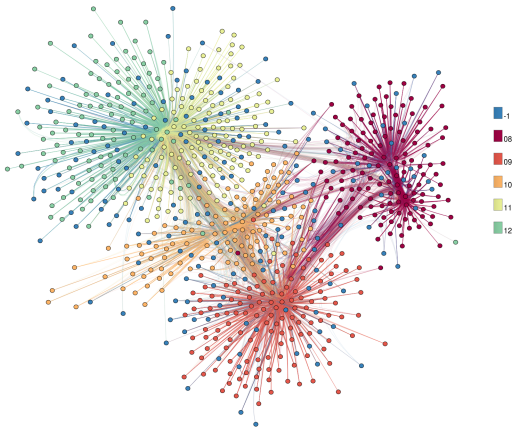
$$\forall \quad \Pi' \in NP \quad \Rightarrow \quad \Pi' \leq \log \Pi \quad \textbf{e} \quad \Pi \in NP$$

Teoria dei grafi

GRAFO

*Un grafo è un insieme di elementi detti **nodi o vertici** che possono essere collegati fra loro da linee chiamate **archi o lati o spigoli.***

$$G = (N, A)$$



PUO' ESSERE ORIENTATO

CONNESSO: Se esiste una catena tra ogni coppia di nodi

CATENA

*Dato un grafo $G = (N, A)$ e due nodi $i, j \subset N$ si definisce catena la **sequenza di archi che collega i a j***

$$(i, k_1), (k_1, k_2), \ldots, (k_n, j)$$

PUO' ESSERE ORIENTATA: CAMMINO

CICLO: Catena chiusa
CIRCUITO: Catena orientata (cammino) chiusa

ALBERO RICOPRENTE

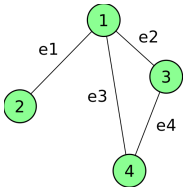
Un albero ricoprente è un grafo che contiene tutti i nodi

$$G = (N, A) \,|\, G' = (N', A') => N = N'$$

MATRICE DI INCIDENZA

La matrice di indicenza è una matriche che ha n righe quanti i nodi, m colonne quanti gli archi, il cui generico elemento c_{ik} vale

$$c_{ik} = \begin{cases} +1 & \text{arco K uscente dal nodo i} \\ -1 & \text{arco K entrante dal nodo i} \\ 0 & \end{cases}$$



$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

MATRICE DI ADIACENZA

La matrice di adiacenza nodi nodi è una matrice $n \cdot n$ dove n è il numero di nodi, avente come generico elemento c_{ij} ***costo***

$$d_{ij} = \begin{cases} 0 \\ \Re \\ \infty \end{cases}$$

PROCEDURA AD ALBERO

Determinare l'esistenza o meno di un cammino tra i nodi i, j

1. **ETICHETTARE IL NODO i**
   Tutti gli altri nodi non sono etichettati

2. **RICORSIONE**
   Per ogni nodo k **etichettato ma non esaminato**, esaminare il nodo k individuando tutti i nodi **non etichettati raggiungibili con un solo arco. Etichettare questi nodi con k** e marcare come esaminati i nodi di partenza.

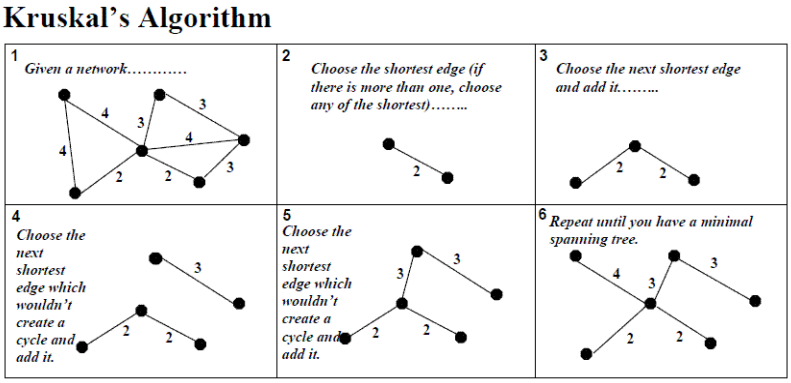
3. **CONTROLLO**
   SE: i nodo j è etichettato **STOP|RISOLTO ED ESISTENTE**
   SE: nessuno dei nodi non esaminato è etichettabile **STOP|NON RISOLTO**
   ALTRIMENTI: **Tornare allo step 2**

ALGORITMO KRUSKAL

Dato un generico grafo $G = (N, A)$ trovare un albero ricoprente tale che il suo costo sia minimo.
CC: $O(m \log m + mn)$

ALBERO RICOPRENTE

1. **ORDINO PER COSTO MINIMO**
2. **SELEZIONARE SPIGOLO COSTO MINIMO**
   Arbitrariamente scelto
3. **SELEZIONARE ARCO COSTO MINIMO**
   Connesso al primo dato, in modo tale che **non formino cicli. Controllo che gli estremi dell'arco non siano già stati inclusi**



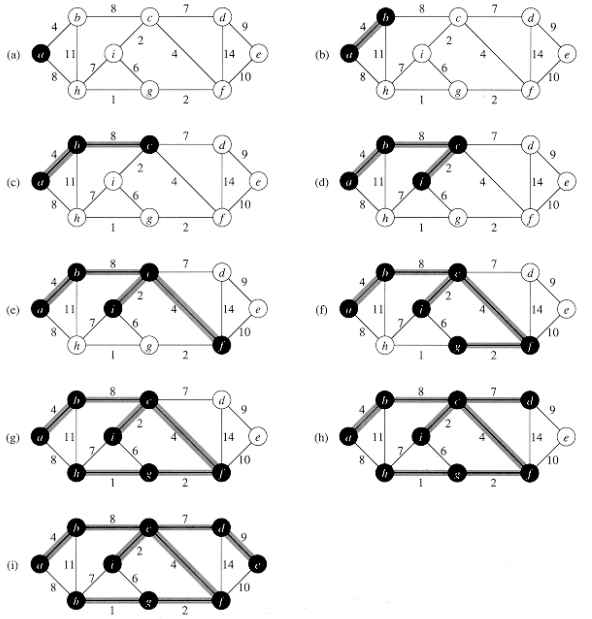
ALGORITMO DI PRIM

Dato un generico grafo $G = (N, A)$ trovare un albero ricoprente tale che il suo costo sia minimo.

CC: $O(n^2)$

ALBERO RICOPRENTE

1. **SELEZIONARE UN VERTICE i E AGGIUNGERLO ALL'INSIEME T**
   Rimuovere la riga di i
2. **RICERCARE**
   Nella colonna dei **vertici appartenenti a T** il minor valore numerico. Attenzione all'**ISOCOSTO MINIMO;** Aggiungerlo alla **lista dei costi** e tornare al **passo 1.** **FARE SEMPRE CONTROLLO GRAFICO**



GRAFO DI FLUSSO

*Dato un generico grafo $G = (N, A)$ esso si dice **di flusso** se possiede le seguenti caratteristiche:*

1. **ORIENTATO E CONNESSO**
2. **COSTO c_{ij}**
   Costo di trasporto associato all'arco (i, j)
3. **FLUSSO x_{ij}**
   Associato ad ogni arco (i, j)
4. **NODI POZZO (P) E SORGENTE (S)**
   Tali che abbiano solo archi **entranti o uscenti.**
5. **BILANCIAMENTO RETE**
   $\sum b_i = 0$
   In base al segno del b_i locale i nodi si dicono **sorgente, destinazione o transito.**

TEOREMA SULLA MATRICE A

Sia data una rete composta da n nodi e la relativa matrice di incidenza A. Allora il rango di A **è pari a** $n - 1$ **ovvero** $\rho(A) = n - 1$

TEOREMA SULLA MATRICE RICOPRENTE

Data una rete e la relativa matrice di incidenza A**,** ogni **base di A è albero ricoprente della rete**

SIMPLESSO PER IL FLUSSO

1. **TROVO UNA SOLUZIONE DI BASE**
   Primo passo dell'algoritmo del simplesso

2. **CALCOLO DELLE VARIABILI DUALI**
   $\lambda_i - \lambda_j = c_{ij}$ per ogni arco con costo
   Dato che il sistema ha $(n-1)$ parametri liberi, **fisso arbitrariamente un valore di** λ_i

3. **CALCOLO COSTI RIDOTTI NON IN BASE**
   $r_{ij} = c_{ij} - (\lambda_i - \lambda_j)$
   SE $r_{ij} \geq 0$ **STOP: OTTIMO**

4. **ENTRA IN BASE**
   Un arco a costo $r_{ij} < 0$, facendo uscire l'arco che **azzera il flusso netto.**

CAMMINO MINIMO

*Nella teoria dei grafi, il **cammino minimo**, o **shortest path**, tra due vertici detti m **iniziale e** i **finale** un grafo è quel percorso che collega i suddetti vertici e che **minimizza la somma dei costi** associati all'attraversamento di ciascun arco*

$$G = (N, A)$$

1. l_{ij} **LUNGHEZZA ARCO** ij
2. $P(i)$ **PREDECESSORE**
   Predecessore del nodo i nel cammino minimo da 1 a i
3. $n = |N|$ **NUMERO NODI**
4. Γ_i **INSIEME SUCCESSORI**
   Del nodo generico i nel grafo
5. Γ_i^{-1} **INSIEME PREDECESSORI**
   Del nodo generico i nel grafo
6. $\Pi(i)$ **CAMMINO MINIMO**

ALGORITMO DIJKSTRA

Svuotare iterativamente un insieme di nodi S
CC: $O(n + n^2 + m) = O(n^2 + m)$

1. **INIZIALIZZO**
   $\overline{S} = \{2,3,4,...,n\}$ e $\Pi(1) = 0$
   SE $i \subset \Gamma(1)$ => $\Pi(i) = l_{1i}$
   ALTRIMENTI $\Pi(i) = \infty$
   $P(i) = 1 \forall i \in \Gamma(1)$

2. **SELEZIONO NODO** j: $\min_{i \in \overline{S}} \{\Pi(i)\}$
   $\overline{S} = \overline{S} - j$ Aggiorno l'insieme $\overline{S}$
   SE $\overline{S} = \emptyset$ **STOP**
   ALTRIMENTI PASSO 3

3. **SELEZIONO SUCCESSORE**
   $\forall i \in \Gamma(j) \cap \overline{S} \quad \Pi(i) = min\{\Pi(i), \Pi(j) + lji\}$
   SE $\Pi(i) = \Pi(j) + lji \qquad P(i) = j;$ **PASSO 2**

Esempio 5 *Si applichi l'algoritmo di Dijkstra per il calcolo dei cammini minimi nel grafo in Figura 5 a partire dal nodo 1.*

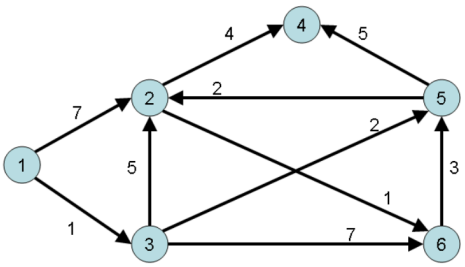


Figure 5: Grafo per l'Esempio 5.

Anche in questo caso possiamo organizzare i calcoli delle varie iterazioni in una tabella.

it.	nodo 1	nodo 2	nodo 3	nodo 4	nodo 5	nodo 6	$\bar{S}$	$\hat{v}$
0	$0^{*}_{(\wedge)}$	$7_{(1)}$	$1_{(1)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$2,3,4,5,6$	
1		$6_{(3)}$	*		$3_{(3)}$	$8_{(3)}$	$2,4,5,6$	3
2		$5_{(5)}$		$8_{(5)}$	*		$2,4,6$	5
3		*		–		$6_{(2)}$	$4,6$	2
4					×	*	4	6
5				*			$\emptyset$	4

$*$: etichetta fissata.
$-$: etichetta controllata ma non aggiornata.
$\times$: etichetta non controllata perché il nodo è già fissato.

ALGORITMO BELLMAN-FORD

Aggiornare il ***vettore dei cammini minimi*** *in una data iterazione (cammini minimi con al più k archi) sulla base dei risultati dell'iterazione precedente (cammini minimi con al più $(k-1)$ archi)*

CC: $O(nm)$

1. **INIZIALIZZO**
   $\Pi^0(1) = 0 \quad \Pi^0(i) = \infty \forall i \neq 1, k = 1$

2. **ITERAZIONE K-ESIMA**
   $$\Pi^k(i) = \min\left[\Pi^{k-1}(i), \min_{j \in \Gamma_i^{-1}}\left(\Pi^{k-1}(j) + l_{ji}\right)\right] \quad \forall i$$

3. **DISCRIMINO**
   SE $\Pi^k(i) = \Pi^{k-1}(i) \quad \forall\, nodo\ i$ **STOP**
   SE $k \leq (n-1) \quad k = k+1$ **PASSO 2**
   SE $k = n$ **STOP: Circuito di lunghezza negativa**

Esempio 2 *Si risolva il problema dei cammini minimi con origine nel nodo* 1 *per il grafo in Figura 3.*

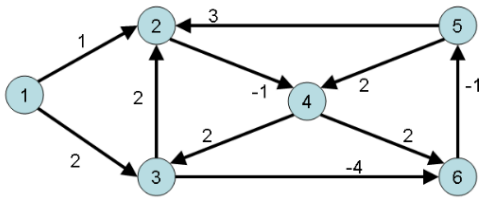


Figure 3: Grafo per l'Esempio 2.

iterazione	nodo 1	nodo 2	nodo 3	nodo 4	nodo 5	nodo 6	`Aggiornati`
$h=0$	$0_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	1
$h=1$	$0_{(\wedge)}$	$+1_{(1)}$	$+2_{(1)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	2, 3
$h=2$	$0_{(\wedge)}$	$+1_{(1)}$	$+2_{(1)}$	$0_{(2)}$	$+\infty_{(\wedge)}$	$-2_{(3)}$	4, 6
$h=3$	$0_{(\wedge)}$	$+1_{(1)}$	$+2_{(1)}$	$0_{(2)}$	$-3_{(6)}$	$-2_{(3)}$	5
$h=4$	$0_{(\wedge)}$	$0_{(5)}$	$+2_{(1)}$	$-1_{(5)}$	$-3_{(6)}$	$-2_{(3)}$	2, 4
$h=5$	$0_{(\wedge)}$	$0_{(5)}$	$+1_{(4)}$	$-1_{(5)}$	$-3_{(6)}$	$-2_{(3)}$	3
$h=6$	$0_{(\wedge)}$	$0_{(5)}$	$+1_{(4)}$	$-1_{(5)}$	$-3_{(6)}$	$-3_{(3)}$	6

L'algoritmo ha terminato con `flag_aggiornato = true` e pertanto esiste un ciclo negativo. Tale ciclo è individuato partendo dal nodo 6, (uno dei nodi) che ha subito un aggiornamento all'ultima iterazione, e procedendo a ritroso attraverso i puntatori $p(\cdot)$: $6 \leftarrow 3 \leftarrow 4 \leftarrow 5 \leftarrow 6$ (di costo $-1+2+2-4=-1$).

Problema del trasporto

DEFINIZIONE

I problemi di trasporto possono essere considerati come una generalizzazione dei problemi di assegnazione

$$m\ origini \quad n\ destinazioni$$

$$\min Z = \sum_{i,j=1}^{ij} c_{ij} x_{ij}$$

$$\sum_{j=1}^{j} x_{ij} = d_i \qquad \sum_{i=1}^{i} x_{ij} = r_i \qquad x_{ij} \geq 0 \ \forall i, j$$

1. **M ORIGINI — N DESTINAZIONI**
2. **COSTO** c_{ij}
   Che modella il costo del trasporto tra sorgente e destinazione
   $c : R2 \times R2 \rightarrow [0,\infty)$
3. **DISPONIBILITA'** D_i
   Di materia nelle origini
4. **RICHIESTA** R_j
   Di materia nelle destinazioni
5. $\sum Rj = \sum D_i$
6. **REGOLE DEL GIOCO**
   La quantità di merce spedita da ogni origine non può superare la disponibilità in quella origine;La quantità di merce ricevuta da ogni centro di consumo non può superare la sua richiesta
7. X_{ij}
   la quantità di merce spedita dal generico centro "i" di produzione, al generico centro "j" di consumo.

TABELLA DELLE ALLOCAZIONI

	D_1	D_2	...	D_j	...	D_n	
O_1	x_{11}	x_{12}	...	x_{1j}	...	x_{1n}	a_1
O_2	x_{21}	x_{22}	...	x_{2j}	...	x_{2n}	a_2
...	...	...	...	...	...	...	...
O_i	x_{i1}	x_{i2}	...	x_{ij}	...	x_{in}	a_i
...	...	...	...	...	...	...	...
O_m	x_{m1}	x_{m2}	...	x_{mj}	...	x_{mn}	a_m
	b_1	b_2	...	b_j	...	b_n	$\Sigma a_i = \Sigma b_j$

PROPRIETA'

ESISTE SEMPRE UNA SOLUZIONE

Essendo il problema dei trasporti un particolare caso del **problema di flusso minimo,** problema di ottimizzazione, **limitato e ammettente una soluzione di base**, esso ammette per forza **una soluzione ottima.**

DIPENDENZA LINEARE

Degli $m + n$ vincoli, **non tutti sono indipendenti,** quindi posso eliminare **uno di questi.**

TRIANGOLARITA' BASI

Ogni base del problema **è sempre triangolare.** Per determinare se una matrice è triangolare, elimino successivamente riga e colonna dove trovo **uno e un solo elemento per riga diverso da zero.**

LEGAME CON IL SIMPLESSO

$z = \lambda^T k \qquad k = [\overline{a} + \overline{b}] \qquad \lambda = [\overline{u} + \overline{v}]$

$k' = k + \delta \quad z' = \sum^{i} a_i u_i + \sum^{j} b_j v_j + \delta(u_s + v_s) \quad u_i + v_j$ **VARIAZIONE**

SOLUZIONE PARADOSSO DEL TRASPORTO

Effettuo **un'analisi di sensibilità sul problema tradotto in simplesso**, e scelgo di **aumentare di δ una variabile per la quale** $u_i - v_j < 0$

RICERCA SOLUZIONE DI BASE

1. **METODO NORD-OVEST**
   Alloco tutte le risorse disponibili alle destinazioni partendo **dall'angolo nord-ovest.**
2. **METODO DEI COSTI MINIMI (RIGA/COLONNA/TABELLA)**
   Alloco le risorse a partire dai valori di **costo più bassi.**

SOLUZIONE DEGENERE

E' possibile che sia necessario introdurre un'allocazione con **quantità zero,** per permettere la **risolubilità** $n_{BASE} = n + m - 1$

ALGORITMO DI DANTZIG-TRASPORTO

1. **DETERMINO COSTI INDIRETTI**
   $c_{ij} = u_i + v_j$ **per le variabili in base**
   Per determinare il valore delle variabili, pongo l'ultima variabile $v_j = 0$ **determinando automaticamente** u_i

2. **MATRICE DEI COSTI INDIRETTI**

$$\begin{array}{c} \\ u_1 \\ \dots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & \dots & v_n \end{array} \\ \begin{bmatrix} r_{11} & \dots & r_{1n} \\ \dots & \dots & \dots \\ r_{n1} & \dots & r_{nn} \end{bmatrix} \end{array} \qquad r_{ij} = \begin{cases} 0 & \text{BASE} \\ c_{ij} - (u_i + v_j) & \text{NO BASE} \end{cases}$$

3. **SE** $r_{ij} > 0 \forall i, j$ **STOP: OTTIMO**
   Altrimenti vado allo **Continuo verso STEP4**

4. **SELEZIONO COSTO INDIRETTO** $r_{ij} < 0$
   La variabile selezionata **entra in base.**

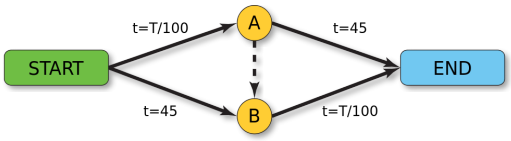
5. **VARIABILE USCENTE DALLA BASE**
   $\varepsilon = min\{(i,j) \subset B \ : \ (i,j) \ \ ha \ \ etichetta \ \ "-"\}$

   Si crea un **ciclo unico** di variabili sulla tabella x_{ij} tale che, per **non mutare i totali di riga e di colonna**, la **quantità aggiunta alla variabile non in base deve essere tolta alle variabili in base** e anzi una di esse deve alla fine avere valore nullo. **IL CICLO HA SEMPRE NUMERO PARI**
   $\varepsilon = min\{(i,j) \subset B \ : \ (i,j) \ \ ha \ \ etichetta \ \ "-"\}$

6. **AGGIORNO TABELLA DI ALLOCAZIONE**
   Con le quantità nuove. **Ritorno allo STEP2**

PARADOSSO DI BRAESS

*Il paradosso di Braess, dimostra che **l'apertura di una nuova strada** in una rete stradale **non implica obbligatoriamente un miglioramento del traffico** e che in determinate circostanze **può provocare anzi un aumento del tempo** medio di percorrenza.*



Metodi multicriteri

DEFINIZIONE

*L'obiettivo dei metodi multicriteri di Aiuto alla Decisione è quello di fornire ai **decisori strumenti che li aiutino** ad affrontare problemi decisionali caratterizzati da una **molteplicità di punti di vista significativi** e spesso da un limitato livello di strutturazione, in processi di decisione che si sviluppano in ambito organizzativo o multiorganizzativo.*

1. **AZIONE** A
   una rappresentazione di un possibile **contributo alla decisione globale**
   FINITO_STABILE | **EVOLUTIVO**

2. **CRITERIO** F
   Criterio **g è un modello** che permette di stabilire **relazioni di preferenza tra azioni**, coerenti con le preferenze dei decisori. Hanno un **verso di preferenza**
   SIGNIFICATIVI - COMUNI - ADATTI

3. **SCALA DI VALUTAZIONE** E
   Un criterio g è funzione dell'insieme A tale che: $\forall a \subset A => g(a) \subset E$
   dove E viene detto scala di valutazione

4. **COEFFICIENTE DI IMPORTANZA RELATIVO** w
   Peso del criterio rispetto alla decisione globale

ELECTRE

*ELECTRE è una **famiglia di metodi decisionali** multicriterio che ebbe origine in Europa nella metà degli anni 60.*

1. **NO TRANSITIVITA'**
   Ammettono **incoerenza** nel decisore
2. **NO COMPLETEZZA**
   Ammettono **incompatibilità**
3. **PESI DEL VALUTATORE**
   E' il valutatore che assegna il peso, quindi irrazionale
4. **EVIDENZIANO SURCLASSAMENTO**

1. **FASE 1**
   Si confrontano **a coppie le azioni su ogni criterio e si aggregano i risultati** ottenuti, mediante la costruzione di indici o l'applicazione di test che verificano la presenza di **condizioni di concordanza e di non discordanza**

2. **FASE 2**
   Si attiva la **procedura di classificazione delle azioni** relativa alla problematica in esame e alla 'regola' decisionale modellizzata.

ELECTRE II

*ELECTRE II utilizza **veri-criteri, quelli, cioè, in cui qualunque scarto di valutazione indica una preferenza** in senso stretto. Le scale possono essere cardinali od ordinali.*

1. **SURCLASSAMENTO CERTO**
   La funzione di criterio ha in output risultati certi

$$\delta(a,a')=\begin{cases}1 & a\ S\ a' \\ 0 & a\ \overline{S}\ a'\end{cases}$$

   Nel grafo di surclassamento compaiono gli **archi da a ad a' per cui δ(a,a')=1**

2. **LIVELLI DI SURCLASSAMENTO**
   P preferenza forte, **Q** preferenza debole, **I** indifferenza, **N** incomparabilità

3. **TEST DI CONCORDANZA - NON DISCORDANZA**
4. **FASE 2**

TEST DI CONCORDANZA

*Il test di concordanza **confronta a coppie tutte le azioni,** criterio per criterio, e suddivide, per ciascuna coppia di azioni, i criteri g_j in **tre sottoinsiemi***

$$J_+(a,a')=\{j\in J: g_j(a)\succ g_j(a')\}$$
$$J_=(a,a')=\{j\in J: g_j(a)= g_j(a')\}$$
$$J_-(a,a')=\{j\in J: g_j(a)\prec g_j(a')\}$$

$$P^{+-=}=\sum_{j=1}^{j\in J^{+-=}} P_j$$

INDICE DI CONCORDANZA
Parametro che rappresenta numericamente la **concordanza o maggioranza.**
Si possono scegliere diverse **soglie** c tra cui quelle **naturali**
$c_f = 3/4$ **SOGLIA FORTE** - $c_d = 2/3$ **SOGLIA DEBOLE**

$$c(a,a'))=\frac{P^+(a,a')+P^=(a,a')}{P}\geq C$$
$$\frac{P^+(a,a')}{P^-(a,a')}\geq 1$$

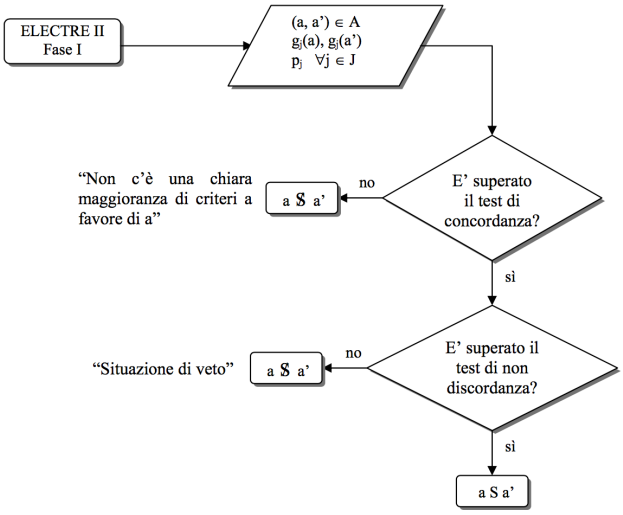
CRITERIO DI NON DISCORDANZA

*Il test di concordanza **confronta a coppie tutte le azioni,** criterio per criterio, e suddivide, per ciascuna coppia di azioni, i criteri g_j in **tre sottoinsiemi***

TEST NON SUPERATO SE

$$gj^*(a)=e \qquad gj^*(a')=e'$$
$$(e,e')\in Dj^*$$

INSIEME DI DISCORDANZA
Si dice insieme di discordanza DJ^* un insieme definito dal prodotto cartesiano ExE delle scale di valutazione.



FASE 2

Nella fase 2 si ordinano le azioni seguendo i surclassamenti ***forte e debole*** *e controllando che non ci siano* ***circuiti sul grafo***

A livello generale questa fase consta nella costruzione di **due grafi di preordinamento** $P^+(A)P^-(A)$**.**
Se i grafici sono **sufficientemente prossimi** si costruisce **il grafo finale.**

In caso il grafo **presenti loop,** si utilizza il **surclassamento debole**, cioè si ripete il **test di concordanza solo tra gli elementi 'a pari merito' rilassando la soglia** e sul sottografo di S si ripete la ricerca degli elementi *'non surclassati da nessun altro elemento'.* Tutti gli elementi con questa caratteristica vengono **attribuiti alla classe C**, gli altri rimangono in A.

1. **PRECEDENZA DI** a **su** a'
   Se a viene **prima di** a' in uno dei due preordini, ed è **ex-equo** nell'altro
2. **EX-EQUO DI** a **su** a'
   Solo se sono ex-equo in entrambi i grafi
3. **INCOMPARABILITA' DI** a **su** a'
   Se a viene **prima di** a' in uno dei due preordini, e **opposto** nell'altro.

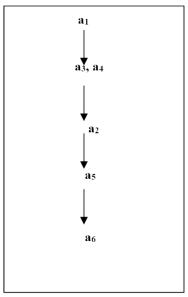
Tabella 1: Valutazioni e parametri

Criteri	Impiego di manodopera	Flessibilità geometrica	Flessibilità produttiva	Produzione integrata	Economico-finanziario
Pesi	0,20	0,28	0,04	0,20	0,28
Alternative					
a_1	Sì	Suff.	Bassa	Basso	4
a_2	No	Ottima	Bassa	Alto	3
a_3	No	Insuff.	Alta	Alto	2
a_4	No	Buona	Alta	Alto	1
Ins. Disc.					(1, 4)

Tabella 2: Fase I di ELECTRE II

	J_+	$J_=$	J_-	$P^+ \geq P^-$	$P^+ + P^=$	Veto	S
$a_1\ a_2$	5	3	1, 2, 4	no			
$a_1\ a_3$	2, 5	/	1, 3, 4	sì	0,56		
$a_1\ a_4$	5	/	1, 2, 3, 4	no			
$a_2\ a_1$	1, 2, 4	3	5	sì	0,72	no	
$a_2\ a_3$	2, 5	1, 4	3	sì	0,96		$a_2\ S\ a_3$
$a_2\ a_4$	2, 5	1, 4	3	sì	0,96		$a_2\ S\ a_4$
$a_3\ a_1$	1, 3, 4	/	2, 5	no		no	
$a_3\ a_2$	3	1, 4	2, 5	no		no	
$a_3\ a_4$	5	1, 3, 4	2	sì	0,72		
$a_4\ a_1$	1, 2, 3, 4	/	5	sì	0,72	sì	
$a_4\ a_2$	3	1, 4	2, 5	no		no	
$a_4\ a_3$	2	1, 3, 4	5	sì	0,72	no	

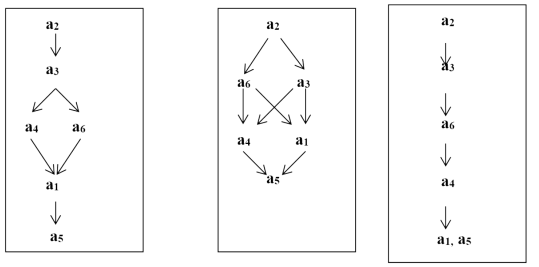
$P(A)^+ = \{a_1\} \succ \{a_3, a_4\} \succ \{a_2\} \succ \{a_5, a_6\}$
$P(A)^- = \{a_1\} \succ \{a_3, a_4\} \succ \{a_2\} \succ \{a_5\} \succ \{a_6\}$



ANALISI DI ROBUSTEZZA

Analisi di robustezza dei risultati a seguito della variazione (dell'ordine di centesimi) delle soglie di tolleranza.

$c_f = 0.74$ e $c_d = 0.63$ | $c_f = 0.75$ (o 0.74) $c_d = 0.66$ | $c_f = 0.75$(o 0.74) $c_d = 0.67$



Parto sempre dal grafo più semplice e ordinato, sovente quello con **le soglie naturali,** per poi confrontarlo **a coppie con gli altri.**

SOGLIA DI INDIFFERENZA

Nei veri criteri, l'indifferenza tra due azioni si ottiente unicamente se $g(a) = g(a')$ *ovvero* $aI_g a'$. *Spesso vi sono però* ***incertezze nei dati, piccole variazioni stocastiche che fanno ammettere una soglia di indifferenza*** q

$$\forall a, a' \subset A,\ if\ |g(a) - g(a')| > q\ \exists P_g\ \ elseif\ |g(a) - g(a')| <= q\ \exists I_g$$

PREUDOCRITERIO

Criterio, nel quale le soglie di indifferenza sono 2 ***s*** *e* ***q***

PRE - CRITERIO

Adottato nei casi in cui non vi è ***incertezza o indifferenza tra i dati*** *ma sul* ***criterio stesso.***

Vero criterio	Nessuna soglia	Preferenza stretta (se scarto tra valutazioni) Indifferenza (se valutazioni identiche)
Pre-criterio	Soglia di preferenza	Preferenza debole (per intervallo definito dalla soglia) Preferenza stretta Indifferenza (se valutazioni identiche)
Quasi-criterio	Soglia di indifferenza	Preferenza stretta Indifferenza (per intervallo definito dalla soglia)
Pseudo-criterio	Soglia di preferenza e di indifferenza	Preferenza debole (per intervallo definito dalla soglia) Indifferenza (per intervallo definito dalla soglia) Preferenza stretta

SCALE CARDINALI E ORDINALI

SCALA CARDINALE

Viene elaborato un ordinamento in base al risultato numerico restituito dal criterio sull'azione.

SCALA ORDINALE

L'ordinamento è relativo alle altre azioni. Spesso si tratta di valutazioni effettuate da esperti, o comunque di **conoscenza non strutturata e non traducibile in funzioni analitiche.**

ELECTRE 3

*ELECTRE III utilizza **qualsiasi criterio, con un ordinamento fuzzy***
*La funzione di criterio ha in output **risultati non certi [0,1]**= $\delta(a,a')$*
*Nel grafo di surclassamento compaiono **tutti gli archi** con i rispettivi valori*

1. **SOGLIE s E q**
2. **CREDIBILITA' SURCLASSAMENTO**
   Il risultato della fase I è il **grado di credibilità del surclassamento**, **non compatibile con un calcolo manuale.**
3. **INCERTEZZA**
   Può derivare da **dati, passaggio dati valutazioni, o incertezza dei decisori.**
4. **SOGLIA DI VETO**
   Per evidenziare il rischio associato a una decisione relativamente ad una azione con **valutazioni pessime in uno o più criteri**, mentre altre azioni presentano ottime valutazioni sugli stessi criteri. **PENALIZZA IL RISCHIO**
   Un veto ha effetto solo se **eccede dell'80% dalla scala di valutazione.**

ELECTRE TRI

*ELECTRE Tri utilizza **relazioni di surclassamento su criteri***

1. **RIFERIMENTI**
   Le azioni vengono confrontate con i riferimenti.
   Le classi **sono note indipendentemente dall'insieme A, e sono ordinate**

2. **PROBLEMA DELLA CERNITA**
   La problematica della cernita in presenza di riferimenti consiste **nell'attribuire a classi (o categorie) un insieme**, finito o evolutivo, **di azioni** di varia natura

3. **PROCEDURA DI ATTRIBUZIONE**
   l'insieme delle regole che, partendo da modelli espliciti di preferenza, permette **di assegnare tutte le azioni di A alle classi** precedentemente definite.
   *Se il profilo(cioè il vettore delle valutazioni) di un'azione è situato all'interno dei due profili limite associati ad una classe, allora **questa azione deve essere assegnata alla classe individuata dai profili***

6 REGOLE DI ATTRIBUZIONE

1. **UNICITA'**
2. **INDIPENDENZA**
3. **CONFORMITA'**
4. **OMOGENEITA'**
5. **MONOTONIA**
6. **STABILITA'**
   Il raggruppamento di due classi vicine non deve modificare l'attribuzione delle azioni alle altre classi.

Branch and Bound

DEFINIZIONE

*Il branch and bound è una tecnica generale per la risoluzione di problemi di ottimizzazione combinatoria, cioè problemi con **spazio di soluzioni X finito**, e si basa sulla **scomposizione del problema originale in sottoproblemi** più semplici da risolvere.*

CC Worst: $O(2^n)$

$$\begin{aligned} &\min / \max \quad c^T x \\ &A x = b \\ &x \in Z_+^n \end{aligned}$$

1. **BRANCHING**
   Si partizionano le soluzioni ammissibili del problema in **sottoinsiemi** (rami) da cui si diramano tutte le possibili **soluzioni del problema**

$$\bigcup_i S_i' = S \qquad S_i' \cap S_j' = \oslash \quad \forall i, j \quad i \neq j$$

2. **BOUNDING**
   Procedura di valutazione del nodo, tramite il **rilassamento delle soglie**, che mira ad escludere **alcuni rami dalle soluzioni**

REGOLA DI ESPLORAZIONE

1. **DEPTH-FIRST**
   Analizzo il **primo nodo** finchè non sono in grado di **chiuderlo.**

2. **BEST FIRST**
   Si sceglie il **nodo più promettente**, ossia il nodo con il **bound migliore** (*lower/upper)*

REGOLA DI BRANCH

*divide et **impera***

$$\bigcup_i S_i' = S \qquad S_i' \cap S_j' = \oslash \quad \forall i, j \quad i \neq j$$

1. **BINARIO CONTINUO**
   Divido lo spazio delle soluzioni, considerando che la soluzione deve essere intera e non contenuta nell'intervallo
   $x_i = p \quad \Rightarrow \quad x_i \leq [p] \quad x_i \geq [p]$

2. **LAGRANGIANO**
   Eliminio un vincolo e rilasso la soglia

CRITERI DI FATHOMING

1. INAMMISSIBILITA'
Dato il nodo corrente t, se i **vincoli del branch** sono incompatibili con quelli del problema iniziale, il **nodo viene chiuso.**

2. OTTIMALITA'
Dato il nodo corrente t, e il costo associato UB_t o LB_t, se questo è **minore o maggiore della soluzione ottima** $\overline{z}$ **aggiorno z** e **chiudo il nodo t.**

3. NON MIGLIORABILITA
MIN Dato il nodo corrente t e il lower bound LB_i, data $\overline{z}$ il valore della **migliore funzione obiettivo** chiudo il nodo se $LB_i \geq \overline{z}$
MAX Dato il nodo corrente t e l'upper boundUB_i, data $\overline{z}$ il valore della **migliore funzione obiettivo** chiudo il nodo se $UB_i \leq \overline{z}$

SCHEMA ALGORITMICO

1. TROVO UNA SOLUZIONE AMMISSIBILE
Denotata con x_{OPT} avente un valore di **upper bound corrente** dato da $UB^* = c^T x_{OPT}$

2. REGOLA DI ESPLORAZIONE
Viene definita una regola per l'esplorazione: *Depth first o Best first.*

3. REGOLA DI BRANCH
La regola di branch **genera il partizionamento**.
Generalmente si utilizza la regola di **rilassamento continuo:**

$$x_i = p \quad \Rightarrow \quad x_i \leq [p] \quad x_i \geq [p]$$

4. INIZIALIZZO LA CODA DI OPERAZIONI

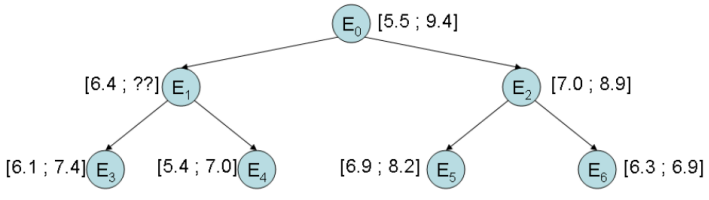
(A) RILASSAMENTO DELLA SOGLIA PER IL NODO N
Utilizzo un **criterio di rilassamento** definito nella regola di branch.

(B) CRITERIO DI FATHOMING
Utilizzo un **criterio di fathoming per valutare se continuare l'analisi del ramo** (*inammissibilità, soluzione intera o assenza di soluzione migliorante*).
Se viola un criterio torno allo **STEP1**

(C) IF $N = 0$
La soluzione è nel ramo avente soluzione migliore $UB^* = c^t x_{OPT}$.

EX

Esercizio 2 *Si consideri il seguente albero di B&B relativo ad un problema di massimo. Ad ogni nodo sono stati valutati sia un upper bound (UB) sia una soluzione ammissibile (SA), come riportato accanto ad ogni nodo, nel formato [SA;UB].*



- *Quale è un possibile valore per l'upper bound al nodo* E_1*?* $\{UB_1 \in [7.4\ 9.4]\}$[1]
- *Quale nodo sarà selezionato per il branching se si adotta una strategia di esplorazione* Best Bound First*?* $\{E_5\}$
- *Entro quale intervallo di valori è sicuramente compreso il valore ottimo della funzione obiettivo?* $\{[7.0\ 8.2]\}$
- *Si supponga di fare branching sul nodo* E_5*, ottenendo due nodi figli* E_7 *ed* E_8*. Sia inoltre il problema* P_8 *inammissibile* ($E_8 = \emptyset$)*. Dare un esempio di possibili valori SA e UB per il nodo* E_7 *che permettano di riconoscere subito una soluzione ottima, senza ulteriori operazioni di branching.* $\{SA_7 = LB_7 \in [7.4\ 8.2]\}$

TIPS

1. **RICONOSCERE IL TIPO DI PROBLEMA**
   MIN Se i **LB** sono **crescenti di padre in figlio.**
   MAX Se i **UB** sono **decrescenti di padre in figlio.**

2. **OTTIMO FUNZIONE OBIETTIVO**
   E' sempre associata al **LB e UB** della miglior soluzione trovata.

BOUND PER KP *Problema dello zaino risolto con rilassamento continuo della variabile*

CC Worst: $nlog(n)$

$$\max \sum_{i=1}^{n} p_i x_i$$

$$\sum_{i=1}^{n} w_i x_i \leq W \qquad x_i \in \{0,1\}$$

1. **ORDINO IN BASE A $\frac{p_i}{w_i}$ DECRESCENTE**

2. **ELEMENTO CRITICO $s = X_s$**
   s è il primo nell'ordine che supera la capienza dello zaino
   $s = \min_j \{j : \sum w_i > W\}$

3. **REGOLA DI BRANCH**
   Si fa sempre branch sulla **variabile critica X_s per BEST FIRST**

4. **INIZIALIZZO LA CODA DI OPERAZIONI**

   (A) DETERMINO LA SOGLIA DI RILASSAMENTO
   Tenendo conto che $W*$ indica **lo spazio residuo**

   $$x_i = \begin{cases} 1 & i<s \\ \frac{W*}{w_i} & i=s \\ 0 & i>s \end{cases} \qquad W* = W - \sum_{i=1}^{s-1} w_i$$

   $$LB = \sum_{i=0}^{i<x_s} p_i$$

   UB MARTELLO DI TOOTH

   $$UB^{tooth} = \sum_{i=1}^{s-1} p_i + \max\{W* \frac{p_{s+1}}{w_{s+1}}, \quad p_s - (w_s \quad - W*) \frac{p_{s+1}}{w_{s+1}}\}$$

   UB RILASSAMENTO CONTINUO $$UB = \sum_{i=0}^{j<x_s} p_i + R_{es} * p_s$$

   (B) CRITERIO DI FATHOMING
   - **A** La variabile critica satura la capacità disponibile $W*$
   - **B** Non ammissibilità
   - **C** Soluzione non migliorante

   (C) IF $N = 0$
   La soluzione è nel ramo avente soluzione migliore $UB* = c^t x_{OPT}$.

Esempio 3 *Si risolva con il metodo del Branch-and-Bound il seguente problema dello zaino* $0/1$*:*

$$\begin{array}{rrrrrrrrrrrrrrrrrrrr} \max & 36 & x_1 & + & 15 & x_2 & + & 3 & x_3 & + & 5 & x_4 & + & 11 & x_5 & + & 30 & x_6 & & \\ s.t. & 12 & x_1 & + & 6 & x_2 & + & 2 & x_3 & + & 3 & x_4 & + & 5 & x_5 & + & 9 & x_6 & \leq & 17 \\ & & x_1 & , & & x_2 & , & & x_3 & , & & x_4 & , & & x_5 & , & & x_6 & \in & \{0,1\} \end{array}$$

Innanzitutto calcoliamo il valore dei rapporti p_i/w_i e ordiniamo in maniera decrescente:

$$\frac{30}{6} \geq \frac{36}{12} \geq \frac{15}{6} \geq \frac{11}{5} \geq \frac{5}{3} \geq \frac{3}{2}$$

L'ordine delle variabili è pertanto:

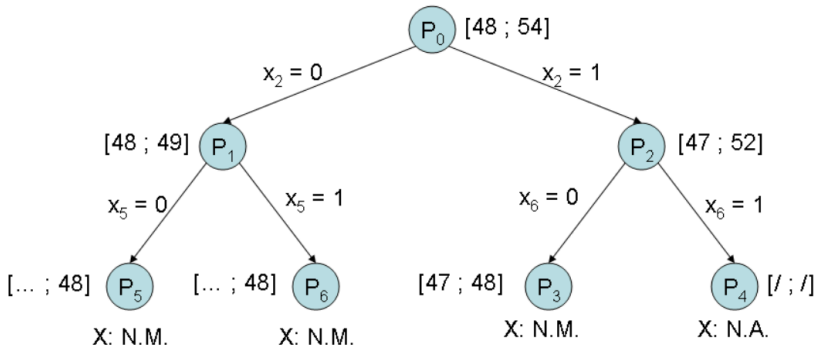
$$x_6 \rightarrow x_1 \rightarrow x_2 \rightarrow x_5 \rightarrow x_4 \rightarrow x_3$$

Per praticità, riscriviamo il problema con le variabili ordinate:

$$\begin{array}{rrrrrrrrrrrrrrrrrrrr} \max & 30 & x_6 & + & 36 & x_1 & + & 15 & x_2 & + & 11 & x_5 & + & 5 & x_4 & + & 3 & x_3 & & \\ s.t. & 9 & x_6 & + & 12 & x_1 & + & 6 & x_2 & + & 5 & x_5 & + & 3 & x_4 & + & 2 & x_3 & \leq & 17 \\ & & x_6 & , & & x_1 & , & & x_2 & , & & x_5 & , & & x_4 & , & & x_3 & \in & \{0,1\} \end{array}$$

Otteniamo il seguente albero di Branch-and-Bound (etichette nel formato [SA;UB]).

Otteniamo il seguente albero di Branch-and-Bound (etichette nel formato [SA;UB]).



Nodo P0:
$x_6 = 1 \quad \bar{W} = 8$
$x_1 = \frac{8}{12} \quad \bar{W} = 0 \quad x_1$ critica

$UB_0 = 30 + \frac{8}{12}36 = 54$

$SA_0 = 30_{(x_6)} + 15_{(x_2)} + 3_{(x_3)} = 48$

Generazione e valutazione dei figli di P_0:

Nodo P1:
$x_1 = 0 \quad \bar{W} = 17$

$x_6 = 1 \quad \bar{W} = 8$
$x_2 = 1 \quad \bar{W} = 2$
$x_5 = \frac{2}{5} \quad \bar{W} = 0 \quad x_5$ critica

$UB_1 = 0 + 30 + 15 + \frac{2}{5}11 = 49.4$

Osserviamo che, essendo i coefficienti della funzione obiettivo tutti interi, non è possibile ottenere valori frazionari della funzione obiettivo stessa, in corrispondenza di soluzioni ammissibili. $UB_1 = 49.4$ (49.4 è il valore del rilassamento continuo) ci dice che il valore della funzione obiettivo, una volta fissato $x_1 = 0$, non può essere superiore a 49.4. Per l'interezza dei coefficienti p_i, possiamo migliorare tale upper bound con il valore $\lfloor 49.4 \rfloor = 49$. Tale valore è un valore più stringente ed è da preferire. In generale

Se z_R^* e il valore ottimo del rilassamento continuo e i coefficienti (e le variabili)

della funzione obiettivo di massimo sono tutti interi, allora si può considerare come upper bound il valore $UB = \lfloor z_R^* \rfloor$. (Analogamente, per problemi di minimo con le stesse caratteristiche, $LB = \lceil z_R^* \rceil$.)

$SA_1 = 30_{(x_6)} + 15_{(x_2)} + 3_{(x_3)} = 48$

Nodo P2:

$x_1 = 1 \quad \bar{W} = 5$

$x_6 = \frac{5}{9} \quad \bar{W} = 0 \quad x_6$ critica

$UB_2 = 36 + \lceil \frac{5}{9} 30 \rceil = 52$

$SA_2 = 36 + 11_{(x_5)} = 47$

Adottando una strategia Best Bound First, effettuiamo il branch del nodo P_2.

Nodo P3:

$x_1 = 1 \quad \bar{W} = 5$

$x_6 = 0 \quad \bar{W} = 5$

$x_2 = \frac{5}{6} \quad \bar{W} = 0 \quad x_2$ critica

$UB_3 = 36 + \lceil \frac{5}{6} 15 \rceil = 48$

$SA_3 = 36 + 11_{(x_5)} = 47$

$UB_3 = 48 \geq 48$ (valore della migliore soluzione disponibile): chiudo P_3 perché non migliorante.

Nodo P4:

$x_1 = 1 \quad \bar{W} = 5$

$x_6 = 1 \quad \bar{W} = -4 \quad$ non ammissibile

Chiudo P_4 per non ammissibilità.

Rimane aperto ancora il nodo P_1, sul quale faccio branch.

Nodo P5:

$x_1 = 0 \quad \bar{W} = 17$

$x_5 = 0 \quad \bar{W} = 17$

$x_6 = 1 \quad \bar{W} = 8$

$x_2 = 1 \quad \bar{W} = 2$

$x_4 = \frac{2}{3} \quad \bar{W} = 0 \quad x_5$ critica

$UB_5 = 0 + 0 + 30 + 15 \lceil \frac{2}{3} 5 \rceil = 48$

$UB_5 = 48 \geq 48$ (valore della migliore soluzione disponibile): chiudo P_5 perché non migliorante (e non valuto la soluzione ammissibile corrispondente, che non migliorerà sicuramente quella corrente).

Nodo P6:

$x_1 = 0 \quad \bar{W} = 17$

$x_5 = 1 \quad \bar{W} = 12$

$x_6 = 1 \quad \bar{W} = 3$

$x_2 = \frac{3}{6} \quad \bar{W} = 0 \quad x_5$ critica

$UB_6 = 0 + 11 + 30 + \lceil \frac{3}{6} 15 \rceil = 48$

$UB_6 = 48 \geq 48$ (valore della migliore soluzione disponibile): chiudo P_6 perché non migliorante (e non valuto la soluzione ammissibile corrispondente, che non migliorerà sicuramente quella corrente).

A questo punto tutti i nodi sono chiusi e il valore ottimo della funzione obiettivo è 48. Una soluzione ottima è quella trovata come soluzione ammissibile al nodo P_0, e cioè: $x_6 = x_2 = x_3 = 1$, $x_1 = x_4 = x_5 = 0$.

Prontuario di calcolo

Simplesso

ITERAZIONE	OPERAZIONE
PROBLEMA ARTIFICIALE	sommando y_i **ad ogni vincolo e ponendo** $minK = \sum^{i} y_i \quad y_i \geq 0 \forall i.$ **ANMMISSIBILE**: $z = 0$ **INAMMISSIBILE**: $z > 0$
$B \mid D \mid B^{-1}$	$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad B^{-1} = \frac{1}{DET(B)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$
$\overline{\lambda^T} X_b$	$\overline{\lambda^T} = c_b^T B^{-1} \quad x_b = B^{-1} b$
$\overline{r_D}^T$	$r^T = c_d^T - \lambda^T D$ **ENTRANTE**: Regola di **Bland** *(indice minore)*
y_q e θ	$y_q = B^{-1} A_a$ Determina l'uscente $min\{\frac{x_b}{B^{-1}a_{in}}\}$ con $B^{-1}a_{in} > 0$. **PROBLEMA ILLIMITATO** $B^{-1}a_{in} \leq 0$
z	$z = c^T x = c_b^T x_b$

Varie sul simplesso

	DESCRIZIONE
VARIAZIONE COSTI	**OTTIMALITA'** Controllo che i **costi ridotti siano positivi** $r_d \geq 0$ $c = c + \theta \quad \lambda^T = c_{B'}^T B^{-1}$ $r_d = c_d^T - \lambda^t D \geq 0$ **ENTRATA IN BASE** x_i Considero il **costo ridotto negativo** $r_d = c_{xi} - \lambda^T a_{xi} < 0$
VARIAZIONE T. NOTI	**AMMISSIBILITA'** Controllo che le **variabili di base siano positive** $x_i \geq 0$ $b = b + \theta \quad (\Delta z = \lambda^T \theta)$ $x = B^{-1} b \geq 0$ **MIGLIORAMENTO F.O.** Considero il **problema duale** $z_0 = c^T x = \lambda^T b$
DETERMINARE TUTTE LE BASI	Combinazioni basi delle **colonne dei vincoli. (Oppure DET + CRAMER)** **NON E'UNA BASE:** Variabili linearmente dipendenti **BAE NON AMMISIBILE:** $X_B < 0$
SIMPLESS DUALE	Tableau con **vincoli di segno opposto.** **STOP:** $b_i \geq 0$ **ESCE** $min\{b_i\}$ **ENTRA** $min\{\frac{c_{ij}}{\lvert a_{i_{ext}j} \rvert}\}$
COMPLEMENTARIETA'	Per ogni **variabile del problema primale non nulla all'ottimo**, il corrispondente **vincolo del problema duale è soddisfatto all'uguaglianza** e **simmetricamente** $\sum (VINCOLO_i^p - b_i^p)\lambda_i = 0 \quad \sum (VINCOLO_i^d - b_i^d)x_i = 0$

	DESCRIZIONE
PRIMALE DUALE	

	PRIMALE	DUALE	
	$\min c^T x$	$\max b^T u$	
VINCOLI	$= b_i, \quad i \in I$ $\geq b_i, \quad i \in J$	$u_i, \quad i \in I$, libere $u_i, \quad i \in J, u_i \geq 0$	VARIABILI
VARIABILI	$x_j \geq 0, \quad j \in M$ $x_j, \quad j \in N$ libere	$\leq c_j, \quad j \in M$ $= c_j, \quad j \in N$	VINCOLI

Trasporti

ITERAZIONE	OPERAZIONE
BILANCIO IL PROBLEMA	$\sum b_i = \sum a_i$ Inserisco S/O con $c_{ij} = 0$
MOLTIPLICATORI COSTI RIDOTTI	u_i / c_{ij} … / v_j 0 $v_1 \ \dots \ v_n$ $u_1 \begin{bmatrix} r_{11} \dots r_{1n} \\ \dots \ \dots \ \dots \\ r_{n1} \dots r_{nn} \end{bmatrix}$ u_n $r_{ij} = \begin{cases} 0 & \text{BASE} \\ c_{ij} - (u_i + v_j) & \text{NO BASE} \end{cases}$
CONTROLLO COSTI	SE $r_{ij} > 0 \forall i, j$ **STOP: OTTIMO** Altrimenti **CONTINUO**
CAMBIO BASE	$r_{ij} < 0$ **ENTRA IN BASE** **CICLO UNICO:** sulla tabella x_{ij} tale che, per **non mutare i totali di R/C** $\varepsilon = min\{x_{ij} : (i,j) \quad Segno-\}$ **RICALCOLO I COSTI RIDOTTI**
PARADOSSO TRASPORTI	**LEGAME CON IL SIMPLESSO** $z = \lambda^T k \qquad k = [\overline{a} + \overline{b}] \qquad \lambda = [\overline{u} + \overline{v}]$ $k' = k + \delta \quad z' = \sum^{i} a_i u_i + \sum^{j} b_j v_j + \delta(u_s + v_s)$ $u_i + v_j < 0$ **VARIAZIONE** $\Delta_{max} = min\{x_{ij}\}$ **DELLA SOL.OTTIMA**

Albero ricoprente

ALGORITMO	DESCRIZIONE
KRUSKAL ORDINAMENTO **CC:** $O(m \log m + mn)$	1. **ORDINO PER COSTO MINIMO** $C = 1 \quad AB, AC, \ldots$ $C = 2 \quad CD, FG \ldots$ 2. **SCELGO UN ARCO DI COSTO MINIMO** 3. **SELEZIONARE ARCO SUCCESSIVO** **Connesso al primo** dato, in modo tale che **non formino cicli.** **Controllo che gli estremi non siano già inclusi** ***ARCO COSTO NODI_INCLUSI S/N***
PRIM MATRICIALE **CC:** $O(n^2)$	1. **MATRICE DI ADIACENZA** 2. **SCELGO UN ARCO ARBITRARIO** Cancelo la sua riga. $C = C + C_{arco} \quad T = T + i_{arco}$ 3. **SELEZIONARE ARCO SUCCESSIVO** Nella **colonna dell'indice cancellato. ATTENZIONE ISOCOSTO!** ***T ARCO COSTO***

Branch & Bound

PROCEDIMENTO	DESCRIZIONE
FATHOMING **CC:** $O(2^n)$	1. **INAMMISSIBILITA'** 2. **SOLUZIONE INTERA** $LB = UB$ Eventualmente **registro come migliore** 3. **SOLUZIONE NON MIGLIORANTE** **MIN** $UB_{corrente} \leq LB_i$ **SOL:** $LB_{padre} \leq UB_{sol} = LB_{sol} \leq LB_i$ **MAX** $UB_{corrente} \geq LB_i$ **SOL:** $UB_i \leq LB_{sol} = UB_{sol} \leq UB_{Padre}$
KBP 01 **CC:** $O(n \log n)$	1. $\frac{p_i}{w_i}$ **DECRESCENTE** X_1 … X_i P_i … W_i … $\frac{P_i}{W_i}$ 2. **ELEMENTO CRITICO** $s = X_s$ s è il primo nell'ordine che supera la capienza dello zaino $s = \min_j \{j : \sum w_i > W\}$ 3. **DETERMINO LA SOGLIA DI RILASSAMENTO** Tenendo conto che W^* indica **lo spazio residuo** $x_i = \begin{cases} 1 & i<s \\ \frac{W^*}{w_i} & i=s \\ 0 & i>s \end{cases}$ $W^* = W - \sum_{i=1}^{s-1} w_i$ $LB = \sum_{i=0}^{i<x_s} p_i \quad UB = \sum_{i=0}^{j<x_s} p_i + R_{es} * p_s$
	STEP i $X_i + X_j + X_n$ $w_i + w_j + w_n = w_{tot} \leq W_{max}$ $x_{..} = 0 \quad x_{..} = 1$ **LB=**W_{tot} **UB=**$W_{tot} + k w_s$ $k = \frac{W_{max} - W_{tot}}{w_s}$ **ATTENZIONE: CAPACITA' SOTTOPROBLEMA**

Cammino minimo

ITERAZIONE	OPERAZIONE
ALGORITMO DI DIJKSTRA **CC:** $O(n^2+m)$	1. **INIZIALIZZO** $\overline{S} = \{2,3,4,...,n\}$ e $\Pi(1) = 0$ **SE** $i \subset \Gamma(1)$ => $\Pi(i) = l_{1i}$ **ALTRIMENTI** $\Pi(i) = \infty$ $P(i) = 1 \forall i \in \Gamma(1)$ 2. **SELEZIONO NODO** j: $\min_{i \in \overline{S}} \{\Pi(i)\}$ $\overline{S} = \overline{S} - j$ Aggiorno l'insieme $\overline{S}$ **SE** $\overline{S} = \varnothing$ **STOP** **ALTRIMENTI PASSO 3** 3. **SELEZIONO SUCCESSORE** $\forall i \in \quad \Gamma(j) \cap \overline{S} \quad \Pi(i) = min\{\Pi(i), \Pi(j) + lji\}$ **SE** $\Pi(i) = \Pi(j) + lji \qquad P(i) = j$; **PASSO 2**
	ITERAZIONE i $\overline{S} = \overline{S} - X \qquad J = X$ $\Gamma(J) = \{Y_1, Y_2, \dots\}$ $\Pi(Y_i) = min\{\Pi(Y_i), \quad \Pi(J) + l_{yi}\}$ **SE AGGIORNO** $P(Y_i) = J$ **RISCRIVO IL VETTORE** Π

Complessità computazionale

BELLMAN-FORD **CC:** $O(nm)$	1. **INIZIALIZZO** $\Pi(0) = i \quad O(n)$ 2. **TUTTI I PREDECESSORI (E ARCHI) -** m $O(m)$ 3. **ITERNO** n **VOLTE** $O(nm)$
PRODOTTO MATRICIALE **CC:** $O(mnp)$	1. **PRODOTTO RIGA-COLONNA** $O(n)$ $(n-1)$ somme e n moltiplicazioni 2. **ITERO** mp **VOLTE** $O(mnp)$
METODO N-O **CC:** $O(n+m)$	1. **CONFRONTI COSTANTI** $O(n+m-1)$ Grandezza dello spazio delle soluzioni
ALGORITMO DI DIJKSTRA **CC:** $O(n^2+m)$	1. **INIZIALIZZO** S $O(n)$ 2. **VALORE MINIMO** Π **-** n **VOLTE** $O(n) \cdot n$ 3. **CONSIDERO TUTTI GLI** m **ARCHI** $O(m)$
VARI	1. **ORDINAMENTO VETTORE NUMERI** $O(n^2) -> O(nlogn)$ 2. $B^{-1} = O(m^3) \quad B^{-1}b = O(m^2) \quad x_B \geq 0 = O(m) \quad r_D \geq 0 = O(mn)$

PL

VAR	***READ - THINK - DEFINE - FORMULATE*** **DISCRETE** $x_i \in Z$ **REALI** $x_i \in R$ **BINARIE** $x_i \in \{0,1\}$ $max(min\{e_1, .., e_n\}) = maxY \qquad y \leq e_{1...n}$ $min(max\{e_1, .., e_n\}) = minY \qquad y \geq e_{1..n}$ $min\|e\| = minY \qquad y \geq -e \quad y \geq e$ **IF- THEN ELSE** $\qquad x_i \leq My_i \quad x_j \leq My_j \quad y_i + y_j \leq 1$

Note finali

Alcuni dei contenuti presenti nelle seguenti dispense sono stati liberamente tratti dai materiali didattici disponibili al Politecnico di Torino.
Le dispense sono state elaborate dal sottoscritto come complemento allo studio e non intendono in alcun modo sostituire la completezza dei libri di testo e delle lezioni dalle quali sono state liberamente tratte.
Le dispense sono state scritte per l'esame di Ricerca Operativa dell'A.A. 2016-2017, docente Federico Della Croce, corso di laurea in Ingegneria Gestionale L8.
E' doveroso quindi citare alcuni delle fonti da cui sono stati liberamente tratti alcune parti di esercizi e/o metodologie di soluzione:

- Federico Della Croce, Rosario Scattamacchia, Whiteboard e appunti del corso di Ricerca Operativa, A.A. 2016-2017.
- wikipedia.org
- Maria Francesca Norese, Introduzione ai metodi multicriteri di surclassamento