

Pianificazione Automatica e sistemi di Supporto delle Decisioni

Matteo Aprile

Professore: Giampaolo Ghiani, Enamuele Manni

INDICE

I Definizioni

I-A	Business Analytics	2
I-B	Decisioni	2
I-C	Business Intelligence (BI)	2
I-D	Data visualization	2
I-E	Decision Support Systems (DSS)	2
I-F	Operations Research (OR)	2
I-G	Agents	3
I-H	Artificial Intelligence (AI)	3
I-I	Machine Learning (ML)	3
I-J	Deep Learning	3
I-K	Data Mining (DM)	3

II Software solutions and languages for AP and DSS

II-A	Decisioni operative/strutturate	4
II-B	Decisioni non strutturate o destrutturate	4
II-C	Decisioni semistrutturate	4

III Introduzione all'ottimizzazione matematica

III-A	Introduzione	5
III-B	Ingredienti principali	5
III-C	Descrizione del problema	5
III-D	Dati del problema	5
III-E	Descrizione del problema con un modello matematico	5
III-F	Risolvere il modello matematico	6
III-G	Terminologia	6
III-H	Implementazione in Python	6
III-I	Esercitazione	6

IV Formulazioni equivalenti di un problema di programmazione lineare

IV-A	Problema in FORMA GENERALE	8
IV-B	Problema in FORMA CANONICA	8
IV-C	Problema in FORMA STANDARD	8
IV-D	Terminologia	8
IV-E	Trasformazioni per ricondursi alla forma standard	8
IV-F	Esempio 1	9
IV-G	Esempio 2	9

V Optimization models review

V-A	Scheduling	10
V-B	Project scheduling	10
V-C	Esempio	11
V-D	Velocizzazione del progetto	11
V-E	Esempio velocizzazione progetto	11
V-F	Low sizing models	11
V-G	Scrivere il modello di ottimizzazione	12
V-H	Algoritmo del simplesso	13
V-I	Tableau	13
V-J	Esempio Tableau	14
V-K	Forma canonica	14
V-L	Generalizzazione forma canonica	15
V-M	Esercizio forma canonica	17
V-N	Soluzioni ottime multiple	18
V-O	Esempio soluzioni ottime multiple	18
V-P	Algoritmo della convergenza	18
V-Q	Esempio BFS degenerare	19
V-R	Regola dell'anticycling: Bland rule	20
V-S	Variabili artificiali	20
V-T	Metodo del simplesso in 2 fasi	20
V-U	Costruzione del problema artificiale	20
V-V	Esempio problema artificiale	20
V-W	Esercitazione variabili artificiali	21

VI Branch-and-Bound Technique for Solving Integer Programs

VI-A	Principi dell'algoritmo Branch-and-Bound	22
VI-B	Approccio Branch-and-Bound	22
VI-C	Branching	23
VI-D	Fathoming	25
VI-E	Esercizio Branch-and-Bound	25

VII Constructive heuristics

VII-A	Introduzione	26
VII-B	Travelling Salesman Problem (TSP)	26
VII-C	Algoritmo Greedy	26
VII-D	Miller-Tucker-Zemlin	27
VII-E	Algoritmo relax-and-fix	27
VII-F	Algoritmo rolling horizon	28
VII-G	Multiple criteria decision making	28
VII-H	Esempio problema scheduling	28
VII-I	Superior solutions	28
VII-J	Metodo a priori	29
VII-K	Metodo a posteriori	29
VII-L	Metodo interattivo	29

I. DEFINIZIONI

Ci occuperemo di 2 tipi di scenari:

- usare **algoritmi a supporto delle decisioni**
- usare algoritmi che **sostituiscono completamente l'uomo**

A. Business Analytics

Disciplina che utilizza dati, statistiche, modelli matematici per **aiutare a prendere delle decisioni in base a dei dati**.

Possiamo racchiudere i suoi **passaggi** in:

- 1) **descriptive analytics**: capire **cosa sia successo nel passato** tramite i dati disponibili
- 2) **predictive analytics**: cercare di **fare delle previsioni** in base ai dati già disponibili
- 3) **prescriptive analytics**: **creare un piano di azione** per poter massimizzare il KPI (Key Performance Indicator)



Figura 1. Fasi della business analytics

B. Decisioni

Rappresenta la **scelta di un elemento tra più soluzioni** dopo aver ponderato le opzioni.

Possiamo avere più casi d'uso:

- **simplest case**: abbiamo **poche alternative** quindi una semplice scelta
- **multiple criteria**: abbiamo **più metri di paragone** delle performance, quindi si dovranno tenere in conto:
 - **soluzioni migliori** di altre (dette di Pareto)
 - **vincoli** dovuti dai clienti o da casi logistici da gestire (es: spedizioni)
 - ottimizzazioni matematiche
 - **conflitti tra i vincoli**
- **incertezze e rischi**:
 - **decisioni operative**: di **breve periodo reversibili e limitate** a "n" persone del team
 - **decisioni tattiche**: **coinvolge una parte dell'organizzazione** per un medio periodo
 - **decisioni strategiche**: di **lungo periodo non reversibili e coinvolgono denaro**

- **decisioni strutturate**: hanno una **procedura di risoluzione specifica**
- **decisioni non strutturate**: richiedono **creatività ed esperienza** in un dato settore

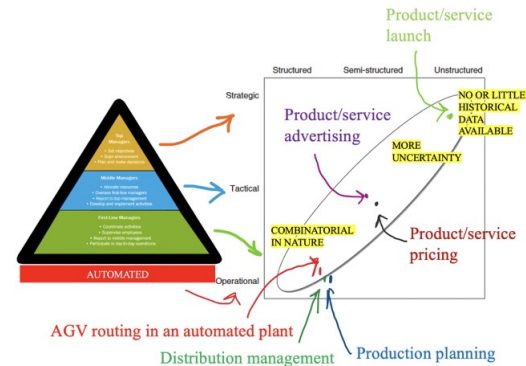


Figura 2. Diagonale decisionale

C. Business Intelligence (BI)

Usato per indicare un **sistema dedicato alla raccolta di dati e alla loro elaborazione** al fine di un reporting, infatti per "Intelligence" si intende investigazione. Venivano **usati su dati atomici** per avere delle conoscenze approfondite in un determinato business.

D. Data visualization

Consiste nel **prendere dati e plottare un grafico**, ma in realtà ora si ha una trattazione più metodologica, cioè se visualizzare in modo statico o meno i dati.

E. Decision Support Systems (DSS)

Si indicava un **sistema computerizzato dotato di un sistema di "data management"** per creare un modello di ottimizzazione, fornendo un feedback tramite un'interfaccia. Ora indica una varietà di sistemi per visualizzare i dati in larga misura o meno.

F. Operations Research (OR)

Attività organizzative per portare avanti un sistema logistico. Per "research" si indica la ricerca delle operation per conseguire dei risultati, avremo come sottocategorie:

- **ottimizzazione matematica**
- **queueing theory**: studio matematico delle linee in attesa il limite è che funzionano solo con sistemi semplici e con richieste di servizio in ordine stocastico
- **simulazione**: per usarle è **necessario generare dei numeri randomici quindi inconveniente** (bisogna fare un'analisi statistica dei risultati dalle quali si farà una stima)
- **game theory**: decisioni con **più players**

G. Agents

È un **sistema che si muove in un environment** (ambiente), ha dei **sensori** tramite i quali percepisce alcuni aspetti del mondo che lo circonda quindi si crea una **rappresentazione del mondo circostante** che può vedere. È capace di **influenzare l'ambiente tramite degli attuatori** come ruote o braccia (intendiamo anche agenti software).

Possiamo classificarli come:

- **agenti autonomi**: se è concepito in modo tale che **tramite un'istruzione sintetica raggiunge un goal sviluppando le azioni per raggiungerlo** In realtà può anche non essere una sequenza di azioni dato che **potrebbero esserci degli imprevisti**
- **agenti intelligenti**: se
- **impara dall'esperienza**
- crea una **rappresentazione dell'ambiente** che lo circonda e **ci ragiona sopra** per un possibile risultato delle proprie azioni
- **si adatta ad un ambiente mutevole**

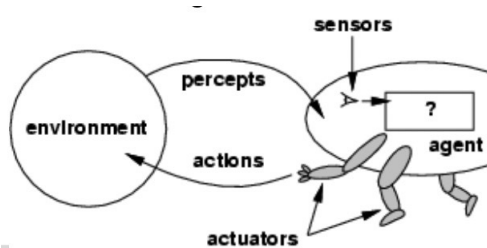


Figura 3. Schematizzazione di un agente e sue caratteristiche

H. Artificial Intelligence (AI)

Comprende tante sottodiscipline:

- **automated reasoning**: legato alla **rappresentazione del mondo e come ragionare su di essa** ma anche calcolandone le probabilità
- **automated planning**: usato in ambienti industriali
- **automated learning**
- **natural language processing**: **sviluppare agenti software** per fare sintesi di testi, scrivere automaticamente articoli, chat bot, ecc
- **perception**: visione artificiale
- **manipulation**: avere un **agente che può modificare l'agente circostante**

I. Machine Learning (ML)

Consiste nell'**apprendimento automatico** e quindi lo sviluppo degli **agenti che apprendo tramite la loro esperienza pregressa**. Ci sarà allora una fase di **training**. Una delle possibili **architetture che permette di farlo sono le Neural Networks** prima avevano solo 2/3 neuroni, ora ne hanno vari strati il che fornisce delle prestazioni impressionanti

J. Deep Learning

Si basa sull'**apprendimento automatico** con reti neurali tramite un gran numero di strati di neuroni.

K. Data Mining (DM)

Usare metodi di Machine Learning per **estrarre manualmente dei pattern dai dati**, cioè una **regolarità o un trend**. È quindi la parte nobile del knowledge discovery in db, dato che i dati sono in genere disponibili su db o da altre piattaforme.

La **sequenza** nella quale interviene è:

- 1) **prendere** i dati
- 2) trovare i vari **target**
- 3) **preprocessare** i dati
- 4) trasformare i dati tramite il **data mining**
- 5) trovare dei **patterns** (dopo il data mining)

II. SOFTWARE SOLUTIONS AND LANGUAGES FOR AP AND DSS

A. Decisioni operative/strutturate

Sono una classe importante, **si possono prendere tramite una procedura standard** che può seguire un manuale o delle normative, automatizzata o no. Queste decisioni di breve periodo si collocano in basso a destra in figura 2.

Non essendo decisioni dove possiamo solo supportare, allora si possono andare a **codificare in un linguaggio di programmazione procedurale** come C++, Java, ecc...

Potremo avere un **approccio**:

- **procedurale**: dove devo far **generare delle azioni** in seguito di un obiettivo
- **dichiarativo**: si divide in:
 - 1) **modellazione** del problema
 - 2) **descrivono tramite un linguaggio di modellazione** (modelling language) che è un linguaggio di programmazione matematico come AMPL, oppure in linguaggi come python con Amply e Pulp
 - 3) **solver of the shelf**, che ci darà delle istruzioni per il nostro contesto

Si usano degli **spreadsheet** che però non riescono a gestire big data e tendono a generare errori.

Il linguaggio più usato è **Python** ma non è la soluzione più efficiente per tutte quelle applicazioni dove il tempo di calcolo è importante.

C. Decisioni semistruzzurate

Vogliamo solo **valutare le prestazioni di un sistema**. Un esempio sono i sistemi che **presentano un comportamento random** per motivi:

- 1) i **server hanno un tempo di risposta** che possiamo modellare
- 2) le richieste del sistema **arrivano in maniera stocastica**

Si usano, in questo caso, **metodi simulativi** tramite dei Visual Interactive Modelling System, **per simulare la rete** per la quale passano le informazioni e i server ognuno con diverse proprietà di ciascun linker.

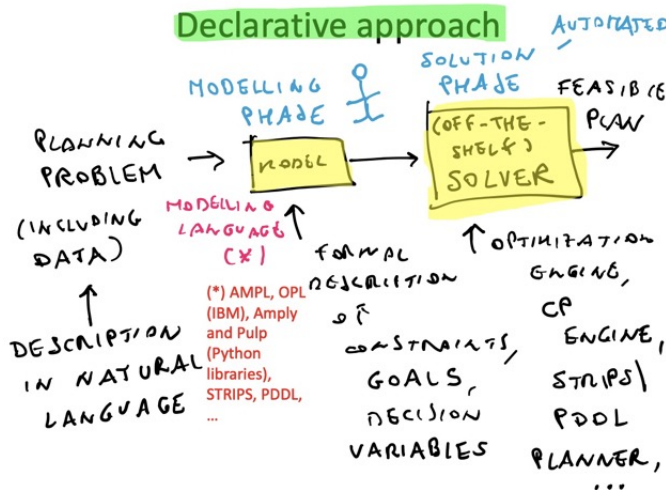


Figura 4. procedura implementata

Il che è utile dato che **per agire su un problema basterà cambiare il modello** senza cambiare solver, dovrò solo cambiare il modello. È la soluzione più **economico e flessibile** ma è **meno performante** se in ambienti realtime devo prendere soluzioni in tempi molto stretti. Quindi in questi casi servono approcci procedurali.

Per sistemi che devono prendere soluzioni nel breve, si usa C, C++, C#.

B. Decisioni non strutturate o destrutturate

In questo caso **non possiamo automatizzare**, quindi:

- 1) tiro fuori i **dati aggregati**
- 2) si creano statistiche con **modelli di ottimizzazione**

III. INTRODUZIONE ALL'OTTIMIZZAZIONE MATEMATICA

A. Introduzione

Partiamo da un **insieme di formule ed equazioni che model-
leranno il problema**. Con questo modello proviamo a trovare
una **soluzione** al nostro problema **attraverso algoritmi o ri-
solutori**. L'output è una soluzione per il nostro modello da
implementare nel mondo reale.

B. Ingredienti principali

Gli ingredienti principali sanno:

- **dati** del problema
- variabili: dette anche var decisionali: scelte da fare in merito al problema. rappresentano quindi le scelte, quello su cui il decisore può intervenire
- vincoli: equazioni che definiscono i valori che le variabili possono assumere
- funzione obiettivo: sarà una formula che rappresenta una misura di tipo quantitativo per capire quando è buona la soluzione che abbiamo ottenuto. quindi dovremo ottimizzare questo valore in base al contesto

Parleremo di **programmazione lineare con modelli matema-
tici** o relazioni lineari, dato che **molti problemi reali si rifanno
a modelli lineari**, per quanto essi possano essere complessi.

C. Descrizione del problema

Proviamo a risolvere un problema di mix di produzione, cioè un sistema con un impianto con 2 stabilimenti in cui:

- 1) nel primo: diamo le materie prime e vengono realizzati i componenti in uscita
- 2) nel secondo: diamo i componenti realizzati che vengono assemblati per creare il prodotto finito



Figura 5. Catena tra i due stabilimenti

Supponendo di voler realizzare 2 prodotti A, B con un differente profitto. Determinare il **mix di produzione**, cioè quante unità di A e B produrre la prossima settimana. Saranno presenti dei **vincoli** creati dalle risorse come i macchinari o gli addetti che potranno lavorare un numero di ore finito.

D. Dati del problema

- ore di lavoro:

Stab	A	B	Addetti
1	4 ore	2 ore	10
2	2 ore	4 ore	10

Tabella I

TABELLA DELLE ORE DI LAVORO

- ogni addetto lavora 40 ore/settimana
- profitto €/pallet:
- richiesta del prodotto nella prossima settimana:

A	B
15k	10k

Tabella II

TABELLA DEL PROFITTO €/PALLET

A	B
40	120

Tabella III

TABELLA DEL PROFITTO EURO/PALLET

E. Descrizione del problema con un modello matematico

Per **effettuare una modellazione** faremo:

- 1) **identificare le variabili decisionali**:
 - x_A : # di pallet di prodotto A da realizzare
 - x_B : # di pallet di prodotto B da realizzare
- 2) **definire la funzione obbiettivo (FO)**, per massimizzare il profitto
- 3) **definire i vincoli espressi come uguaglianza o disuguaglianza**
 - vincolo 1: capacità produttiva dello stab 1 $4x_A + 2x_B$ che non può superare 40×10 cioè ore disponibili ogni settimana per un addetto * numero di addetti:

$$4x_A + 2x_B \leq 400$$

- vincolo 2: capacità produttiva dello stabilimento 2 $2x_A + 4x_B$ che non può superare 40×10 cioè ore disponibili ogni settimana per un addetto * numero di addetti:

$$2x_A + 4x_B \leq 400$$

- vincolo 3: vincolo sulla richiesta di A:

$$x_A \leq 40$$

- vincolo 4: vincolo sulla richiesta di B:

$$x_B \leq 120$$

- vincolo 5: vincolo di non-negatività:

$$x_A, x_B \geq 0$$

Nella forma completa il **modello complessivo** è:

$$MAX = z = 15x_A + 10x_B$$

sottoposto ai vincoli (sv):

- $4x_A + 2x_B \leq 400$
- $2x_A + 4x_B \leq 400$
- $x_A \leq 40$
- $x_B \leq 120$
- $x_A, x_B \geq 0$

F. Risolvere il modello matematico

Rappresentiamo sul piano cartesiano tutte le soluzioni ammissibili cercando quella che massimizza il nostro risultato

Impostiamo delle rette per ogni vincolo:

- presa $4x_A + 2x_B \leq 400$ poniamo $= 0$, a turno, x_A e x_B : (200, 100)
- presa $2x_A + 4x_B \leq 400$ poniamo $= 0$, a turno, x_A e x_B : (100, 200)
- presa $x_A \leq 40$: (40, 0)
- presa $x_B \leq 120$: (0, 120)

Avremo allora una **regione ammissibile** dove valgono tutti i vincoli e nella quale dovrebbe essere presente la nostra soluzione ammissibile. Per trovare il punto che rende massima la funzione z usiamo il **metodo del gradiente**:

$$\nabla z = \begin{bmatrix} \frac{dz}{dx_A} \\ \frac{dz}{dx_B} \end{bmatrix} = \begin{bmatrix} 15 \\ 10 \end{bmatrix}$$

dove ∇z sarà la massima crescita che viene rappresentata tramite (15, 10).

Tracciando una **retta perpendicolare (curve di livello)** alla retta del gradiente avremo valori sempre buoni ma più bassi di quelli sul gradiente, a patto che siano validi. Troveremo in fine il punto massimo che consente di massimizzare, cioè il più estremo alla regione ammissibile sarà il nostro punto.

Seguendo la retta del gradiente troviamo che la **soluzione ottimale** si trova nell'intersezione tra le rette del vincolo 2 con il 3: $x_A = 40$ $2 \cdot 40 + 4x_B = 400$ quindi $x_B = 80$.

La soluzione ottimale sarà:

$$\begin{cases} x_A = 40 \\ 2x_A + 4x_B = 400 \end{cases} \quad \begin{cases} x_A = 40 \\ x_B = 80 \end{cases}$$

Si nota che lo stabilimento 2 viene saturato e quello 1 no, dal fatto che la soluzione giace sulla retta del vincolo per il quale si satura.

G. Terminologia

Possiamo avere altre forme di modelli di PL:

- fo da minimizzare
- vincoli di uguaglianza
- vincoli \geq
- variabili negative
- variabili non vincolate

Terminologie da sapere:

- **soluzione**: quella di output
- **soluzione ammissibile**: soluzione, se esiste, che **soddisfa tutti i vincoli**
- **soluzione inammissibile**: se **viola almeno un vincolo**
- **regione ammissibile**: tutti i punti che rispettano i vincoli
- **prob inammissibile**: **regione ammissibile vuota**
- **prob ammissibile**:
 - soluzione ottima singola
 - soluzioni multiple
 - fo illimitata

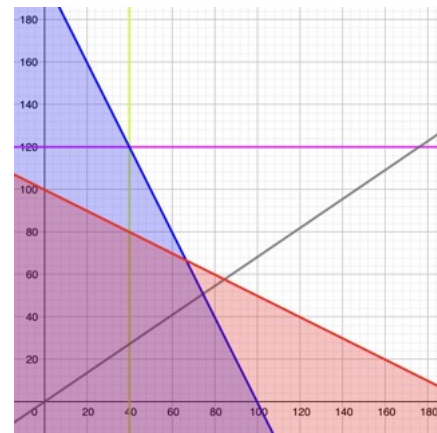


Figura 6. Rappresentazione grafica esempio

H. Implementazione in Python

```
1 import pulp as p
2
3 # 1. creazione del modello
4 model = p.LpProblem("ProductMix", p.LpMaximize)
5
6 # 2. definisco le variabili decisionali
7 x_A = p.LpVariable("x_A", cat="Continuous", lowBound=0)
8 x_B = p.LpVariable("x_B", cat="LpContinuous", lowBound=0)
9
10 # 3. definisco la funzione obiettivo in funzione delle variabili decisionali
11 model += 15 * x_A + 10 * x_B
12
13 # 4. definire i vincoli
14 model += 4 * x_A + 2 * x_B \leq 400
15 model += 2 * x_A + 4 * x_B \leq 400
16 model += x_A \leq 40
17 model += x_B \leq 120
18
19 # 5. risolvere il problema
20 model.solve()
21
22 # print della soluzione
23 print("next week produce {} pallets of A".format(x_A.varValue))
24 print("next week produce {} pallets of B".format(x_B.varValue))
```

I. Esercitazione

1) Massimizzare la f.o. $z = 8x_1 + 6x_2$, con i vincoli:

- $x_1 \leq 5$
- $x_2 \leq 7$
- $4x_1 + 3x_2 \leq 29$
- $x_1, x_2 \geq 0$

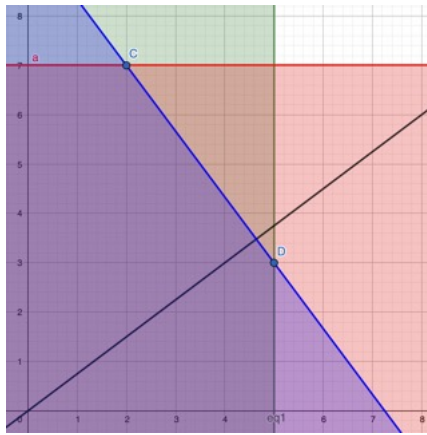


Figura 7. Rappresentazione grafica esempio 1

$$\nabla z = \begin{bmatrix} \frac{dz}{dx_A} \\ \frac{dz}{dx_B} \end{bmatrix} = \begin{bmatrix} 8 \\ 6 \end{bmatrix}$$

Abbiamo che esiste una curva di livello coincidente con lo spigolo CD, quindi abbiamo delle soluzioni ottime multiple:

- vertice C, prendiamo allora vincolo 2 e 3:

$$x_2 = 7 \rightarrow x_1 = 2$$

- vertice D, prendiamo allora vincolo 1 e 3:

$$x_1 = 5 \rightarrow x_2 = 3$$

- punti del segmento CD

Quindi $z = 58$

2) Minimizziamo la f.o. $z = 25x_1 + 22x_2$, con i vincoli:

- $x_1 + x_2 \geq 5$
- $3x_1 + 2x_2 \geq 12$
- $3x_1 + 6x_2 \geq 18$
- $x_1, x_2 \geq 0$

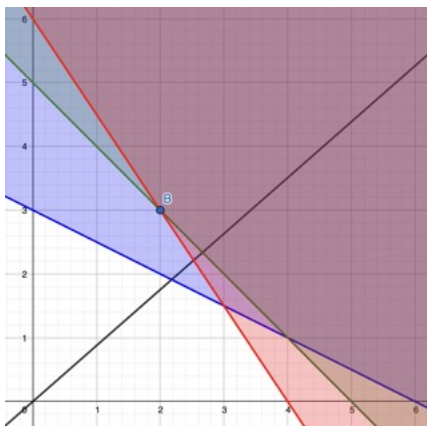


Figura 8. Rappresentazione grafica esempio 2

$$\nabla z = \begin{bmatrix} \frac{dz}{dx_A} \\ \frac{dz}{dx_B} \end{bmatrix} = \begin{bmatrix} 25 \\ 22 \end{bmatrix}$$

Per poter minimizzare, tracciando la curva di livello, trovando che la soluzione ottima si troverà dal punto B dato dall'intersezione dei vincoli 1 e 2:

$$x_1 = 2, x_2 = 3$$

Quindi $z = 116$

3) Massimizziamo la f.o. $z = 2x_1 + x_2$, con i vincoli:

- $x_1 - x_2 \leq 1$
- $2x_1 + x_2 \geq 6$
- $x_2 \geq 6$
- $x_1, x_2 \geq 0$

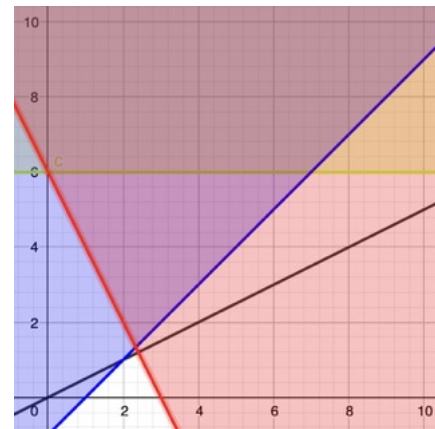


Figura 9. Rappresentazione grafica esempio 3

$$\nabla z = \begin{bmatrix} \frac{dz}{dx_A} \\ \frac{dz}{dx_B} \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Non raggiungeremo la regione ammissibile, quindi il problema non ammette una soluzione ottima.

4) Minimizziamo la f.o. $z = -2x_1 + 3x_2$, con i vincoli:

- $x_1 - 2x_2 \geq -2$
- $2x_1 - x_2 \leq 3$
- $x_2 \geq 4$
- $x_1, x_2 \geq 0$

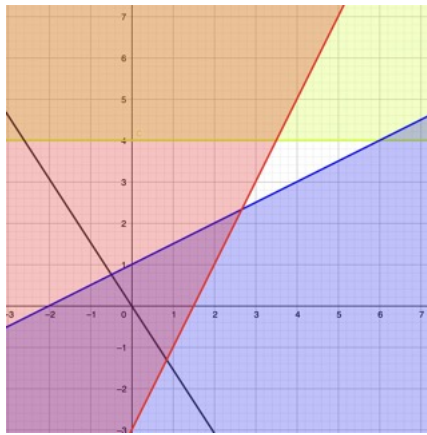


Figura 10. Rappresentazione grafica esempio 4

$$\nabla z = \begin{bmatrix} \frac{dz}{dx_A} \\ \frac{dz}{dx_B} \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$$

La regione ammissibile e' vuota e per tanto il problema e' inammissibile, quindi non esiste un punto che soddisfa contemporaneamente tutti i vincoli.

- 5) L'azienda vuole decidere oltre al piano di produzione anche la giusta riallocazione degli addetti (10-10) tra i due reparti. Le variabili decisionali sono:

- x_A, x_B : i prodotti
- n_p : # addetti allocati al reparto produzione
- n_a : #addetti allocati al reparto assemblaggio

Quindi andremo ad aggiungere il vincolo per cui $n_p + n_a = 20$.

Perciò avremo: $z = 15x_A + 10x_B$, con vincoli:

- $4x_A + 2x_B \leq 40n_p$
- $2x_A + 4x_B \leq 40n_a$
- $x_A \leq 40$
- $x_B \leq 120$
- $n_a + n_p = 20$
- $x_A, x_B, n_a, n_p \geq 0$

IV. FORMULAZIONI EQUIVALENTI DI UN PROBLEMA DI PROGRAMMAZIONE LINEARE

A. Problema in FORMA GENERALE

In generale nella **zona ammissibile** diciamo:

$$X = \{x \in R^n : A_x \geq b, D_x = l, x_j \geq 0\} \quad (*)$$

$$\forall j \in J \subseteq \{1, 2, \dots, n\}$$

dove **definiamo la possibilità di vincoli di \geq , $=$ e variabili ≥ 0 .**

La funzione obiettivo è definita da:

$$z = c_x \quad \text{t.c.} \quad z = \min z, x \in X$$

che rappresenta il **problema espresso in forma generale**.

B. Problema in FORMA CANONICA

Se in (*) abbiamo:

- $D = 0$
- $J = \{1, 2, \dots, n\}$

allora il **problema si dice in forma canonica**:

$$\min_x \{z = c_x : A_x \geq b, x \geq 0\}$$

C. Problema in FORMA STANDARD

Se in (*) abbiamo:

- $A = 0$
- $J = 1, 2, \dots, n$

allora il problema è:

$$\min_x \{z = c_x : D_x = l, x \geq 0\}$$

Possiamo sempre **ricorrere tramite trasformazioni alla forma standard**.

D. Terminologia

Nella forma standard abbiamo che:

- **z**: la **funzione obiettivo** per la quale trovare il valore minimo
- **A**: **matrice dei vincoli**
- **D**: **matrice dei coefficienti**, matrice di dimensione $m \times n$
- **l**: **vettore dei termini noti** vettore colonna
- **c**: **detto vettore dei coefficienti di costo**, vettore di riga

E. Trasformazioni per ricondursi alla forma standard

- 1) **variabili non vincolate di segno**:

$$x_j \text{ t.c. } j \notin J$$

per trasformarla possiamo **sostituire a x_j la somma algebrica di 2 variabili non negative**:

$$x_j = x_j^+ - x_j^-$$

$$\forall x_j^+ \geq 0, x_j^- \geq 0$$

Se abbiamo k ($\leq n$) variabili non vincolate in segno, possiamo evitare di introdurre k coppie di variabili non negative. È possibile considerare una variabile $x_0 \geq 0$ e sostituire la generica variabile non vincolata di segno con $x_j = x_j^+ - x_0$. Così introduciamo "solo" $k + 1$ variabili.

- 2) vincoli non espressi in forma di uguaglianza (\leq)

$$\sum_{j=i}^n a_{ij}x_j \leq b_i$$

presa la variabile di Slack: $S_i \geq 0$, avremo:

$$\sum_{j=i}^n a_{ij}x_j + S_i = b_i$$

questa variabile misura lo Slack che esiste per far sì che il vincolo sia rispettato per uguaglianza o no (vincolo > 0)

- 3) vincoli non espressi in forma di uguaglianza (\geq)

$$\sum_{j=i}^n a_{ij}x_j \geq b_i$$

presa una variabile di Surplus: $S_i \geq 0$, avremo:

$$\sum_{j=i}^n a_{ij}x_j - S_i = b_i$$

- 4) trasformazione di vincoli da uguaglianza in disuguaglianza sostituendo:

$$\sum_{j=i}^n a_{ij}x_j = b_i$$

sostituendolo a:

$$\sum_{j=i}^n a_{ij}x_j \geq b_i$$

$$\sum_{j=i}^n a_{ij}x_j \leq b_i$$

- 5) funzione obiettivo:

Se la f.o. è $\max z = c_x$, si può trasformare:

$$\max z = -\min -z$$

F. Esempio 1

- 1) DATI

f.o.: $\min z = x_1 + 2x_2$

vincoli:

- $6x_1 + 4x_2 \leq 24$
- $4x_1 + 8x_2 \leq 32$
- $x_2 \geq 3$
- $x_1, x_2 \geq 0$

- 2) TRASFORMAZIONE IN FORMA STANDARD

Per il primo vincolo del tipo \leq , aggiungiamo una variabile non negativa (slack):

$$6x_1 + 4x_2 + x_3 = 24$$

per il secondo vincolo operiamo in maniera analoga:

$$4x_1 + 8x_2 + x_4 = 32$$

per il terzo vincolo del tipo \geq , aggiungiamo una variabile ausiliaria negativa:

$$x_2 - x_5 = 3$$

vincolo sulle variabili:

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

quindi nella formulazione standard avremo $j = 1, 2, 3, 4, 5$ quando in precedenza avevamo: $j = 1, 2$

G. Esempio 2

- 1) DATI

f.o.: $\max z = z_1 + z_2$

vincoli:

- $8x_1 + 6x_2 \geq 48$
- $5x_1 + 10x_2 \geq 50$
- $13x_1 + 10x_2 \leq 130$
- $x_1 \geq 0$

- 2) TRASFORMAZIONE IN FORMA STANDARD

Dato che non abbiamo vincoli su x_2 poniamo:

$$x_2 = x_2^+ - x_2^-$$

la f.o. sarà:

$$\max z = -\min -z = -x_1 - x_2 = -x_1 - x_2^+ + x_2^-$$

invece i vincoli:

- $8x_1 + 6x_2^+ - 6x_2^- - x_3 = 48$
- $5x_1 + 10x_2^+ - 10x_2^- - x_4 = 50$
- $13x_1 + 10x_2^+ - 10x_2^- + x_5 = 130$
- $x_1, x_2^+, x_2^-, x_3, x_4, x_5 \geq 0$

V. OPTIMIZATION MODELS REVIEW

A. Scheduling

Nell'ambito dei problemi dello scheduling abbiamo degli elementi ben specificati:

- **task/job** già assegnati
- **n macchine/processori**
- potremmo attrezzare le macchine con dei **tools**

Intendiamo **allocare i tasks alla macchine in "overtime"** quindi capire anche la **fascia temporale nella quale eseguire il task**. Potrebbe esserci un unico tempo di esecuzione oppure un task può avere dei tempi di esecuzione differenti su macchine differenti.

L'output sarà un diagramma di Ganth.

I task possono avere degli istanti di rilascio dove non potrebbe essere rilasciato dopo un certo istante di tempo (**ready time**).

Possono esserci delle **relazioni di precedenza tra i tasks**. Quindi non posso effettuare un task se prima non ho concluso l'altro.

Il diagramma mi dice nel tempo a che macchina è associato quale task ed in quali intervalli di tempo e con quale tool.

B. Project scheduling

Per progetto intendiamo un **insieme di tasks che sono realizzati al fine di raggiungere un goal**. La caratteristica di un progetto è che nel complesso le attività non sono mai state eseguite in precedenza.

Le **caratteristiche di un progetto** sono:

- **durata delle attività** che nota
- ha **a capo un Project Manager**: responsabile del progetto e dei tempi di realizzazione, costi di produzione, ecc...

Un progetto è **rappresentato da diverse attività** in una tabella fornita dal Project Manager:

Attività	Durata stimata d_i	Predecessori
1	10	-
2	10	-
3	10	1
4	10	1, 2

Tabella IV

TABELLA DI ORE DI LAVORO E PREDECESSIONI DELLE ATTIVITÀ

e in un diagramma aciclico (Activity On Node (AoN)):



Figura 11. Diagramma Activity On Node

dove abbiamo degli "archi" che rappresentano le predecesioni. Avremo anche dei **vertici fittizi**:

- **start**: lo colleghiamo tutte le attività che non hanno predecessori
- **end**: ci colleghiamo tutte le attività finali

Una funzione fondamentale del Project Manager è la possibilità di accelerare alcune attività agendo su:

- **Variabili decisionali**:

nel nostro caso, è lo **start time s_i** . Ipotizziamo che il progetto inizi al tempo $t = 0$, quindi per ogni task abbiamo che:

$$s_i \geq 0 \quad \forall \quad i \in TASKS$$

In più possiamo definire **$T \geq 0$ tempo di completamento del progetto (completion time)**.

Minimizziamo il completion time:

$$\min z = T$$

con $z = 1T + 0s_1 + 0s_2 + \dots + 0s_n$

- **relazioni di precedenza**:

relazioni che portano alcuni nodi a dipendere da altri:

$$p_{ij} = \begin{cases} 1 & \Leftrightarrow i \in j \\ 0 & \text{altrimenti} \end{cases}$$

con **p_{ij} matrice** costate e binaria:

$$p = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **vincoli di precedenza**:

Sia **T maggiorante del tempo di completamento delle task**:

$$s_i + d_i \leq T$$

allora:

$$p_{ij}(s_i + d_i) \leq s_j \quad \forall \quad i, j \in TASKS$$

Possiamo avere che:

- **$p_{ij} = 1$** : allora **i è predecessore di j** quindi il tempo di inizio del task j deve essere successivo o uguale al task i cioè $s_i + d_i$
- **$p_{ij} = 0$** : i non è predecessore quindi avremo $0 \leq s_j$ allora il vincolo è ridondante

scriviamo allora:

$$s_i + d_i \leq s_j \quad \forall \quad i, j \in TASKS, \quad p_{ij} \geq 0$$

C. Esempio

Un modello espanso per problemi di istanza:

1) funzione obiettivo: $\min z = T$

2) vincoli:

- $s_1 + 10 \leq T$
- $s_2 + 10 \leq T$
- $s_3 + 10 \leq T$
- $s_4 + 10 \leq T$
- $s_1 + 10 \leq s_3 (p_{13} = 1)$
- $s_1 + 10 \leq s_4 (p_{14} = 1)$
- $s_2 + 10 \leq s_4 (p_{24} = 1)$
- $s_1, s_2, s_3, s_4 \geq 0$
- $T \geq 0$

D. Velocizzazione del progetto

Il Project Manager ha un **budget** per poter velocizzare il progetto.

Se considero un **task i con durata non costante (d_i^N)**, avremo un valore nominale che dipende da un budget extra.

Il più semplice è l'**andamento lineare** dove all'aumentare delle risorse la durata si riduce in modo lineare. Il che è vero finché non si incontra un **vincolo inferiore d_i^m** .

Le **3 risorse** alle quali si possono far riferimento sono le **3M**:

- Man
- Machine
- Money

quindi $d_i = d_i^N$.



Figura 12. Diagramma budget

con:

- **pendenza w** : riduzione della durata del task i per unità di extra risorse (mesi di lavoro / k euro)
- $d_i = d_i^N - w_i x_i \geq d_i^m$ vincolo del valore minimo per task
- x_i : denaro usato per il task i
- B : budget totale

E. Esempio velocizzazione progetto

Avremo un modello con:

1) funzione obiettivo: $\min z = T$

2) vincoli:

- $s_i + d_i^N - w_i x_i \leq T \quad \forall i \in TASKS$
- $s_i + d_i^N - w_i x_i \leq s_j \quad \forall i, j \in TASKS, p_{ij} = 1$
- $d_i^N - w_i x_i \geq d_i^m \quad \forall i \in TASKS$
- $\sum_{i \in TASKS} x_i \leq B$
- $T \geq 0$
- $s_i \geq 0 \quad \forall i \in TASKS$
- $x_i \geq 0 \quad \forall i \in TASKS$

F. Low sizing models

Sono in genere usati da aziende manifatturiere.



Figura 13. Processo produttivo

Supponendo di avere un **tasso di domanda d** costante in base al tipo di prodotto. Ogni tipo di prodotto si differenzia dagli altri con una piccola modifica come può essere un differente gusto per una produzione di yogurt.

Questa differenziazione porta ad un **costo di setup** delle macchine che andranno pulite, generando un costo fisso k .

Il livello di scorte sarà rappresentato con dei picchi con ampiezza q :



Figura 14. Livello di inventario

avendo una domanda costante, avremo una diminuzione lineare nello scorte di magazzino.

Ovviamente avremo dei **costi medi di stockaggio h** dato che le scorte si "muoveranno" scambiandosi con altre scorte che entrano nel magazzino. Quindi andremo a calcolare il costo in base alla giacenza del **numero di scorte medie $\frac{q}{2}$** .

In base alla strategia avremo:

1) **caso estermo**:

gestione di tipo **just in time** dove **produco solo sotto commissione del cliente**.

Avremo quindi:

- **livello di scorte molto basso** con un livello medio delle scorte molto basso e dei **costi di stockaggio bassi**
- **maggioramento dei costi del setup**
- **pago k più volte durante l'anno**

- 2) **caso produzione annua:**
si produce un **quantitativo pari alla domanda annua**.
Avremo quindi:
- grandi **costi di stockaggio**
 - pago k solo una volta all'anno



Figura 15. Casi particolari

G. Scrivere il modello di ottimizzazione

Le fasi da seguire prevedono la scrittura di:

- 1) **variabili decisionali:** variabile matematica per descrivere la mia decisione
- 2) **funzione obiettivo:** costo totale annuo composto dal **costo di scorta e quello di setup**

Avremo allora:

$$z = k \frac{d}{q} + h \frac{q}{2}$$

Per la **soluzione ottima**, faccio il gradiente:

$$\frac{dz}{dq} = 0 \Leftrightarrow -k \frac{d}{q^2} + \frac{h}{2} = 0$$

Concludiamo che il **lotto economico**, per minimizzare i costi, sarà raggiunto da:

$$q^* = \sqrt{\frac{2kd}{h}}$$



Figura 16. Caso di lotto economico

Questo modello **nella pratica ha dei limiti** dati i possibili vincoli come:

- spazio limite di un magazzino
- produzione di più prodotti contemporaneamente
- vincolo di immobilizzo
- vincolo sul capitale
- ecc ...

Associati al vincolo di immobilizzo possiamo avere anche un **limite nei prodotti che posso fare di A e di B**. Se si ipotizza una domanda costante, ho che il **costo totale annuo z** sarà data dal costo annuo di A e di B:

$$z = (k_A \frac{d_A}{q_A} + h_A \frac{q_A}{2}) + (k_B \frac{d_B}{q_B} + h_B \frac{q_B}{2})$$

Avremo che z è dato da 2 termini uno che dipende da A ed uno da B. Per noi supponiamo che $k_A = k_B = k$.

Di conseguenza anche i lotti di approvvigionamento possono essere diversi. Quindi, quando è necessario, avremo che dovremo **decidere quante confezioni produrre di A e quante di B**.

Nel **caso peggiore ipotizzo che la produzione contemporanea di 2 lotti**, quindi non dovrà superare la capacità di magazzino Q :

$$q_A + q_B \leq Q \quad \forall \quad q_A, q_B \geq 0$$

dove la produzione contemporanea indica la **sovrapposizione dei denti di sega**.

Se la **capacità del magazzino è minore del lotto economico**, la soluzione ottima sarà la nostra capacità e non più q^* .

Per il **vincolo del capitale**, indiciamo con c_A e c_B il valore di un singolo prodotto di A e B allora:

$$c_A q_A + c_B q_B \leq C$$

con C capitale massimo.

Dal **punto di vista grafico** avremo 2 variabili q_A e q_B con dei vincoli sono di tipo lineare:

$$q_A + q_B \leq Q \quad c_A q_A + c_B q_B \leq C$$

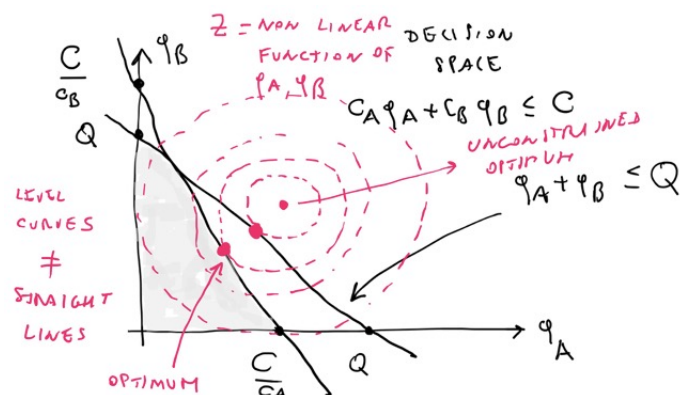


Figura 17. Grafico con linee di livello non lineari

La **funzione obiettivo non è lineare** dato che q_A e q_B sono al denominatore. Le **curve di livello non sono delle rette** ma saranno concentriche (su ognuna il costo è sempre costante).

Avremo una **soluzione ottima dalla curva di livello tangente all'insieme di ammissibilità**. Si vado allora a prendere le curve peggiori fino ad arrivare a quella che interseca la regione ottima.

H. Algoritmo del simplesso

Ogni problema di ottimizzazione lineare si può **trasformare in forma standard**. Prendiamo un f.o.:

$$\min z = \underline{c}^T \underline{x}$$

e dei vincoli:

- $\underline{A} \underline{x} = \underline{b}$
- $\underline{x} \geq 0$
- $n > m$

con n righe e m colonne delle matrice \underline{A} .

Utilizziamo l'**operazione di Pivot**, quindi il vincolo:

$$\underline{A} \underline{x} = \underline{b}$$

sarà:

$$\begin{bmatrix} 4 & 1 & 5 & 7 \\ 2 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

Il Pivot che ci viene assegnato è:

$$(r, s) = (1, 3)$$

alla quale coordinata corrisponde il valore della prima riga e terza colonna: 5.

Dai dati **creiamo un sistema di eq lineari**:

$$\begin{cases} 4x_1 + x_2 + 5x_3 + 7x_4 = 10 \\ 2x_1 + 3x_2 + 2x_3 + 1x_4 = 10 \end{cases}$$

Per eseguire l'operazione di Pivot andremo a far sì che **in corrispondenza della colonna di Pivot (s) ci sia solo un vettore unitario**:

$$\begin{bmatrix} ? & ? & 1 & ? \\ ? & ? & 0 & ? \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} ? \\ ? \end{bmatrix}$$

Gli steps da seguire sono:

1) facciamo $\frac{r}{a_{rs}}$, con:

- r : riga Pivot
- $a_{rs} \neq 0$: valore che si trova dal Pivot, nel nostro caso 5

Diremo quindi che:

$$a_{rj} = \frac{a_{rj}}{a_{rs}} \quad \forall \quad j = 1, \dots, n+1$$

$$\begin{bmatrix} \frac{4}{5} & \frac{1}{5} & 1 & \frac{7}{5} \\ ? & ? & 0 & ? \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ ? \end{bmatrix}$$

2) per ogni riga non Pivot $i \neq r$ applichiamo il principio di equivalenza per eq non lineari:

$$\text{riga } i = \text{riga } i + (-a_{is}) * \text{nuova riga } r$$

in questo caso andiamo a sommare -2 in modo da avere la configurazione $(1, 3) = 1$ e $(2, 3) = 0$.

Diremo quindi che:

$$a_{ij} = a_{ij} + (-a_{is})a_{rj} \quad \forall \quad i = 1, \dots, m; \quad i \neq r$$

col 1	col 2	col 3	col 4	ris
2	3	2	1	10 +
$-\frac{8}{5}$	$-\frac{2}{5}$	-2	$-\frac{14}{5}$	-4 =
$\frac{2}{5}$	$\frac{13}{5}$	0	$-\frac{9}{5}$	6

quindi abbiamo:

$$\begin{bmatrix} \frac{4}{5} & \frac{1}{5} & 1 & \frac{7}{5} \\ \frac{2}{5} & \frac{13}{5} & 0 & -\frac{9}{5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

I. Tableau

Possiamo quindi riscrivere la forma standard con il Tableau con una **forma tabellare** del tipo:

$$\underline{\bar{A}} = \begin{bmatrix} \underline{A} & \underline{b} \\ \underline{c}^T & 0 \end{bmatrix}$$

dove m righe e n colonne.

I nostri problemi hanno variabili continue dove usando l'algoritmo del simplesso avremo come **forma generale**:

$$\min c_1 x_1 + \dots + c_n x_n$$

per i vincoli invece:

- $a_{11}x_1 + \dots + a_{an}x_n = b_1$
- ...
- $a_{m1}x_1 + \dots + a_{mn}x_n = b_m$
- $x_1, \dots, x_n \geq 0$

Assumiamo che:

1) $n > m$

2) $\text{rank}(\underline{A}) = n$

così **non avremo vincoli ridondanti**.

Applichiamo poi la definizione di **insieme di base B** (con $\underline{x}_B \in R^m$) ed **insieme non di base N** (con $\underline{x}_N \in R^{n-m}$).

quindi avrò che:

$$\underline{x} = (\underline{x}_B, \underline{x}_N)$$

allora:

$$\underline{\underline{A}} = [\underline{\underline{B}} | \underline{\underline{N}}]$$

Se la matrice B non è singolare posso ricavare una soluzione imponendo

$$\underline{x}_N := 0$$

Per le variabili B avremo:

$$\underline{\underline{B}} \underline{x}_B = \underline{b} \rightarrow \underline{x}_B = \underline{\underline{B}}^{-1} \underline{b}$$

sarà anche ammissibile se ≥ 0

Tutto questo grazie al teorema fondamentale che dice:

- 1) se un problema ha soluzione ammissibile, allora almeno una è di base.
- 2) se ammette soluzioni ottime, c'è n'è almeno una di base

Le soluzioni di base sono quindi più comode dato che sono più piccole in un insieme R^n di soluzioni non ammissibili, ammissibili e ottime.

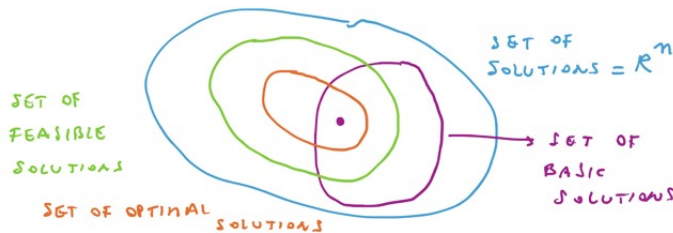


Figura 18. Insiemi delle soluzioni

Capiamo allora che i modi per poter scegliere x_B saranno dati dal numero di combinazioni di n elementi di classe m :

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

J. Esempio Tableau

Funzione obiettivo:

$$\min z = 2x_1 + 3x_2 + 4x_3 - 5x_4$$

vincoli:

- $x_1 - x_2 + x_3 + 2x_4 = 3$
- $2x_2 + x_4 = 7$
- $x_1 + 2x_3 = 10$
- $x_1, x_2, x_3, x_4 \geq 0$

Supponiamo: $m = 3$ e $n = 4$.

e scegliamo sull'insieme di tutte le variabili, combinazioni lineari di tanti numeri quante sono le equazioni. Prendiamo per esempio:

- $\underline{x}_B = (x_1, x_3, x_4)$
- $\underline{x}_N = (x_2)$

Andiamo allora a prendere i termini delle variabili e a inserirli in A :

$$A = \begin{bmatrix} 1 & -1 & 1 & 2 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

Dalla f.o. troviamo:

$$c^T = (2, 3, 4, -5)$$

Andremo poi a distribuire a ogni matrice i suoi dati:

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

$$\underline{c}_B = (2, 4, -5)$$

$$N = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}$$

nel nostro caso c sarà:

$$\underline{c}_N = (3)$$

possiamo allora riscrivere come:

$$\min \underline{c}_B^T \underline{x}_B + \underline{c}_N^T \underline{x}_N$$

K. Forma canonica

Con le operazioni Pivot poniamo il problema in forma canonica rispetto a una base:

$$\begin{bmatrix} 2 & 1 & 4 & -2 & 10 \\ 1 & 2 & 4 & 2 & 10 \\ 10 & 10 & -2 & 2 & 0 \end{bmatrix}$$

dove:

- **vincoli** (prime righe):

$$2x_1 + x_2 + 4x_3 - 2x_4 = 10$$

$$x_1 + 2x_2 + 4x_3 - 2x_4 = 10$$

- **funzione obiettivo** (ultima riga):

$$10x_1 + 10x_2 - 2x_3 + 2x_4 + (-z) = 0$$

quindi la funzione obiettivo è:

$$\min z = 10x_1 + 10x_2 - 2x_3 + 2x_4 + 0$$

e avrà ovviamente tutte le variabili ≥ 0 .

Se effettuo un'operazione di Pivot su $(0, 3)$:

$$\begin{bmatrix} -1 & -0.5 & -2 & 1 & -5 \\ 3 & 3 & 8 & 0 & 20 \\ 12 & 11 & 2 & 0 & 10 \end{bmatrix}$$

ed un'altra operazione di Pivot su $(1, 1)$:

$$\begin{bmatrix} -0.5 & 0 & -0.6 & 1 & -1.6 \\ 1 & 1 & 2.6 & 0 & 6.6 \\ 1 & 0 & -27.6 & 0 & -63.3 \end{bmatrix}$$

Il sistema vincolare è risolto per x_2 e x_4 dato che sono quelle scelte dal Pivot. Poniamo poi $x_1, x_3 := 0$ ed avremo:

$$x_4 = -1.6, x_2 = 6.6$$

quindi avremo una **soluzione base**:

$$\underline{x} = (0, 6.6, 0, -1.6)$$

Questa soluzione è **inammissibile** perché $x_4 \leq 0$ quindi non è Basic Feasible Solution (BFS).

Per esprimere la forma canonica diciamo che gli **indici di colonna delle variabili di B e N** sono:

$$I_B = \langle 1, 3 \rangle$$

$$I_N = \langle 0, 2 \rangle$$

possiamo dire la **posizione delle variabili in base alle equazioni** con:

$$\beta(0) = 3$$

$$\beta(1) = 1$$

Una forma canonica mi dà una **soluzione base** andando a porre le **variabili non di base = 0** e trovando quelle di base.

Il **valore in basso a destra del Tableau** rappresenta il valore:

$$z = x_1 - 27.3x_3 + 63.3$$

dove essendo x_1 e x_3 non di base:

$$\bar{z} = 63.3$$

quindi rappresenta il **valore della soluzione di base associato a questa forma canonica**.

L. Generalizzazione forma canonica

Generalizzando avrò come **funzione obiettivo**:

$$\min z = \sum_{j \in I_N} \bar{c}_j x_j + \bar{z}$$

e con **vincoli**:

$$x_{\beta(i)} + \sum_{j \in I_N} \bar{a}_{ij} x_j = \bar{b}_i \quad \forall \quad i = 0, \dots, m-1$$

con ovviamente $x_j \geq 0, j \in J_N \cup J_B$

L'**equazione di base associata** sarà:

$$x_j := 0 \quad \forall \quad j \in J_N$$

$$x_{\beta(i)} = \bar{b}_i \leq 0 \quad \forall \quad i = 0, \dots, m-1$$

Avremo la **forma generale** quando è \leq e $\bar{b} \geq 0$.

Per il **test di ottimalità in forma canonica**:

$$\min z = \sum_{j \in I_N} \bar{c}_j x_j + \bar{z}$$

il che significa che per \underline{x} la soluzione ammissibile sarà:

$$z(\underline{x}) = \sum_{j \in I_N} \bar{c}_j x_j + \bar{z}$$

allora **se $\bar{c}_j \geq 0$ avremo una forma ottimale**, dato che $z(\underline{x}) \geq \bar{z}$

Se **parto dalla soluzione di base ammissibile e prendo una variabile fuori base rendendola positiva**, la funzione obiettivo varia in base alla relazione:

$$z = \sum_{j \in J_N} \bar{c}_j \dots (\text{slide})$$

cioè $x_j = 0 \rightarrow 1$

Avremo quindi che la variazione della funzione obiettivo è uguale al coefficiente di posto ridotto \bar{c}_j : $\Delta z = \bar{c}_j$.

Riusciamo così a **diminuire z** il che ci piace perché stiamo minimizzando.

Se il **test di ottimalità fallisce** perché esiste una variabile di base con coefficiente di costo ridotto negativo potremo avere 2 situazioni:

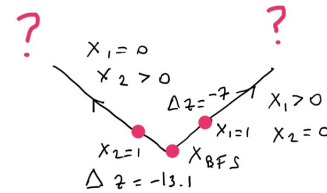


Figura 19. Soluzioni per ottimalità

Se ho più variabili di base negative allora devo capire **quale delle due devo perturbare**, cioè rendere > 0 , e la scegliamo con un **"euristica"** che è una scelta algoritmica basata sull'intuizione e info pregresse.

Sceghieremo come variabile, detta variabile entrante, quella con **coefficiente di costo più' negativo**:

$$x_s = \min_{j \in I_N} \bar{c}_j$$

quindi se $\Delta z = \bar{c}_s x_s$

Se dobbiamo **minimizzare** avremo l'interesse e far raggiungere a x_s il **valore più' elevato possibile** dato che ci sarà un miglioramento.

Applichiamo un **vincolo sul massimo valore che x_s può' raggiungere**:

$$x_s = 0 \rightarrow > 0$$

$$x_j = 0 \quad \forall \quad j \in I_N, j \neq s$$

allora avremo:

$$\min z = \bar{c}_s x_s + \bar{z}$$

con vincoli:

- $x_{\beta(i)} + \bar{a}_{is} x_s = \bar{b}_i \quad \forall \quad i = 1, \dots, m$
- $x_j \geq 0 \quad \forall \quad j = 1, \dots, m$

avremo quindi che **x_s cresce fino al non superamento dei vincoli**:

$$x_{\beta(i)} = \bar{b}_i - \bar{a}_{is}x_s \geq 0$$

A questo punto avremo 2 casi possibili:

1) avremo:

$$\bar{a}_{is} \leq 0 \quad \forall \quad i = 1, \dots, m$$

allora:

$$x_{\beta(i)} \geq 0 \quad \forall \quad x_s \rightarrow +\infty$$

con: $z \rightarrow -\infty$

2) avremo:

$$\exists i : \bar{a}_{is} > 0$$

per calcolare il massimo valore che x_s può assumere è dato da:

$$\begin{aligned} -\bar{a}_{is}x_s &\geq -\bar{b}_i \\ \Rightarrow x_s &\leq \frac{\bar{b}_i}{\bar{a}_{is}} \quad \forall \quad i = 1, \dots, m; \quad \bar{a}_{is} > 0 \end{aligned}$$

di conseguenza i valori con \bar{a}_{is} negativo non ci danno problemi dato che $x_{\beta(i)}$ sarà positivo lo stesso e poi avremo che che:

$$x_s = \min \frac{\bar{b}_i}{\bar{a}_{is}} \quad \forall \quad i = 1, \dots, m; \quad \bar{a}_{is} > 0$$

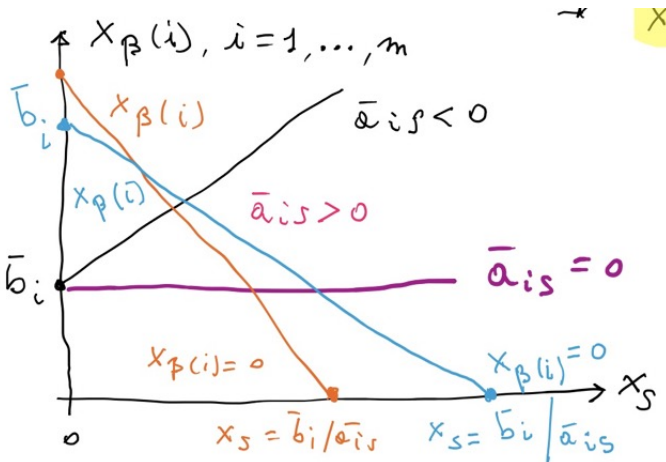


Figura 20. Possibilità con i valori di \bar{a}_{is}

Per $\bar{a}_{is} > 0$ avremo che ad un certo punto x_s arriverà a 0. Ma potrebbe capitare che ci sia un'altra variabile che diventi 0 prima, per un valore inferiore di x_s .

Per esempio se abbiamo una f.o.:

$$\min z = -10x_1 - 10x_2 + 0$$

vincoli:

- $2x_1 + x_2 + x_3 = 10$
- $x_1 + 2x_2 + x_4 = 10$
- $x_1, x_2, x_3, x_4 \geq 0$

BFS:

- $x_1 = x_2 = 0$
- $x_3 = 10$
- $x_4 = 10$
- $\hat{z} = 0$

allora:

$$x_s = \operatorname{argmin}(-10, -10) = x_1$$

annullando x_2, x_3, x_4 :

$$x_2 \leq \min\left(\frac{10}{2}, \frac{10}{1}\right) = 5$$

il che significa che $x_s = x_1$, ed abbiamo:

$$x_3 = 10 - 2x_1 \geq 0$$

$$x_4 = 10 - x_1 \geq 0$$

allora abbiamo:

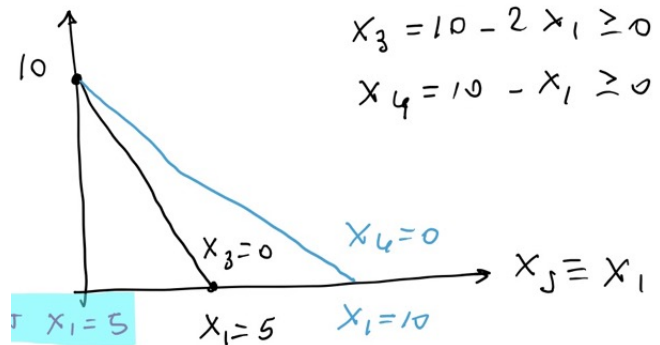


Figura 21. Permutazione di x_1

Dopo la nostra perturbazione $x_1 = 0 \rightarrow 5$ e $x_3 = 10 \rightarrow 0$, per determinare quale sia la variabile, detta variabile uscente, che si annulla per prima:

$$x_r = \operatorname{argmin}\left(\frac{10}{2}, \frac{10}{1}\right) = x_3$$

Pseudocodice dell'algoritmo del semplice:

trovo la prima BFS

while $\exists \bar{c}_j < 0 \quad \forall \quad j \in I_N$:

trovo argmin per "c" del Pivot: $x_s = \operatorname{argmin}_{j \in I_N} \bar{c}_j$
if $\bar{a}_{is} \leq 0$:

abbiamo un problema unbounded

trovo argmin per "r" del Pivot: $x_s = \operatorname{argmin}_{i=1, \dots, m} \frac{\bar{b}_i}{\bar{a}_{is}}$

eseguo il Pivot

se trovo una soluzione di base ammissibile ottima faccio un'analisi per capire se è unica o meno

M. Esercizio forma canonica

Funzione obiettivo:

$$\max z = 10x_1 + 10x_2$$

vincoli:

- $2x_1 + x_2 \leq 10$
- $x_1 + 2x_2 \leq 10$
- $x_1, x_2 \geq 0$

gradiente:

$$\nabla z = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

grafico:

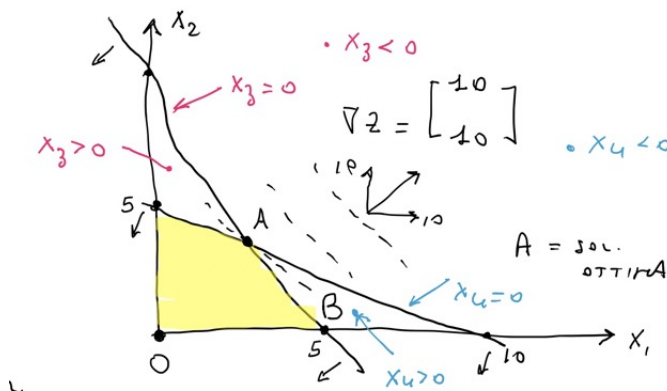


Figura 22. Graficazione dei vincoli

Forma standard:

$$\min z = -10x_1 - 10x_2$$

vincoli:

- $2x_1 + x_2 + x_3 = 10$
- $x_1 + 2x_2 + x_4 = 10$
- $x_1, x_2, x_3, x_4 \geq 0$

avremo che:

- $A = (3.\bar{3}, 3.\bar{3}, 0, 0)$
- $B = (5, 0, 0, > 0)$
- $O = (0, 0, > 0, > 0)$

dove nei vertici abbiamo 2 variabili > 0 . Quindi ad ogni vertice abbiamo una BFS con cardinalità 1 : M.

Tableau:

$$\begin{bmatrix} 2 & 1 & 1 & 0 & 10 \\ 1 & 2 & 0 & 1 & 10 \\ -10 & -10 & 0 & 0 & 0 \end{bmatrix}$$

con variabili in base B e non in base N:

- $J_B = \langle X_3, x_4 \rangle$
- $J_N = \langle X_1, x_2 \rangle$

BFS:

- $x_1 = x_2 = 0$

- $x_3 = 10$
- $x_4 = 10$
- $\bar{z} = 0$

Il test di ottimalità fallisce e quindi possiamo avere una soluzione migliore. Applico l'euristica del valore di costo ridotto. Sceglio a caso $x_s = x_1$.

Prendiamo la colonna x_1 e in base al valore unitario presente in x_3 e x_4 allora avremo:

$$\min\left(\frac{10}{2}, \frac{10}{1}\right) = 5$$

non posso avere che $x_3 > 5$ dato che si deve fermare per non diventare negativo, cioè non ammissibile, dato che superiamo in punto B.

Allora dato che $x_3 = 0$ e x_1 prende il suo posto avremo che le variabili di base cambiano in:

$$J_B = \langle x_1, x_4 \rangle$$

Dobbiamo allora effettuare un'operazione di Pivot:

- divido per 2 la riga 1:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 5 \\ 1 & 2 & 0 & 1 & 10 \\ -10 & -10 & 0 & 0 & 0 \end{bmatrix}$$

- multiplico per -1 la prima riga e poi sommo alla seconda:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 5 \\ 0 & 1.5 & -0.5 & 1 & 5 \\ -10 & -10 & 0 & 0 & 0 \end{bmatrix}$$

- multiplico per -10 la prima riga e poi sommo alla terza:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 5 \\ 0 & 1.5 & -0.5 & 1 & 5 \\ 0 & -5 & 5 & 0 & 50 \end{bmatrix}$$

allora abbiamo che la soluzione di base ammissibile (BFS) è:

- $x_2 = x_3 = 0$
- $x_1 = 5$
- $x_4 = 5$
- $\bar{z} = -50$

Nuovamente il test di ottimalità fallisce per la presenza di -5, quindi la variabile entrante è $x_s = x_2$ quindi:

$$x_r = \argmin\left(\frac{5}{0.5}, \frac{5}{1.5}\right) = x_4$$

il che corrisponde a stare in B dove $x_2 = 0$, quindi ci spostiamo in direzione di $A = (> 0, > 0, 0, 0)$. Avremo allora:

$$J_B = \langle x_1, x_2 \rangle$$

dato che abbiamo x_2 entrante e x_4 uscente.

Il perno del nuovo Pivot sarà 1.5:

$$\begin{bmatrix} 1 & 0 & 0.\bar{6} & -0.\bar{3} & 3.\bar{3} \\ 0 & 1 & -0.\bar{3} & 0.\bar{6} & 3.\bar{3} \\ 0 & 0 & 3.\bar{3} & 3.\bar{3} & 66.\bar{6} \end{bmatrix}$$

avremo allora:

- $x_3 = x_4 = 0$

- $x_1 = 3.\bar{3}$
- $x_2 = 3.\bar{3}$
- $\bar{z} = -66.\bar{6}$

dove abbiamo ottenuto una **soluzione ottima**.

N. Soluzioni ottime multiple

Per individuarle ricordiamo che esiste una soluzione ottima se:

$$\bar{c}_j \geq 0$$

esistono soluzioni multiple se esiste $j^* \in J_N$ e quindi se ha:

- coefficienti di costo ridotti > 0
- coefficienti delle **variabili non di base** sono > 0

allora **almeno uno sarà $= 0$** . Facendo il solito ragionamento prendo una variabile $x_{j^*} = 0 \rightarrow > 0$, allora avrò che:

$$\Delta z = \bar{c}_{j^*} x_{j^*} = 0$$

O. Esempio soluzioni ottime multiple

Funzione obiettivo:

$$\max z = 20x_1 + 10x_2$$

vincoli:

- $2x_1 + x_2 \leq 10$
- $x_1 + 2x_2 \leq 10$
- $x_1, x_2 \geq 0$

gradiente:

$$\nabla z = \begin{bmatrix} 20 \\ 10 \end{bmatrix}$$

grafico:

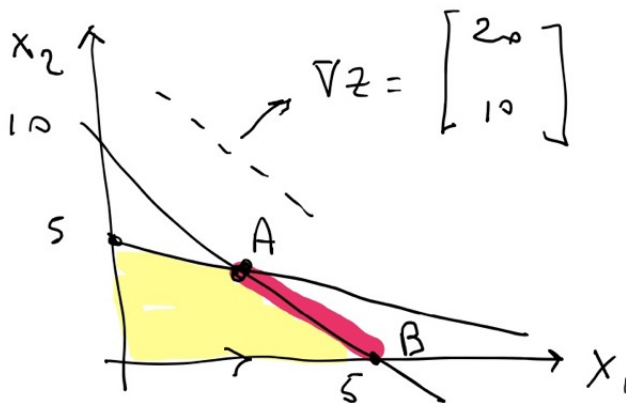


Figura 23. Graficazione dei vincoli

tableau:

$$\begin{bmatrix} 2 & 1 & 1 & 0 & 10 \\ 1 & 2 & 0 & 1 & 10 \\ -20 & -10 & 0 & 0 & 0 \end{bmatrix}$$

BFS:

- $x_1 = x_2 = 0$
- $x_3 = 10$
- $x_4 = 10$
- $\bar{z} = 0$

Usiamo le operazioni di Pivot:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 5 \\ 0 & 1.5 & -0.5 & 1 & 5 \\ 0 & 0 & 10 & 0 & 100 \end{bmatrix}$$

BFS:

$$B = \begin{cases} x_2 = x_3 = 0 \\ x_1 = 5 \\ x_4 = 5 \\ \bar{z} = -100 \end{cases}$$

Il **test di ottimalità** sulle variabili non di base $J_N = \{x_2, x_3\}$ è OK, **ma non c'è una soluzione unica** dato che **abbiamo 0 per x_2** che andrà **perturbata** assegnandole un valore > 0 . Allora la variazione della funzione obiettivo per una perturbazione è:

$$\bar{c}_{j^*} x_{j^*} \quad \forall \quad \bar{c}_{j^*} = 0, x_{j^*} > 0$$

allora **lungo lo spigolo \overline{AB} avremo un costo minimo**. **Forziamo il Pivot in x_2** facendo uscire x_4 :

$$\begin{bmatrix} 1 & 0 & 0.\bar{6} & -0.\bar{3} & 3.\bar{3} \\ 0 & 1 & -0.\bar{3} & -0.\bar{6} & 3.\bar{3} \\ 0 & 0 & 10 & 0 & 100 \end{bmatrix}$$

BFS:

$$A = \begin{cases} x_3 = x_4 = 0 \\ x_1 = 3.\bar{3} \\ x_2 = 3.\bar{3} \\ \bar{z} = -100 \end{cases}$$

Abbiamo generato i due punti estremi A e B riuscendo ad avere **infinito soluzioni ottime** dato che ci muoveremo nel segmento \overline{AB} . Al massimo potranno essere:

$$\binom{n}{m}$$

P. Algoritmo della convergenza

Abbiamo che **se c_s è la variabile entrante** allora il costo è dato dalla relazione:

$$\Delta z = \bar{c}_s \min_{i=1, \dots, m} \frac{\bar{b}_i}{\bar{a}_{is}} \quad \forall \quad \bar{a}_{is} > 0$$

$$\text{con } \bar{c}_s < 0 \text{ e } \min_{i=1, \dots, m} \frac{\bar{b}_i}{\bar{a}_{is}} \geq 0$$

L'algoritmo del simplesso si comporta in modo decrescente e ad ogni iterazione genera una nuova BFS. Se ad ogni iterazione $\Delta z < 0$ vuol dire che avremo come casi sfavorevoli:

- $\min < 0$: l'algoritmo visita tutte le BFS
- $\min = 0$: uno dei numeratori è nullo

Se abbiamo che $x_s = 0$ esisterà un termine noto $\bar{b}_i = 0$ per $\bar{a}_{is} > 0$.

Avremo allora che se non abbiamo variazioni di z allora non ci sarà un miglioramento, dato che l'algoritmo è deterministico genererà la stessa sequenza all'infinito. Questo fenomeno è detto **cycling** e avremo che l'algoritmo non converge.

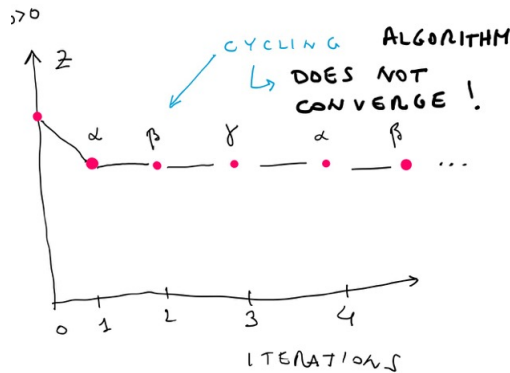


Figura 24. Cycling algorithm result

Potrebbe anche capitare di avere un miglioramento dopo n iterazioni.

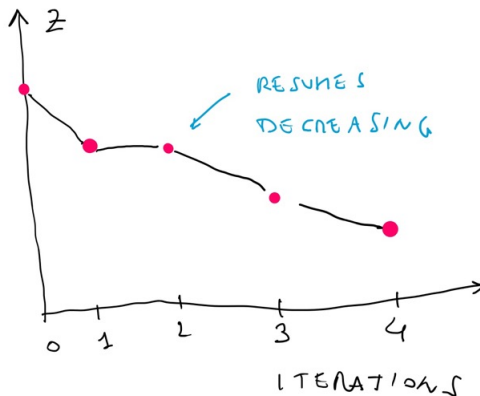


Figura 25. Not cycling algorithm result

Q. Esempio BFS degenera

Funzione obiettivo:

$$\max z = 10x_1 + 10x_2$$

vincoli:

- $x_1 - x_2 \leq 0$
- $x_2 \leq 1$
- $x_1, x_2 \geq 0$

grafico:

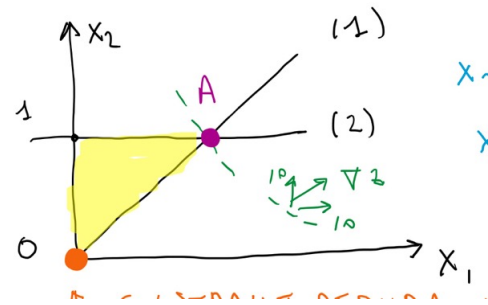


Figura 26. Grafico di un sistema inizialmente degenera

tabelau:

$$\begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ -10 & -10 & 0 & 0 & 0 \end{bmatrix}$$

dove $\Delta z = 0 * x_1 = 0 * 0 = 0$

Notiamo di avere una variabile in base = 0 che sarà detta **variabile degenera**. Dato che abbiamo dei termini negativi, nell'ultima riga, allora prendiamo quello con numero piu' negativo, in questo caso anche con l'indice piu' negativo $x_s = x_1$, allora:

$$x_r = \operatorname{argmin}\left(\frac{0}{1}\right) = x_3$$

con $x_1 \leq \min\left(\frac{0}{1}\right) = 0$

Notiamo che abbiamo preso x_3 dato che in corrispondenza del numero piu' grande abbiamo il valore unitario della colonna di x_3 .

Abbiamo un'altra soluzione degenera quindi ci troviamo su un **plateau**. Graficamente abbiamo che x_1 rimane 0 nel punto (0,0), avremo allora che nel segmento \overline{OA} ci sarà un **numero ridondante e superfluo di vincoli**.

$$\begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & -20 & 10 & 0 & 0 \end{bmatrix}$$

Dove seguendo i motivi detti prima avremo $x_s = x_2$, allora:

$$x_r = \operatorname{argmin}\left(\frac{1}{1}\right) = x_4$$

con $x_2 \leq \min\left(\frac{1}{1}\right) = 1$

Abbiamo quindi che:

$$\Delta z = -20 * x_2 = -20 * 1 = -20$$

prendendo come perno il valore più alto della colonna x_2 avremo un miglioramento:

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 10 & 20 & 20 \end{bmatrix}$$

quindi con $\Delta z = -20$.

R. Regola dell'anticycling: Bland rule

Possiamo evitare di avere variabili degeneri usando la Bland Rule. Andremo semplicemente, a parità di valore, a prendere quello con indice più negativo. Quindi in:

S. Variabili artificiali

Iniziamo aggiungendo ad ogni vincolo una variabile artificiale:

$$\sum_j a_{ij}x_j + \alpha_i = b_i \quad \forall \quad i = 1, \dots, m$$

con $\alpha_i \geq 0 \quad \forall \quad i = 1, \dots, m$.

Indicato con P_a il problema artificiale, la sua regione ammissibile sarà:

$$\Omega(P_a) = \{x \in R^m : \underline{A}x + \underline{\alpha} = \underline{b}, \underline{x} \geq 0, \underline{\alpha} \geq 0, \underline{\alpha} \in R^m\}$$

Invece per il problema di partenza P , la regione ammissibile è:

$$\Omega(P) = \{x \in R^m : \underline{A}x = \underline{b}, \underline{x} \geq 0\}$$

Avremo allora che:

$$\Omega(P_a) \geq \Omega(P) \Leftrightarrow x \in \Omega(P) \Rightarrow x \in \Omega(P_a)$$

La soluzione potrebbe comunque non essere ammissibile per P . Allora cerchiamo una soluzione equivalente ma che sia valida sia per P_a che per P .

Il problema artificiale (teorema 1) avrebbe come funzione obiettivo:

$$\min \rho = \underline{e}^T \underline{\alpha} \quad \forall \quad e = [1, \dots, 1]$$

vincoli:

- $\underline{A}x + \underline{\alpha} = \underline{b}$
- $\underline{x}, \underline{\alpha} \geq 0$

Notare che ammette sempre soluzioni ottime.

Per il teorema 2 avremo che P è ammissibile $\Leftrightarrow P_a$ ha $\rho^* = 0$.

Dati questi 2 problemi avremo che per:

- 1) $\rho^* > 0$: il problema P è inammissibile
- 2) $\rho^* = 0$: alla soluzione ottima P_a corrisponde una BFS per P

T. Metodo del simplesso in 2 fasi

Per prima cosa risolviamo il P_a con il metodo del simplesso. Se $\rho^* = 0$ usiamo la soluzione ottima di P_a senza le variabili artificiali. Prenderemo poi la f.o. z come BFS per P e da questa applicare il metodo del simplesso.

U. Costruzione del problema artificiale

Funzione obiettivo:

$$\min z = \sum_{j=1}^n c_j x_j$$

vincoli:

- $\sum_{j=1}^n a_{ij}x_j \leq b_i \quad \forall \quad i = 1, \dots, m_1$
- $\sum_{j=1}^n a_{ij}x_j \geq b_i \quad \forall \quad i = m_1 + 1, \dots, m_2$

- $\sum_{j=1}^n a_{ij}x_j = b_i \quad \forall \quad i = m_2 + 1, \dots, m$
- con $x_j \geq 0 \quad \forall \quad j = 1, \dots, m$.

Forma standard abbiamo che la funzione obiettivo:

$$\min z = \sum_{j=1}^n c_j x_j$$

vincoli:

- $\sum_{j=1}^n a_{ij}x_j + x_{n+i} = b_i \quad \forall \quad i = 1, \dots, m_1$
- $\sum_{j=1}^n a_{ij}x_j - x_{n+i} = b_i \quad \forall \quad i = m_1 + 1, \dots, m_2$
- $\sum_{j=1}^n a_{ij}x_j = b_i \quad \forall \quad i = m_2 + 1, \dots, m$

con

- $x_j \geq 0 \quad \forall \quad j = 1, \dots, n$
- $x_{n+i} \geq 0 \quad \forall \quad i = 1, \dots, m_2$

Per fare in modo di avere un problema artificiale con il minor numero di variabili artificiali dovremo:

- 1) per $i = 1, \dots, m_1$ NON aggiungiamo variabili artificiali, dato che abbiamo quelle di slack
- 2) per $i = m_2 + 1, \dots, m$ aggiungo una variabile artificiale per ogni vincolo
- 3) per $i = m_1 + 1, \dots, m_2$ aggiungo una α_0 in ogni vincolo

Indichiamo un l'indice $h (m_1 + 1 \leq h \leq m_2)$ t.c.:

$$h = \operatorname{argmax}_i \{b_i, i = m_1 + 1, \dots, m_2\}$$

Rimpiazziamo ogni vincolo $i \neq h$ con un vincolo equivalente:

$$\sum (a_{hj} - a_{ij})x_j - x_{n+h} + x_{n+i} = b_n - b_i \quad \forall \quad i = m_1 + 1, \dots, m_2, i \neq h$$

avremo allora:

$$\sum a_{hj}x_j - x_{n+h} + \alpha_0 = b_n$$

$$\sum a_{ij}x_j - x_{n+i} + \alpha_0 = b_i$$

V. Esempio problema artificiale

Funzione obiettivo:

$$\min z = 4x_1 + 5x_2$$

vincoli:

- $x_1 - x_3 = 6$
- $x_2 - x_4 = 4$
- $x_1 + 3x_2 = 21$
- $x_1, x_2, x_3, x_4 \geq 0$

Il problema artificiale è dato dalla funzione obiettivo:

$$\min \rho = \alpha_1 + \alpha_2$$

con vincoli:

- $x_1 - x_3 + \alpha_1 = 6$
- $x_2 - x_4 + \alpha_1 = 4$
- $x_1 + 3x_2 + \alpha_2 = 21$
- $x_1, x_2, x_3, x_4, \alpha_1, \alpha_2 \geq 0$

Vado ora a sostituire al vincolo 2 la differenza tra il vincolo 1 e il 2:

$$x_1 - x_2 - x_3 + x_4 = 2$$

Avremo allora che P_a e' data da una funzione obiettivo:

$$\min \rho = \alpha_1 + \alpha_2$$

e dai vincoli:

- $x_1 - x_3 + \alpha_1 = 6$
- $x_1 - x_2 - x_3 + x_4 = 2$
- $x_1 + 3x_2 + \alpha_2 = 21$
- $x_1, x_2, x_3, x_4, \alpha_1, \alpha_2 \geq 0$

Applichiamo al **prima fase del metodo a 2 fasi**:

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 0 & 6 \\ 1 & -1 & -1 & 1 & 0 & 0 & 2 \\ 1 & 3 & 0 & 0 & 0 & 1 & 21 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Per poter **ripristinare la forma canonica** dovremo avere i **coefficienti di $\alpha_1, \alpha_2 = 0$** . Tramite vari **operazioni di Pivot** arriviamo a non avere più in base α_1 e α_2 facendo:

$$\text{f.o.} - \text{vincolo di } \alpha_1 - \text{vincolo di } \alpha_2$$

In fine avremo:

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 6 \\ 0 & 0 & 0.3 & 1 & 1 \\ 0 & 1 & 0.3 & 0 & 5 \\ 0 & 0 & 2.3 & 0 & -49 \end{bmatrix}$$

Avremo allora **verificato il criterio di arresto**, perciò avremo una soluzione ottima:

$$x^* = (6, 5, 0, 1), z^* = 49$$

W. Esercitazione variabili artificiali

Usando il metodo delle variabili artificiali, risolvere il seguente problema con funzione obiettivo:

$$\min z = 3x_1 - 7x_2 + 2x_3$$

con vincoli:

- $2x_1 + 3x_2 + 5x_3 \leq 15$
- $x_1 + x_2 + x_3 \geq 1$
- $-x_1 + x_2 = 5$
- $x_1, x_2, x_3 \geq 0$

il problema in forma standard avrà come funzione obiettivo:

$$\min z = 3x_1 - 7x_2 + 2x_3$$

con vincoli:

- $2x_1 + 3x_2 + 5x_3 + x_4 = 15$
- $x_1 + x_2 + x_3 - x_5 = 1$
- $-x_1 + x_2 = 5$
- $x_1, x_2, x_3 \geq 0$

Il problema artificiale è dato dalla funzione obiettivo:

$$\min \rho = \alpha_1 + \alpha_2$$

con vincoli:

- $2x_1 + 3x_2 + 5x_3 + x_4 = 15$
- $x_1 + x_2 + x_3 - x_5 + \alpha_1 = 1$
- $-x_1 + x_2 + \alpha_2 = 5$

- $x_1, x_2, x_3, \alpha_1, \alpha_2 \geq 0$

Avremo allora:

$$\begin{bmatrix} 2 & 3 & 5 & 1 & 0 & 0 & 0 & 15 \\ 1 & 1 & 1 & 0 & -1 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Per ripristinare la forma canonica α_1 e α_2 devono essere 0 facendo:

$$\text{f.o.} - \text{vincolo 2}$$

al risultato:

$$\text{risultato} - \text{vincolo 3}$$

$$\begin{bmatrix} 2 & 3 & 5 & 1 & 0 & 0 & 0 & 15 \\ 1 & 1 & 1 & 0 & -1 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & -2 & -1 & 0 & 1 & 0 & 0 & -6 \end{bmatrix}$$

Ora dobbiamo andare a far uscire le due variabili α_1 e α_2 . Non essendo una soluzione ottima entriamo in base x_2 ($s = 2$):

$$\min\left(\frac{15}{3}, \frac{1}{1}, \frac{5}{1}\right) = 1$$

quindi abbiamo $r = 2$ facendo uscire α_1 . Facendo il Pivot abbiamo:

$$\begin{bmatrix} -1 & 0 & 2 & 1 & 3 & -3 & 0 & 12 \\ 1 & 1 & 1 & 0 & -1 & 1 & 0 & 1 \\ -2 & 0 & -1 & 0 & 1 & -1 & 1 & 4 \\ 2 & 0 & 1 & 0 & -1 & -2 & 0 & -4 \end{bmatrix}$$

che non è ottima. Entra in base x_5 :

$$\min\left(\frac{12}{3}, \frac{4}{1}\right) = 4$$

Notiamo che, avendo valori uguali, ci conviene prendere $r = 3$ in modo che se ne vada α_1 :

$$\begin{bmatrix} 5 & 0 & 5 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 5 \\ -2 & 0 & -1 & 0 & 1 & -1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

che è soluzione ottima per il problema artificiale.

Dato che $\rho = 0$, la soluzione ottima di (P_a) è ammissibile per (P) . Eliminiamo la colonna di α_1 e riscriviamo la f.o. originale:

$$\begin{bmatrix} 5 & 0 & 5 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 5 \\ -2 & 0 & -1 & 0 & 1 & 4 \\ 3 & -7 & 2 & 0 & 0 & 0 \end{bmatrix}$$

si porta in forma canonica f.o. portando a 0 il valore di x_2 della f.o., dato che abbiamo solo 2 variabili in base e 3 vincoli, mentre **serve avere tante variabili in base quanti sono i vincoli**. Allora avremo:

$$(\text{riga } 2 * 7) + \text{f.o.}$$

$$\begin{bmatrix} 5 & 0 & 5 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 5 \\ -2 & 0 & -1 & 0 & 1 & 4 \\ -4 & 0 & 2 & 0 & 0 & 35 \end{bmatrix}$$

dove quindi eseguiamo il Pivot in a_{11} ed avremo:

$$\begin{bmatrix} 1 & 0 & 1 & 0.2 & 0 & 0 \\ 0 & 1 & 1 & 0.2 & 0 & 5 \\ 0 & 0 & -1 & 0.4 & 1 & 4 \\ 0 & 0 & 6 & 0.8 & 0 & 35 \end{bmatrix}$$

VI. BRANCH-AND-BOUND TECHNIQUE FOR SOLVING INTEGER PROGRAMS

A. Principi dell'algoritmo Branch-and-Bound

I problemi che cerchiamo di risolvere sono **lineari con variabili decisionali**, il problema è che **alcune variabili possono essere intere**, o anche binarie, al posto di essere continue. Per questi casi abbiamo alcuni **approcci naive** da usare:

- 1) **brute force**: se abbiamo n variabili binarie le soluzioni ammissibili sono 2^n . Andremo quindi a **generare tutte le soluzioni e verificarne i vincoli** sono soddisfatti (non computazionalmente possibile)
- 2) **rounding**: sostituisco le variabili intere con un "rilassamento" del vincolo di integrità. Quindi il problema P :

$$\max z = 3x_1 + 4x_2 - 2x_3 + 3x_4 - 2x_5$$

con vincoli:

- $2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 \leq 4$
- $x_1, x_2, x_3, x_4, x_5 = 0/1$

diventa $R(P)$:

$$\max z = 3x_1 + 4x_2 - 2x_3 + 3x_4 - 2x_5$$

con vincoli:

- $2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 \leq 4$
- $0 \leq x_1, x_2, x_3, x_4, x_5 \leq 1$

Avremo un **caso fortunato** se la soluzione di $R(P)$ è anche quella di P .

B. Approccio Branch-and-Bound

È detta tecnica di enumerazione parziale dove **ci si procura una stima del valore della soluzione ottima rilassando il vincolo di integrità** sulle variabili intere.

Nella sua risoluzione tramite blackbox potremo avere **4 casi**:

- 1) $R(P)$ inammissibile $\rightarrow P$ inammissibile:

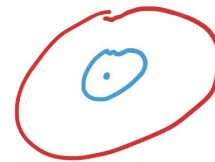


Figura 27. $R(P)$ che contiene P

- 2) $R(P)$ unbounded $\rightarrow P$ può essere unbounded (dove abbiamo variabili intere):

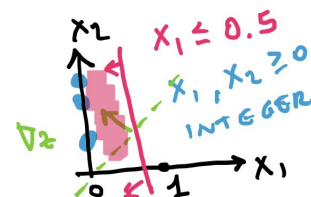


Figura 28. In caso di P unbounded

- 3) $R(P)$ unbounded \rightarrow **P può essere inammissibile** (dove non abbiamo variabili intere):

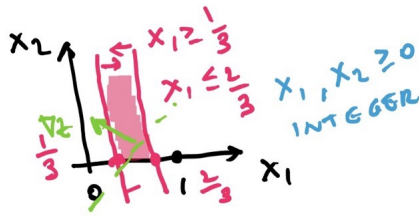


Figura 29. In caso di P inammissibile

(p.s: non posso saperlo dato che la blackbox mi sa solo $R(P)$)

- 4) **$R(P)$ ha soluzione ottima** $\rightarrow P$ inammissibile

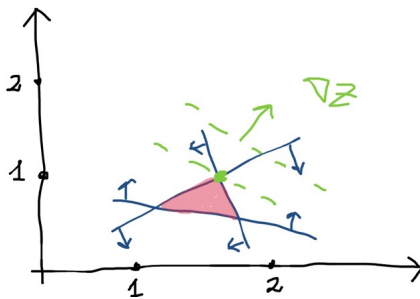


Figura 30. $R(P)$ ottima ma P inammissibile

(p.s: non posso saperlo dato che la blackbox mi sa solo $R(P)$)

- 5) **$R(P)$ ha soluzione ottima** $\rightarrow P$ ammissibile

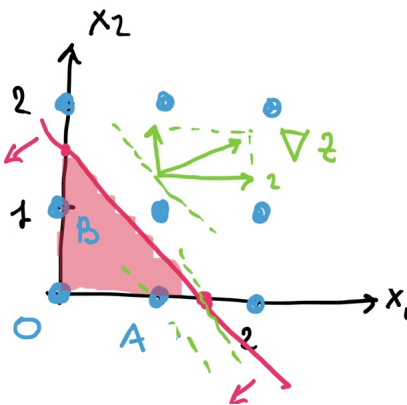


Figura 31. $R(P)$ ottima ma P ammissibile

dove avremo come **soluzioni intere**:

$$O(0,0), A(1,0), B(0,1)$$

Andrò allora a **traslare al curva di livello** fino a trovare la **soluzione $R(P)$** : **primo numero ammissibile** nella regione ammissibile, neceve **per P : primo numero intero ammissibile**

La **soluzione ottima di ottimizzazione** sara' quella di $R(P)$ di rilassamento continuo, perché **ha meno vincoli** fornendo

una soluzione migliore o uguale della nostra stima. Allora se massimizziamo:

$$Z_{R(P)}^* \geq Z_P^*$$

se sto minimizzando:

$$Z_{R(P)}^* \leq Z_P^*$$

dove avrò rispettivamente **upper-bound** e **lower-bound**.

C. Branching

Oltre ad upper-bound e lower-bound su effettua un **branch** cioè una **suddivisione del problema in sottoproblemi**, dove per le soluzioni ammissibili (s.a.):

- P s.a. = P_1 s.a. \cup P_2 s.a.
- P_1 s.a. \cap P_2 s.a. = insieme vuoto
- $\underline{x}^* \notin P_1$ s.a., P_2 s.a.

con \underline{x}^* possibile vincolo applicabile a x_1 o x_2 . Infatti per un problema a 2 variabili abbiamo:

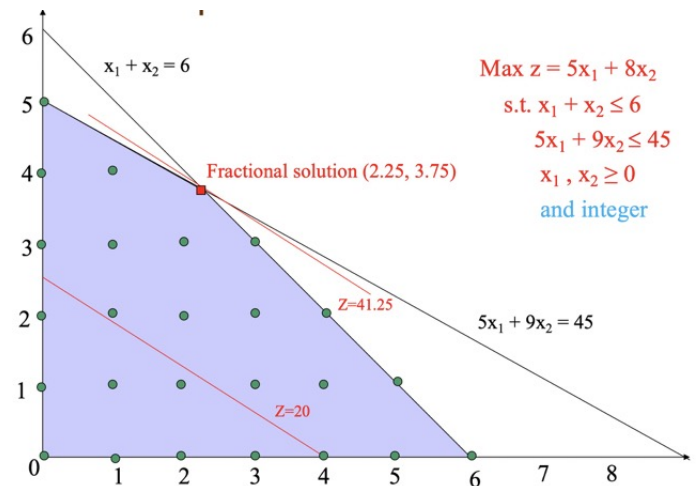


Figura 32. Problema di branching

con soluzione ottima in (2.25, 3.75) con upper: $z = 41.25$.

Avendo dei valori frazionari effettuo un branch per creare dei sottoproblemi. Avendo la soluzione in (2.25, 3.75) allora effettuo il "troncamento" in:

$$x_2 \leq 3, x_2 \geq 4$$

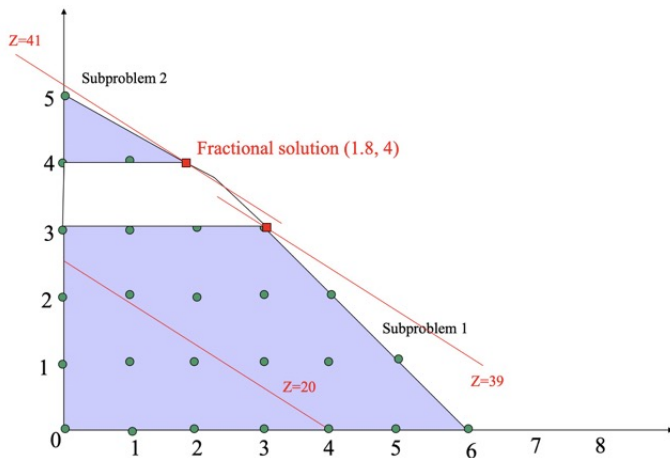


Figura 33. Primo child

Quindi per il sottoproblema P_1 con f.o:

$$\max z = 5x_1 + 8x_2$$

e vincoli:

- $x_1 + x_2 \leq 6$
- $5x_1 + 9x_2 \leq 45$
- $x_2 \leq 3$
- $x_1, x_2 \geq 0$

che avrà come soluzione (3, 3) con upperbound $z = 39$.

invece per il sottoproblema P_2 con f.o:

$$\max z = 5x_1 + 8x_2$$

e vincoli:

- $x_1 + x_2 \leq 6$
- $5x_1 + 9x_2 \leq 45$
- $x_2 \geq 4$
- $x_1, x_2 \geq 0$

che avrà come soluzione (1.8, 4) con upperbound $z = 41$.

Avendo numeri interi in P_1 ho una soluzione ottima e non dovrò fare branch su P_1 e dato il suo valore di z che è cerco in P_2 con f.o: risolveremo allora risolvendo il sottoproblema 1 con soluzione 3, 3 con $z = 39$ allora non ho bisogno di esplorare ancora dato che ho già la soluzione ottima minore di quello di P_2 allora: scarto P_1 e continuo i branch su P_2 .

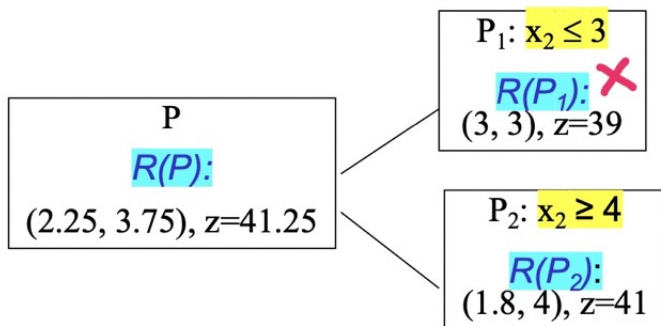


Figura 34. Scarto del primo child

Effettuando il branch avremo:

- P_3 : con vincolo $x_1 \leq 1$
- P_4 : con vincolo $x_1 \geq 2$

tenendo in conto che i child ereditano i vincoli del parent, risolviamo i loro rilassamenti:

- P_4 : $R(P_4)$ inammissibile, dato che non ha intersezioni, allora anche P lo sarà
- P_3 : $R(P_3)$ avrà (1, 4.44) con $z = 40.55$

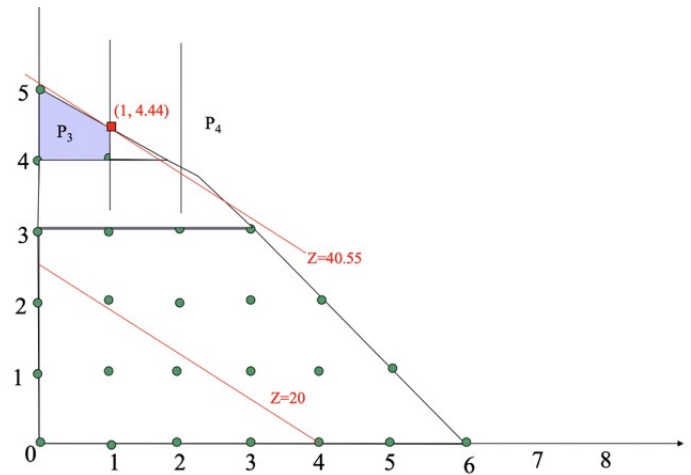


Figura 35. Secondo child

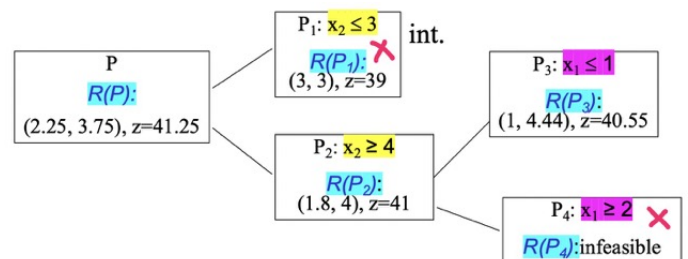


Figura 36. Scarto del secondo child

continuando con il branch avremo:

- P_5 : con $x_2 \leq 4$ con (1, 4) con $z = 37$
- P_6 : con $x_2 \geq 5$ con (0, 5) con $z = 40$

scartiamo allora P_5 che ci fa arrivare ad un massimo di $37 < 40$ di P_6 :

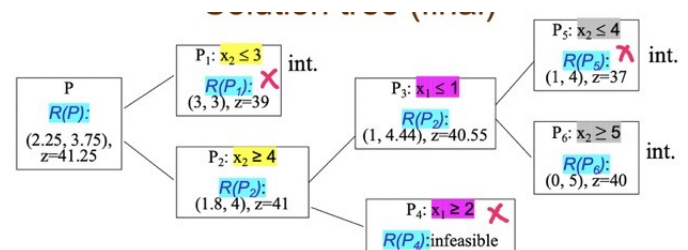


Figura 37. Scarto del terzo child

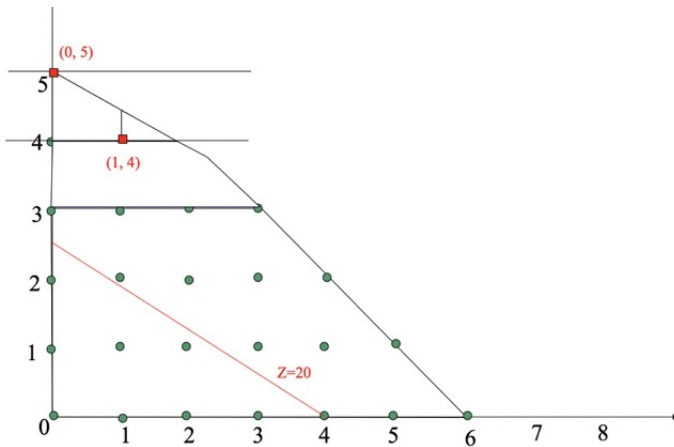


Figura 38. Terzo child

siamo quindi arrivati nella situazione in cui **non ci sono più** branch da creare.

D. Fathoming

Completa l'analisi dei sottoproblemi dicendo che **posso eliminare un sottoproblema** se:

- il **rilassamento e'** inammissibile
- il rilassamento di P ($R(P)$) ha **soluzione ottima intera**, allora è anche soluzione di P
- è verificato il **fenomeno di dominanza**, quindi una z è più grande delle altre

Lo **pseudocodice** è:

```

L := {P} // lista dei sottoproblemi da analizzare
zbest := -∞
xbest := NULL
while L non è vuoto:
    estrarre un sottoproblema k da L
    branching da 1 → nk
    risolvo R(P) e trovo i limiti e l'upperbound UB
    for i = 1 to nk:
        if UBi ≤ zbest then:
            kill childi
        elif R(P) è una soluzione intera:
            zbest := UBi
            xbest := childi
        elif R(P) è una soluzione frazionaria:
            add childi to L
    end for
end while
end while

```

E. Esercizio Branch-and-Bound

Funzione obiettivo:

$$\max z = x_1 + x_2$$

v.o:

- $-6x_1 + 12x_2 \leq 9$

- $6x_1 - 4x_2 \leq 9$
- $x_1, x_2 \geq 0$

gradiente:

$$\nabla z = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

grafico:

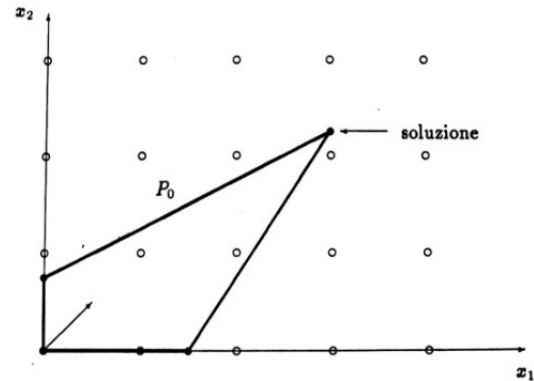


Figura 39. Regione ammissibile

abbiamo che la soluzione ottima (l'intersezione) del risultato continuo è:

$$x_1 = 3, x_2 = 2.25, z = 5.25$$

allora prenderemo come upper-bound 5.

La variabile x_2 ha valore frazionario compreso tra 2 e 3. Effettuiamo un branch con vincoli $x_2 \leq 2$ e $x_2 \geq 3$:

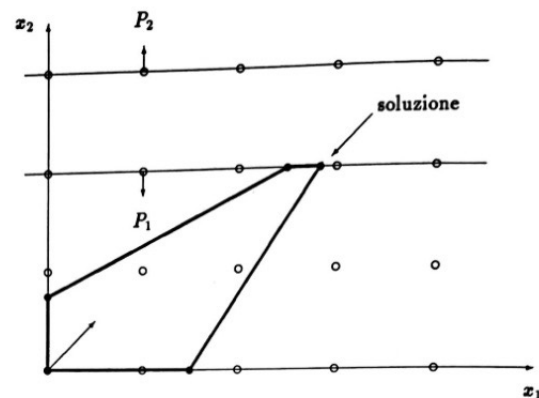


Figura 40. Regione ammissibile

dove:

- P_2 non abbiamo intersezioni quindi è inammissibile
- P_1 con soluzione ottima in:

$$x_1 = 2.83, x_2 = 2, z = 4.83$$

quindi abbiamo upper-bound 4

LA variabile x_1 ha valore frazionario compreso tra 2 e 3. Effettuiamo un branch con vincoli $x_1 \leq 2$ e $x_1 \geq 3$:

VII. CONSTRUCTIVE HEURISTICS

A. Introduzione

Nel (caso peggiore) di variabili binarie il livello computazionale sarà dato da $O(2^n)$ quindi il tempo di calcolo cresce esponenzialmente con il numero di variabili.

In molte applicazioni si fa affidamento ad algoritmi detti **inesatti** che **CERCANO** di generare delle soluzioni ammissibili. Questi rientrano negli **algoritmi euristici** dove è importante che la **conoscenza del progettista venga trasmessa all'algoritmo**. Questi algoritmi possono essere:

- **costruttivi**: dove cercano di generare una prima soluzione ammissibile
- **migliorativi**: dove prova a migliorare la prima soluzione

Trovare una soluzione ammissibile potrebbe essere difficile dato che potrebbe essere più complicato del trovarne una ottima.

B. Travelling Salesman Problem (TSP)

Data una matrice C di transizione dove, per andare dal punto 1 al 2 **pago c_{12}** e così via per tutte le possibili iterazioni **per un numero n di punti da "visitare"**. Lo scopo sarebbe trovare un circuito che tocchi tutti i punti una sola volta con **costo minimo**.

Avremo che il **tempo di ciclo determina la produttività della macchina** tipo un robot che fa n fori e conclusi gli n fori finisce il cilo.

Un vincolo ulteriore potrebbero essere le **finestre temporali** che potrebbero esserci come nei casi di consegna dei pacchi amazon, il che **rende computazionalmente più difficile o anche inammissibile il problema**. Notare come cambiando anche solo un dato potremmo avere una soluzione ammissibile o una crescita esponenziale dell'infattibilità dalla quale deriva l'instabilità.

C. Algoritmo Greedy

Algoritmo che **cerca di massimizzare nel breve periodo** (usato per essere adattato a qualsiasi problema). È un **euristica di tipo costruttivo**, infatti ha una **procedura sequenziale** che costruisce la soluzione passo passo **massimizzando solo l'utilizzo immediato** andando però in contro a:

- una soluzione inammissibile
- non garantire una soluzione ottima

il suo **pseudocodice** è un adattamento sull'algoritmo del Travelling Salesman Problem (TSP):

last = 1 (last = ultimo punto toccato)

$S = \{2, 3, \dots, n\}$

while ($S \neq$ insieme vuoto)

 estrarre da S un punto

$i = \operatorname{argmin}_{i \in S} c_{\text{last}, i}$

 succ_{last} = i

 last = i

 succ_{last} = 1

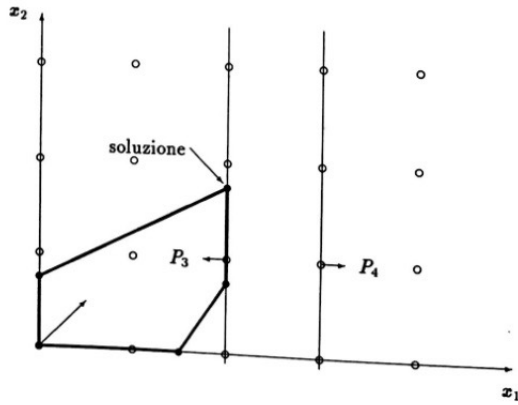


Figura 41. Regione ammissibile

dove:

- P_4 non ha intersezione quindi sarà soluzione inammissibile
- P_3 con soluzione ottima in:

$$x_1 = 2, x_2 = 1.75, z = 3.75$$

quindi abbiamo upper-bound 3

La variabile x_2 ha valore frazionario compreso tra 1 e 2. Effettuiamo un branch con vincoli $x_2 \leq 1$ e $x_2 \geq 2$:

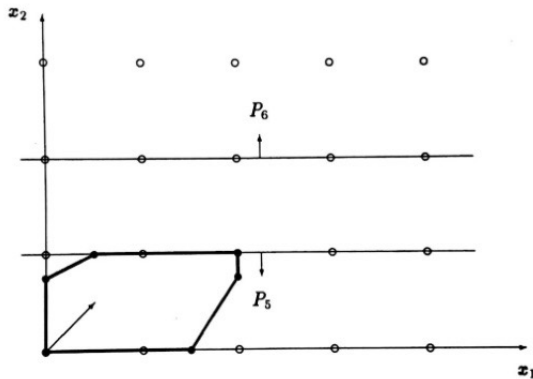


Figura 42. Regione ammissibile

dove:

- P_6 sarà inammissibile
- P_5 con soluzione ottima in:

$$x_1 = 1, x_2 = 2, z = 3$$

La soluzione ottima è intera e quindi il branch and bound si interrompe.

Vediamo un esempio con dati: $n = 4$ e

$$C = \begin{bmatrix} 0 & 10 & 5 & 8 \\ 10 & 0 & 2 & 1 \\ 5 & 2 & 0 & 4 \\ 8 & 1 & 4 & 0 \end{bmatrix}$$

dove applicando l'algoritmo abbiamo:

- last = 1; $\rho = < 2, 3, 4 >$
- last = 3; $\rho = < 2, 4 >$; succ₁ = 3
- last = 2; $\rho = < 4 >$; succ₃ = 2
- last = 4; $\rho = < >$; succ₂ = 4
- succ₄ = 1

D. Miller-Tucker-Zemlin

Questo modello definisce delle **variabili binare x_{ij}** per ogni coppia di punti i e j :

$$x_{ij} = \begin{cases} 1, & \text{se } i, j \text{ sono/saranno visitati} \\ 0, & \text{altrimenti} \end{cases}$$

avremo allora che:

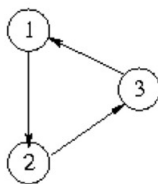
$$\min \sum_{i,j} c_{i,j} x_{i,j}$$

s.t

- $\sum_j x_{ij} = 1 \quad \forall i$
- $\sum_i x_{ji} = 1 \quad \forall i$ — > dice che ogni punto ha un suo successore
- $0 \leq x_{ij} \leq 1 \quad \forall x_{ij}$ integer — > dice ogni punto ha un predecessore

Questi **vincoli non bastano** per poter risolvere il modello, infatti non possiamo escludere di avere una **soluzione disconnessa (sub-tour)**

Subtour length = 3



Subtour length = 2



Figura 43. Soluzione disconnessa

Si introducono allora dei **vincoli di connessione** della soluzione tramite una **variabile per l'ordine di visita: u_i** (una per ogni punto).

Poniamo allora un punto iniziale con $u_i = 1$:

$$\begin{aligned} u_1 &= 1 \rightarrow x_{1,3} = 1 \\ u_3 &= 2 \rightarrow x_{3,4} = 1 \\ u_4 &= 3 \rightarrow x_{4,2} = 1 \\ u_2 &= 4 \rightarrow x_{2,1} = 1 \end{aligned}$$

altri **vincoli da imporre su u_i** :

- $2 \leq u_i \leq 1 \quad \forall i \neq 1$
- $u_i - u_j + 1 \leq n(1 - x_{ij}) \quad \forall i, j \neq 1$

Avremo quindi **2 casi**:

- $x_{ij} = 0 \Rightarrow u_i - u_j + 1 \leq n$: ovvio dato che tutte le variabili u **sono comprese in n**
- $x_{ij} = 1 \Rightarrow u_i - u_j + 1 \leq 0$: dato che $u_j \geq u_i + 1$ allora i **predecessore di j**

E. Algoritmo relax-and-fix

Si usa per **problemi multiperiodali** dato che voglio poter avere un approccio con **suddivisione in più periodi temporali**:

$$\min z = c^T x$$

s.a.

- $Ax = b$
- $x \geq 0$ (integer)

Partiziono le variabili in n gruppi in modo da avere variabili impiegate per un solo intervallo di tempo. Allora il problema diventa:

abbiamo allora delle applicazioni in cui le variabili naturalmente si dividono in gruppi dato un sviluppo temporale, il problema allora diventa:

$$\min z = \sum_{i=1}^n c_i^T x_i$$

s.a.

- $\sum_{i=1}^n a_i x_i = b$
- $x_i \geq 0$ (integer)

la difficoltà del problema si trova nelle molte variabili intere. Posso allora **definire un problema con variabili x_1 del primo gruppo intere**, per le restanti faccio un rilassamento. Potremo avere che:

- ausiliario: inammissibile \rightarrow iniziale: inammissibile
- ausiliario: sol. ottima \rightarrow fisso x_1 alla soluzione: $x_1 = \bar{x}_1$

Quindi per **risolvere il problema** dobbiamo:

1) x_1 diventa **intera**

- **rilasso** le altre variabili
- **risolvo** il problema e **fissiamo** $x_1 = \bar{x}_1$ che viene "rimossa" dal modello

2) ecc, ecc...

Così facendo **tengo conto delle variabili "future"** senza richiede un grande sforzo computazionale, dato che lavora sul singolo gruppo.

Generalizzando:

$$\min x = \sum_{i=1}^{k-1} c_i \bar{x}_i + c_k x_k + \sum_{i=k+1}^n c_i x_i + i$$

s.t

- $\sum_{i=1}^{k-1} a_i \bar{x}_i + a_k x_k + \sum_{i=k+1}^n a_i x_i = b$
- $x_k \geq 0$ (integer)
- $x_i \geq 0$

Lo **pseudocodice** sarà:

```

found = true
for (k = 1  $\rightarrow$  n):
  
```

solve $P_k(\bar{x}_1, \dots, \bar{x}_{k-1})$

if è inammissibile:

found = false

break;

else: (x'_{k+1}, \dots, x'_n) è soluzione fissare $\bar{x}_k = x'_k$

F. Algoritmo rolling horizon

Durante il primo periodo considero di:

- 1) creare un sottoproblema con alcune variabili da x_1 a x_k
- 2) trascurare le variabili dei periodi successivi
- 3) fisso x_1
- 4) mi muovo di uno step
- 5) considero un sottoproblema con x_1
- 6) fisso x_2 e considero le variabili fino a $x_k + 1$.
- 7) ecc, ecc...

Andiamo quindi a risolvere un problema a k variabili ma senza tenere in conto le variabili troppo successive, quindi sarà meno accurato.

G. Multiple criteria decision making

Spesso abbiamo piu' obbiettivi e non riusciamo a scegliere in anticipo quale ha prioritá allora spesso vanno in conflitto, allora cerchiamo di trovare le migliori soluzioni che devono rispettare alcuni vincoli:

$$p \text{ criteria} = \begin{cases} z_1 = f_1(x) \text{ da min o max} \\ z_2 = f_2(x) \text{ da min o max} \\ \dots \\ z_p = f_p(x) \text{ da min o max} \end{cases}$$

s.t.

- $g(x) \geq 0$
- $x \geq 0$

H. Esempio problema scheduling

Preso un problema di scheduling, abbiamo n attività con paramentri:

- S_i : starting time dell'attività i
- d_i : durata dell'attività i
- $p_{ij} = 1$ se l'attività j è prerequisito dell'attività i
- $p_{ij} = 0$ altrimenti

come f.o: $\min T$ con T tempo di completamento:

$$\min T$$

s.t.

- $T \geq S_i + d_i, i = 1, \dots, n$
- $S_i \geq p_{ij}(S_j + d_j), i, j = 1, \dots, n$

Per poter formulare un'attività in modo fattibile:

- D_i : max durata dell'attività i
- b_i : min durata dell'attività i
- d_i : durata dell'attività i
- X_i : costo per accelerare il completamento dell'attività i allora:

- $d_i = D_i - a_i X_i$ con a_i coefficiente in unità di tempo
- $d_i \geq b_i$

abbiamo da minimizzare 2 obiettivi:

$$\min \sum_{i=1}^n X_i$$

$$\min T$$

s.t.

- $T \geq S_i + d_i \quad \forall i$
- $S_i \geq p_{ij}(S_j + d_j) \quad \forall i, j, i \neq j$
- $d_i \geq b_i \quad \forall i$
- $d_i = D_i - a_i X_i \quad \forall i$

I. Superior solutions

Preson un problema con piu' obbiettivi allora ne consideriamo uno con obiettivo singolo, allora rappresento i singoli obiettivi nello spazio.

Preso una soluzione potremo trovarne una migliore ma solo se si trova nella zona ammissibile e che sia nella frontiera, in caso contrario è detta utopia.

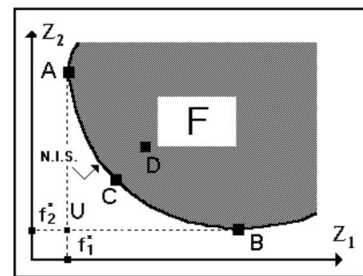


Figura 44. Utopia

Definiamo invece soluzione superiore la soluzione che minimizza o massimizza meglio di tutte (come U se fosse nella zona ammissibile).

Si parla allora di soluzioni dominate e soluzioni non dominate (di Pareto).

Per confrontare due soluzioni di Pareto usiamo il trade-off ratio tamite:

$$\left| \frac{z_i(x_A) - z_i(x_B)}{z_j(x_A) - z_j(x_B)} \right|$$

che rappresenta il miglioramento dell'obiettivo i-esimo in base al miglioramento di un altro obiettivo.

Nel caso di una zona convessa collegheremo i punti limite tramite una tangente immaginaria, dalla quale però non prendere mo i punti di frontiera.

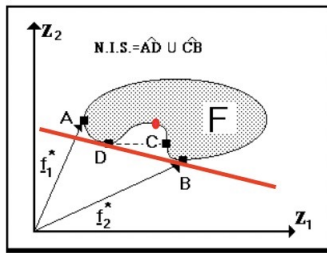


Figura 45. Zona convessa

per trovare le soluzioni efficienti abbiamo 2 metodi:

- 1) **metodo dei vincoli**: considero un solo obiettivo, per gli altri avrò:

$$\min f_i(x)$$

s.t

- $f_k(x) \leq u_k$
- $g(x) \geq 0$
- $x \geq 0$

Prendo allora solo A e D:

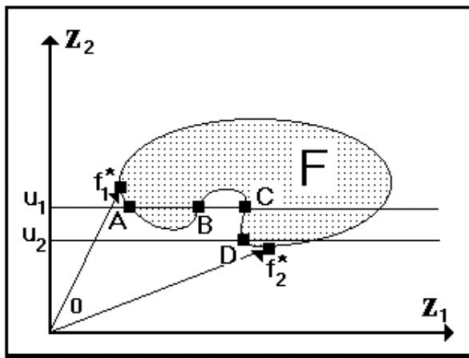


Figura 46. Metodo dei vincoli

- 2) **metodo dei pesi**: abbiamo una sola f.o somma di tutti li obiettivi:

$$\min \sum_{i=1}^p w_i f_i(x)$$

s.t

- $\sum_{i=1}^p w_i = 1$
- $g(x) \geq 0$
- $x \geq 0$
- $w_i \geq 0$

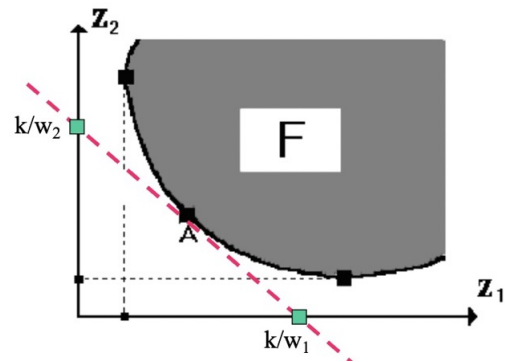


Figura 47. Metodo dei pesi

dove a seconda del valore che diamo ai pesi andiamo a spostare la retta tangente alla zona ammissibile.

J. Metodo a priori

Per risolvere il singolo problema usiamo:

$$\max U(f_1(x), f_2(x), \dots, f_p(x))$$

s.t

- $g(x) \geq 0$
- $x \geq 0$

ma in genere non si trova questa soluzione.

K. Metodo a posteriori

Generiamo tutte le soluzioni efficienti per dire quale e' la migliore. Notare che così facendo si potrebbe avere una crescita esponenziale del problema.

Una buona alternativa è il metodo interattivo.

L. Metodo interattivo

Presi 2 obiettivi:

- troviamo il punto ottimo per entrambi
- troviamo la prima soluzione efficiente che sia tra A e C
- scegliere su quale parte della frontiera posizionarsi
- ecc, ecc...

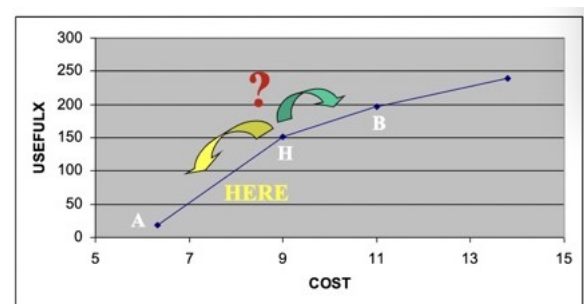


Figura 48. Metodo interattivo

mi fermerò quando il segmento che rimane è così piccolo che i due punti coincidono.