Pianificazione Automatica e sistemi di Supporto delle Decisioni

Matteo Aprile
Professore: Giampaolo Ghiani, Enamuele Manni

INDICE

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)

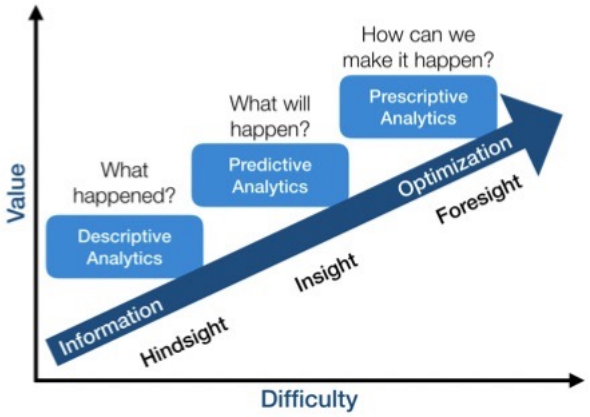

Figura 1. Fasi della business analytics

B. *Decisioni*

Rappresenta la scelta di un elemento tra piu' soluzioni dopo aver ponderato le opzioni.

Possiamo avere più casi d'uso:

- simplest case: abbiamo **poche alternative** quindi una semplice scelta
- multple 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

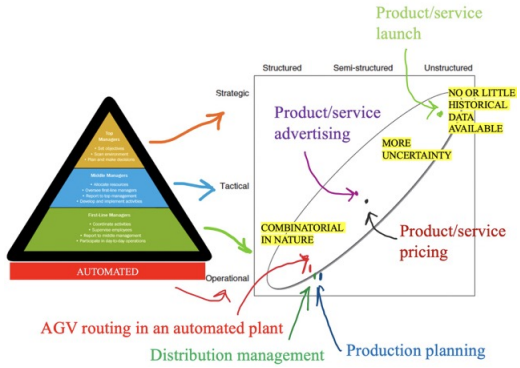

Figura 2. Diagonale decisionale

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 "Inteligence" 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 managment" 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)

Attivita 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**

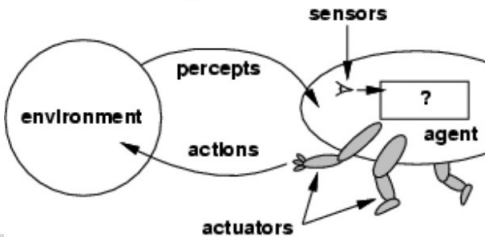

Figura 3. Schematizzazione di un agente e sue caratteristiche

H. Artificial Intelligence (AI)

Comprende tante sottodiscipline:

- automated reasoning: legato alla **rappresentazione del mondo** e **come raggionare 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
- manipuliation: 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 Leaning

Si basa sull'apprendimento automatico con reti neurale 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 regolarita' 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 **??**.

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) **descrivo 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

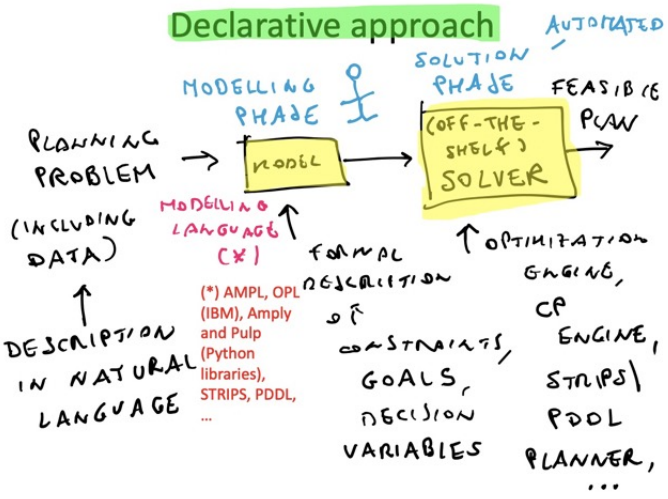

Figura 4. procedura implementata

Il che è utile dato che per agire su un problema bastera' 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**

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 semistrutturate

Vogliamo solo valutare le prestazione 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 Modellling System, per simulare la rete per la quale passano le informazioni e i server ognuno con diverse proprietà di ciascun linker.

III. Introduzione all'ottimizzazione matematica

A. Introduzione

Partiamo da un insieme di formule ed equazioni che modelleranno il problema. Con questo modello proviamo a trovare una soluzione al nostro problema attraverso algoritmi o risolutori. 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 obietivo: 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 matematici 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:
   $$4x_A + 2x_B \leq 400$$
   - 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: $x_A = 40$ $2 * 40 + 4X_B = 400$ 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 ugualianza
- vincoli >=
- 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

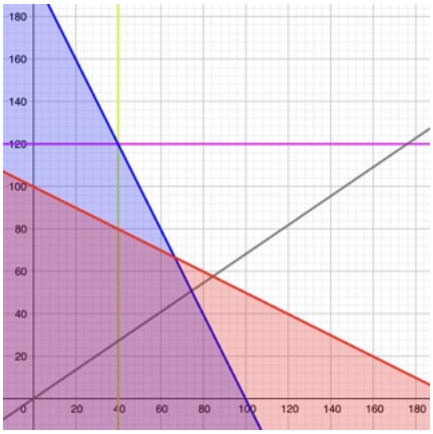

Figura 6. Rappresentazione grafica esempio

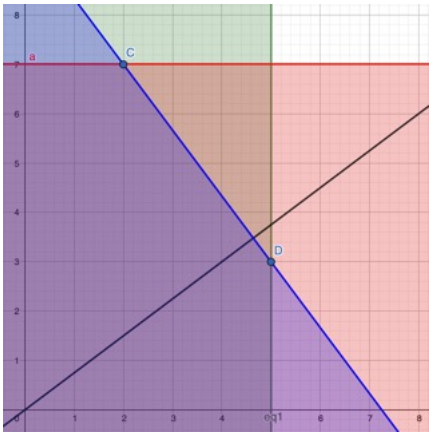

Figura 7. Rappresentazione grafica esempio 1

H. Implementazione in Python

```
import pulp as p

# 1. creazione del modello
model = p.LpProblem("ProductMix", p.LpMaximize)

# 2. definisco le variabili decisionall
x_A = p.LpVariable("x_A", cat="Continuous", lowBound
    =0)
x_B = p.LpVariable("x_B", cat="LpContinuous",
    lowBound=0)

# 3. definisco la funzione obiettivo in funzione
    delle variabili decisionali
model += 15 * x_A + 10 * x_B

# 4. definire i vincoli
model += 4 * x_A + 2 * x_B \leq 400
model += 2 * x_A + 4 * x_B \leq 400
model += x_A \leq 40
model += x_B \leq 120

# 5. ricolvere il problema
model.solve()

# print della soluzione
print("next week produce {} pallets of A".format(x_A
    .varValue))
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:
   - $x_1 \leq 5$
   - $x_2 \leq 7$
   - $4x_1 + 3x_2 \leq 29$
   - $x_1, x_2 \geq 0$

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

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

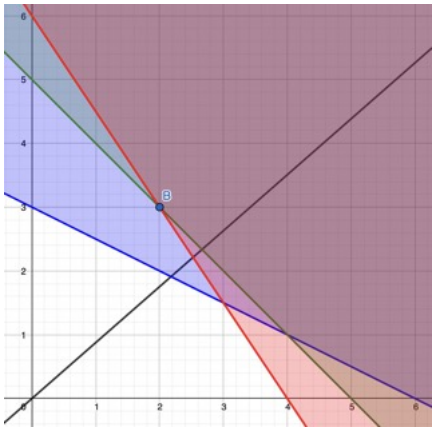

Figura 8. Rappresentazione grafica esempio 2

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

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

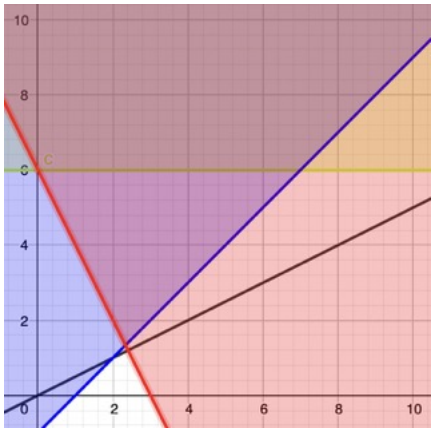

Figura 9. Rappresentazione grafica esempio 3

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

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

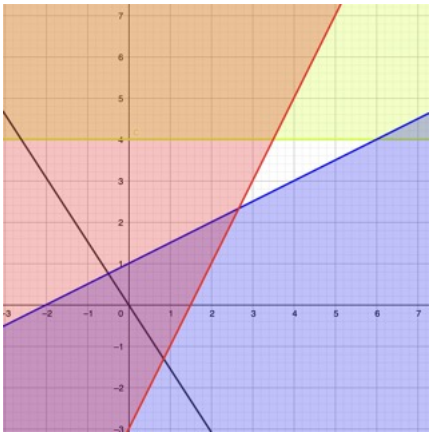

Figura 10. Rappresentazione grafica esempio 4

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

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

5) L'azienda vuole decide 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:

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

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

dove definiamo la possibilita' di vincoli di $\geq$, = e variabili ≥ 0.

La funzione obiettivo è definita da:

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

che rappresenta il problema espresso in forma generale.

B. Problema in FORMA CANONICA

Se in (*) abbiamo:

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

allora il problema si dice in forma canonica:

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

C. Problema in FORMA STANDARD

Se in (*) abbiamo:

- $A = 0$
- $J = 1, 2, .., n$

allora il problema è:

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

Possiamo sempre ricondurci tramite trasformazioni alla forma standard.

D. Terminologia

Nella forma standard abbiamo che:

- z: la **funzione obbiettivo** per la quale trovare il valore minimo
- A: **matrice dei vincoli**
- D: **matrice dei coefficienti**, matrice di dimensione m x 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:

$$x_j t.c. j \notin J$$

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

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

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

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 $x_j = x_j^+ - x_0$. Cosi' 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 si 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:

- $8x_1 + 6x_2^+ - 6x_2^- - x_3 = 48$
- $5x_1 + 10_2^+ - 10x_2^- - x_4 = 50$
- $13x_1 + 10_2^+ - 10x_2^- + x_5 = 130$
- $x_1, _2^+, x_2^-, x_3, x_4, x_5 \geq 0$

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 attivita' 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)):

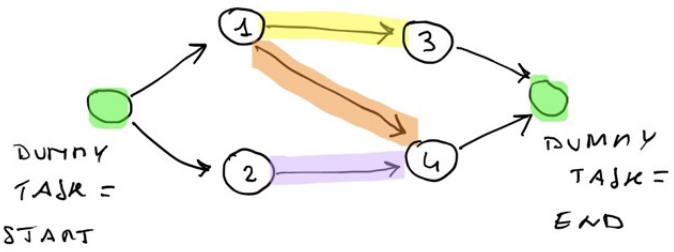

Figura 11. Diagramma Activity On Node

dove abbiamo degli "archi" che rappresentano le predecessioni. Avremo anche dei vertici fittizzi:

- 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 $z = 1T + 0s_1 + 0s_2 + ... + 0s_n$
- relazioni di precedenza:
  relazioni che portano alcuni nodi a dipendere da altri:

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

  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 $s_i + d_i$
  - $p_i j = 0$: i non è predecessore quindi avremo $0 \leq s_j$ allora il vincolo è ridondante

  scriviamo allora:

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

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 $d_i = d_i^N$.

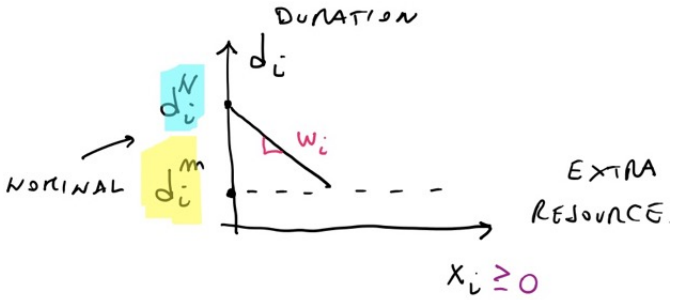

Figura 12. Diagramma budget

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

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:

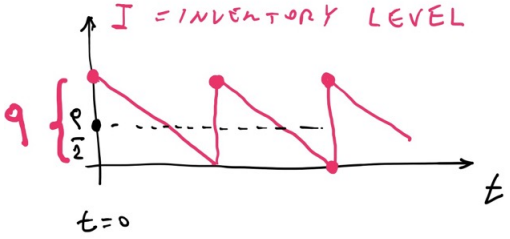

Figura 14. Livello di inventario

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

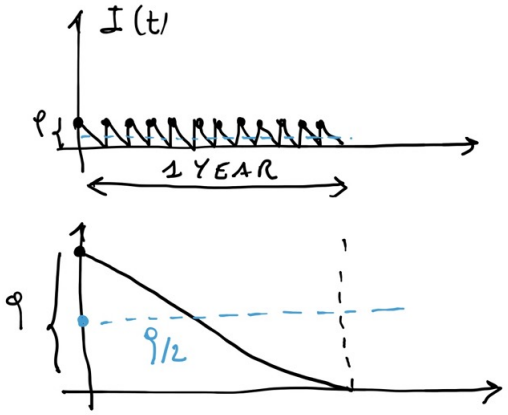

Figura 15. Casi particolari

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 annuale 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:

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

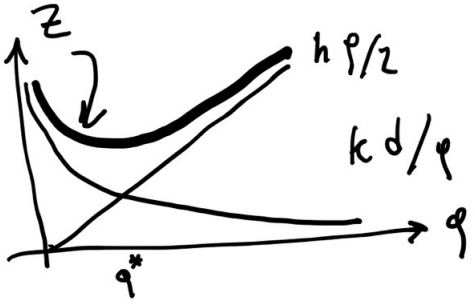

Figura 16. Caso di lotto economico

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

- spazio limite di un magazzino
- produzione di più prodotti contemporaneamente
- vincolo di immobilizio
- 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 annuale z sarà data dal costo annuale 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 capacita' del magazzino e' minore del lotto economico, la soluzione ottima sara' la nostra capacita' 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:
$q_A + q_B \leq Q$ $c_A q_A + c_B q_B \leq C$

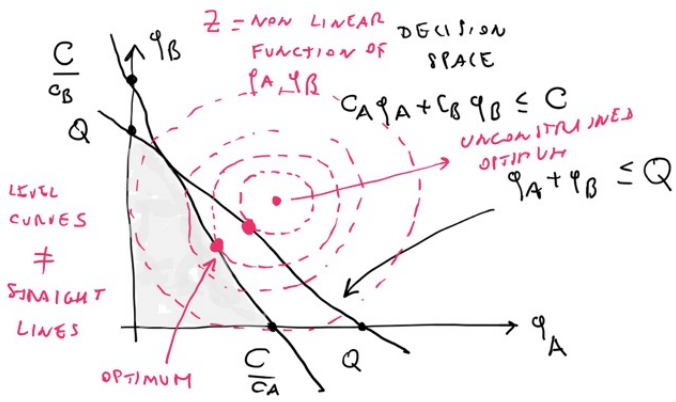

Figura 17. Grafico con linee di livello non lineari

La funzione obiettivo non e' 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 ammissibilita'. 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:

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

con n righe e m colonne delle matrice $\underline{\underline{A}}$.
Utilizziamo l'operazione di Pivot, quindi il vincolo:

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

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 si 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:

$$a_{rj} = \frac{a_{rj}}{ars} \quad \forall \quad j = 1, ..., n+1$$

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

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:

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

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:

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

I. Tableau

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

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

dove m righe e n colonne.
I nostri problemi hanno variabili continue dove usando l'algoritmo del simplesso avremo come forma generale:

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

per i vincoli invece:

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

Assumiamo che:

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

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 e' singolare posso ricavare una soluzione imponendo

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

Per le variabili B avremo:

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

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.

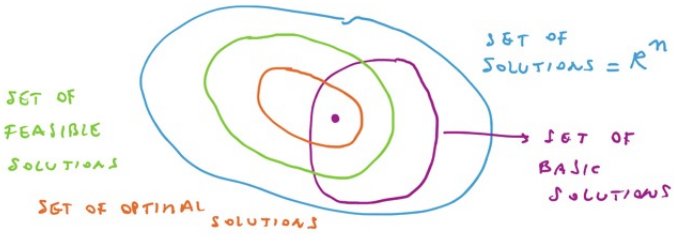

Figura 18. Insiemi delle soluzioni

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 e' 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 perche' $x_4 \leq 0$ 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:

$$I_B =< 1, 3 >$$

$$I_N =< 0, 2 >$$

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

$$\beta(0) = 3$$

$$\beta(1) = 1$$

Una forma canonica mi da 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:

$$\overline{z} = 63.3$$

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

L. *Generalizzazione forma canonica*

Generalizzando avrò come funzione obiettivo:

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

e con vincoli:

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

con ovviamente $x_j \geq 0, j \in J_N \cup J_B$
L'equazione di base associata sarà:

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

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

Avremo la forma generale quando è $\leq$ e $\underline{b} \geq 0$.
Per il test di ottimalita' in forma canonica:

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

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

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

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

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

$$z = \sum_{j \in J_N} \overline{c}_j ...(slide)$$

cioè $x_j = 0 \rightarrow 1$

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

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

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

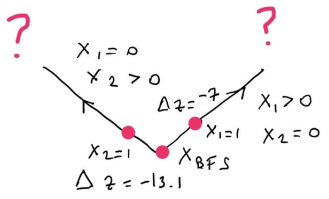

Figura 19. Soluzioni per ottimalità

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.

Sceglieremo come variabile, detta variabile entrante, quella con coefficiente di costo piu' negativo:

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

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

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

Applichiamo un vincolo sul massimo valore che x_s puo' raggiungere:

$$x_s = 0 \rightarrow > 0$$

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

allora avremo:

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

con vincoli:

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

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

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

A questo punto avremo 2 casi possibili:

1) avremo:
$$\overline{a}_{is} \leq 0 \quad \forall \quad i = 1, ..., m$$
allora:
$$x_{\beta(i)} \geq 0 \quad \forall \quad x_s \rightarrow +\infty$$
con: $z \rightarrow -\infty$
2) avremo:
$$\exists\ i : \overline{a}_{is} > 0$$
per calcolare il massimo valore che x_s puo' assumere è dato da:

$$-\overline{a}_{is} x_s \geq -\overline{b}_i$$

$$\Rightarrow x_s \leq \frac{\overline{b}_i}{\overline{a}_{is}} \quad \forall \quad i = 1, ..., m;\ \overline{a}_{is} > 0$$

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

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

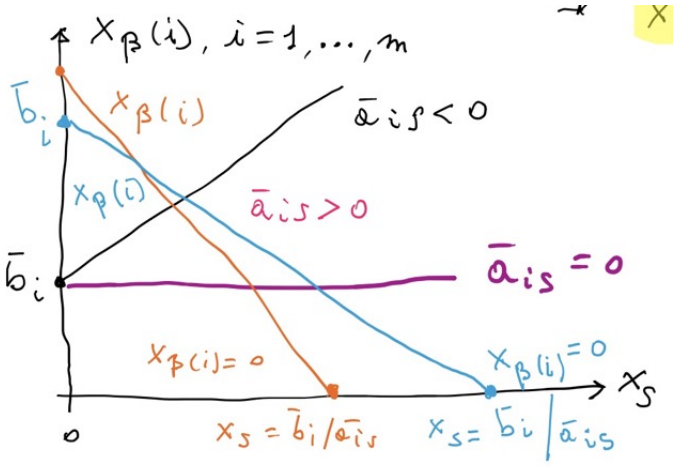

Figura 20. Possibilità con i valori di $\overline{a}_{is}$

Per $\overline{a}_{is} > 0$ avremo che ad un certo punto x_s arrivera' 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:

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

annullando x_2, x_3, x_4:

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

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:

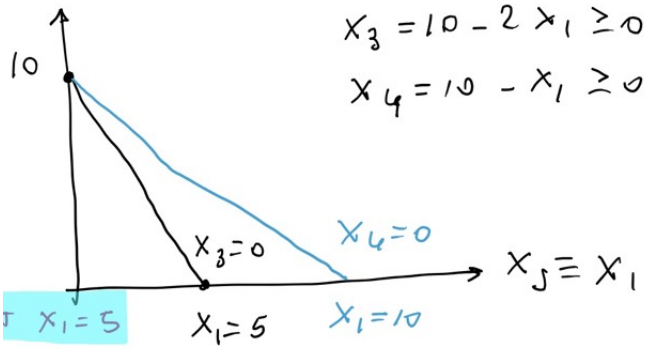

Figura 21. Permutazione di x_1

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:

$$x_r = \text{argmin}(\frac{10}{2}, \frac{10}{1}) = x_3$$

Pseudocodice dell'algoritmo del simplesso:

trovo la prima BFS
while $\exists \bar{c}_j < 0 \quad \forall \quad j \in I_N$:
  trovo argmin per "c" del Pivot: $x_s = \text{argmin}_{j \in I_N} \bar{c}_j$
  if $\bar{a}_{is} \leq 0$:
    abbiamo un problema unbounded
  trovo argmin per "r" del Pivot: $x_s = \text{argmin}_{i=1,\ldots,m} \frac{\bar{b}_i}{\bar{a}_{is}}$
  eseguo il Pivot
se trovo una soluzione di base ammissibile ottima faccio un analisi per capire se e' 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:

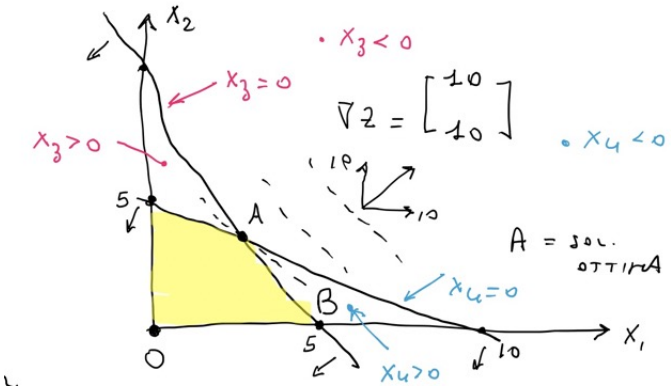

Figura 22. Graficazione dei vincoli

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:

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

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

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:

- $J_B =< X_3, X_4 >$
- $J_N =< X_1, X_2 >$

BFS:

- $x_1 = x_2 = 0$

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

Il test di ottimalita' fallisce e quindi possiamo avere una soluzione migliore. Applico l'euristica del valore di costo ridotto. Sceglo 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:

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

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:

$$J_B =< x_1, x_4 >$$

Dobbiamo allora effettuare un'operazione di Pivot:

- divido per 2 la riga 1:

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

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

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

- moltiplico 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:

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

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

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

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

$$J_B =< x_1, x_2 >$$

dato che abbiamo x_2 entrante e x_4 uscente.

Il perno del nuovo Pivot sarà 1.5:

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

avremo allora:

- $x_3 = x_4 = 0$
- $x_1 = 3.\overline{3}$
- $x_2 = 3.\overline{3}$
- $\overline{z} = -66.\overline{6}$

dove abbiamo ottenuto una soluzione ottima.

N. Soluzioni ottime multiple

Per individuarle ricordiamo che esiste una soluzione ottima se:

$$\overline{c_j} \geq 0$$

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

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

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

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

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:

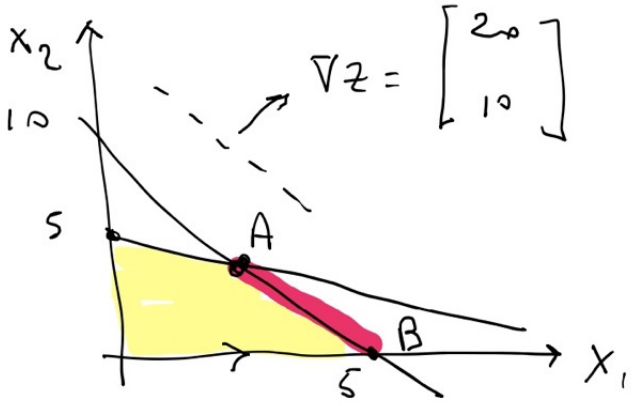

Figura 23. Graficazione dei vincoli

tableau:

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

BFS:

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

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:

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

Il test di ottimalita' sulle variabili non di base $J_N =< x_2, x_3 >$ è OK, ma non e' 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:

$$\overline{c}_{j*} x_{j*} \quad \forall \quad \overline{c}_{j*} = 0, \; x_{j*} > 0$$

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

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

BFS:

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

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

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

P. Algoritmo della convergenza

Abbiamo che se c_s e' la varibiale entrante allora il costo è dato dalla relazione:

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

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

L'algoritmo del simplesso si comporta in modo decrescete 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 $\overline{b}_i = 0$ per $\overline{a}_{is} > 0$.

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

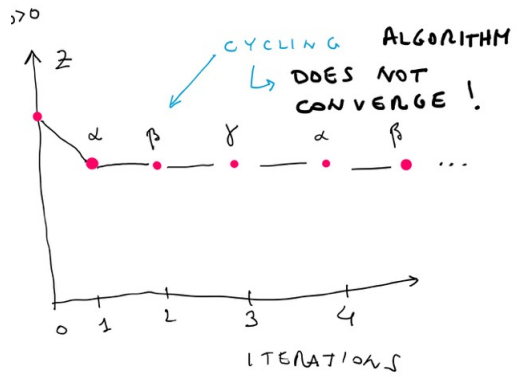

Figura 24. Cycling algorithm result

Potrebbe anche capitare di avere un miglioramento dopo n iterazioni.

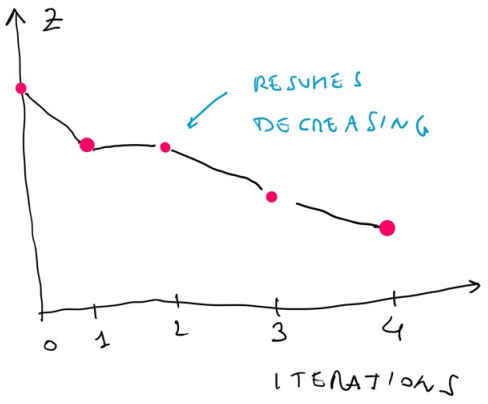

Figura 25. Not cycling algorithm result

Q. Esempio BFS degenere

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:

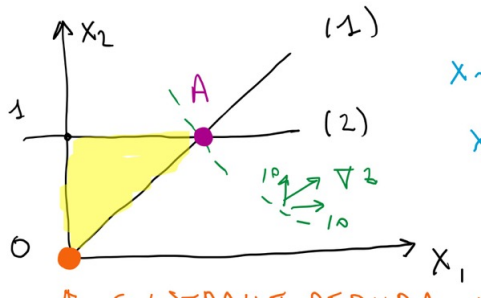

Figura 26. Grafico di un sistema inizialmente degenere

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 degenere. 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:

$$x_r = \text{argmin}(\frac{0}{1}) = x_3$$

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

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 degenere quindi ci troviamo su un plateau. Graficamente abbiamo che x_1 rimane 0 nel punto $(0, 0)$, avremo allora che nel segmento $\overline{0A}$ 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:

$$x_r = \text{argmin}(\frac{1}{1}) = x_4$$

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

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 parita' di valore, a prendere quello con indice piu' negativo. Quindi in:

S. Variabili artificiali

Iniziamo aggiungendo ad ogni vicolo una variabile artificiale:

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

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

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

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

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

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

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 quivalente ma che sia valida sia per P_a che per P.

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

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

vincoli:

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

Notare che ammette sempre soluzioni ottime.

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

Dati quesi 2 problemi avremo che per:

1) $\rho^* > 0$: il problema P è **inammissibile**
2) $\rho^* = 0$: alla soluzione ottima P_a **corrispone 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:

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

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

con $x_j \geq 0 \quad \forall \quad j = 1, ..., m.$

Forma standard abbiamo che la funzione obiettivo:

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

vincoli:

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

con

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

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

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

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

$$h = \text{argmax}_i\{b_i, \ i = m_1+1, ..., m_2\}$$

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

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

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:

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 6 \\ 0 & 0 & 0.\overline{3} & 1 & 1 \\ 0 & 1 & 0.\overline{3} & 0 & 5 \\ 0 & 0 & 2.\overline{3} & 0 & -49 \end{bmatrix}$$

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 ($s = 2$):

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

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:

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

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'argoritmo Branch-and-Bound

I problemi che cerchiamo di risolvere sono lineari con variabli 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 verficarne 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 enumeramerazione parsiale dove ci si procura una stima del valore della soluzione ottima rilassando il vincolo di integrita' sulle veriabili intere.

Nella sua risoluzione tramite blackbox potremo avere 4 casistiche:

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

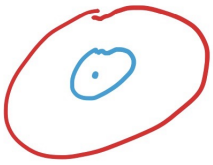

Figura 27. $R(P)$ che contiene P

2) $R(P)$ unbounded $\rightarrow$ P puo' essere unbounded (dove abbiamo variabili intere):

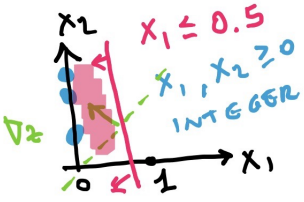

Figura 28. In caso di P unbounded

3) $R(P)$ unbounded $\rightarrow$ P puo' essere inammissibile (dove non abbiamo variabili intere):

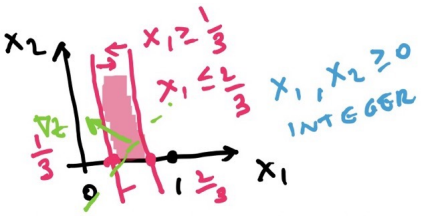

Figura 29. In caso di P inammissibile

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

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

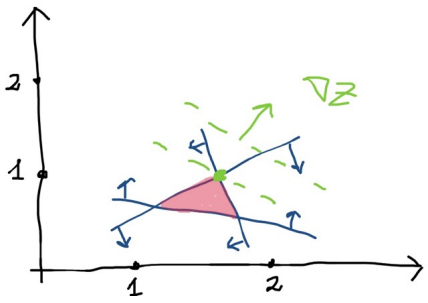

Figura 30. $R(P)$ ottima ma P inammissibile

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

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

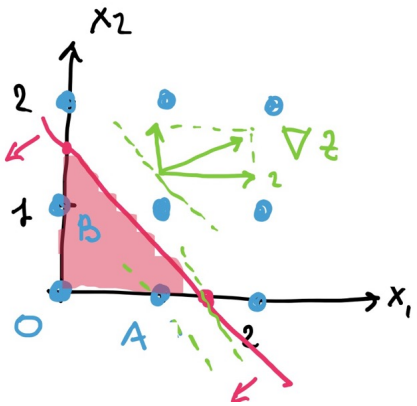

Figura 31. $R(P)$ ottima ma P ammissibile

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, nceve **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:

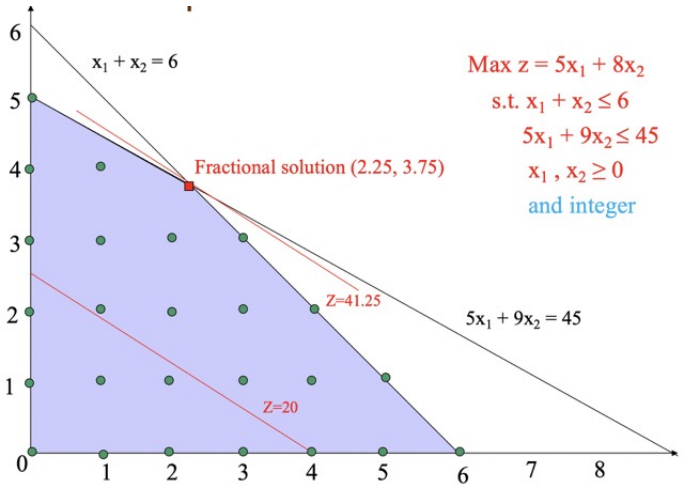

Figura 32. Problema di branching

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

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

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

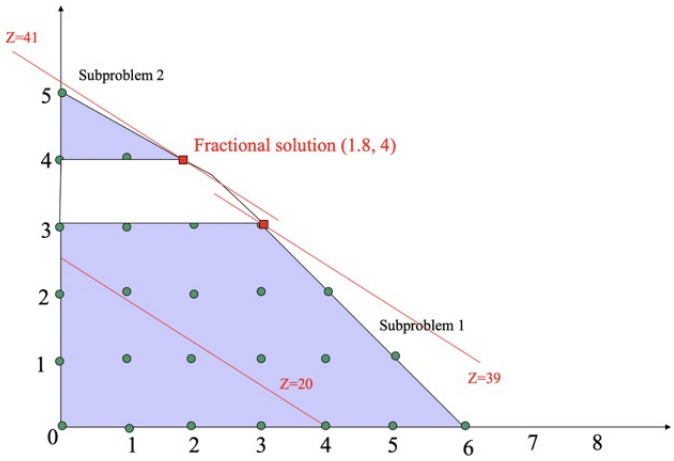

Figura 33. Primo child

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 uppeboound $z = 39$.
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 uppeboound $z = 41$.
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 eplorare 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.

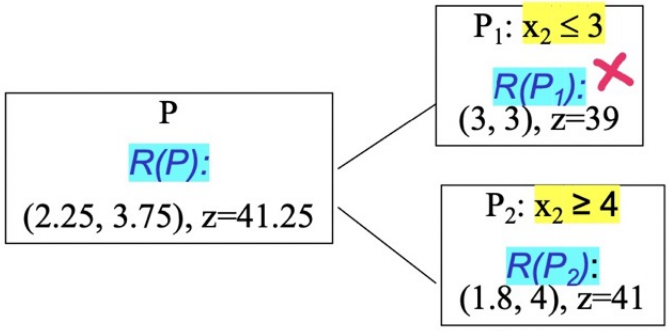

Figura 34. Scarto del primo child

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 $z = 40.55$

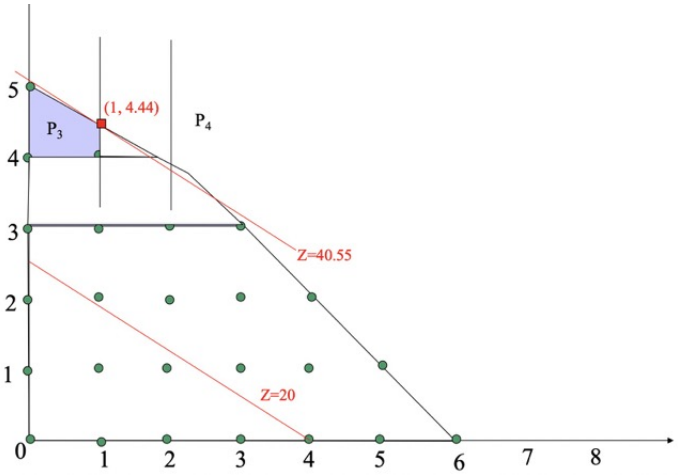

Figura 35. Secondo child

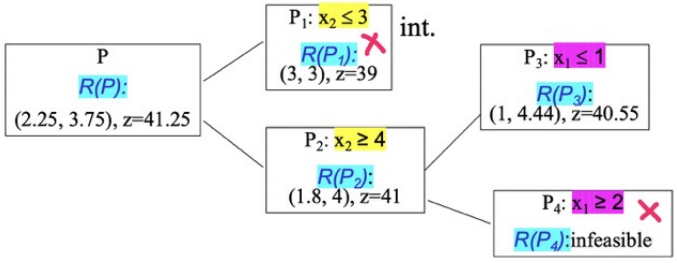

Figura 36. Scarto del secondo child

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:

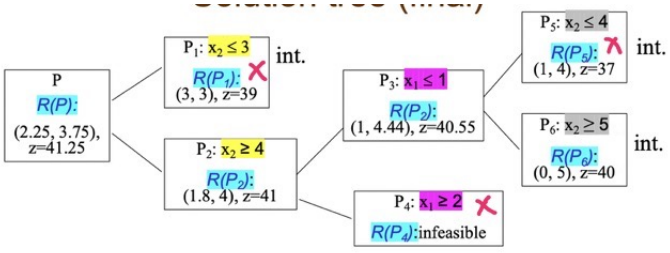

Figura 37. Scarto del terzo child

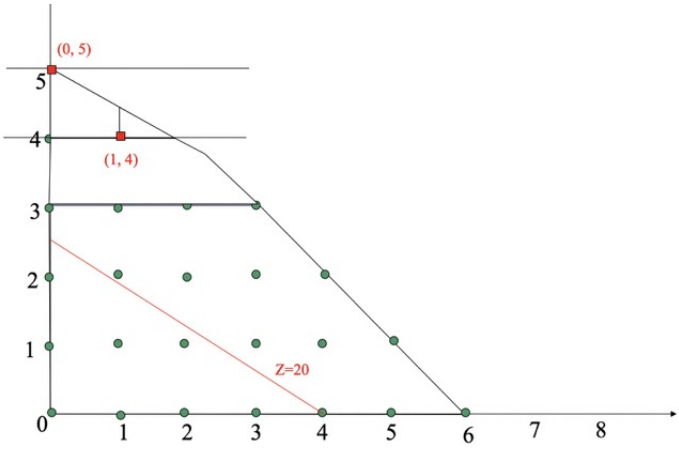

Figura 38. Terzo child

siamo quindi arrivati nella situazione in cui non ci sono piu' 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
$z^{\text{best}} := -\infty$
$x^{\text{best}} :=$ NULL
while L non è vuoto:
  estrarre un sottoproblema k da L
  branching da $1 \to n_k$
  risolvo $R(P)$ e trovo i limiti e l'upperbound UB
  for $i = 1$ to n_k:
    if $UB_i \leq z^{\text{best}}$ then:
      kill child$_i$
    elif $R(P)$ è una soluzione intera:
      $z^{\text{best}} := UB_i$
      $x^{\text{best}} :=$ child$_i$
    elif $R(P)$ è una soluzione frazionaria:
      add child$_i$ to L
  end for
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:

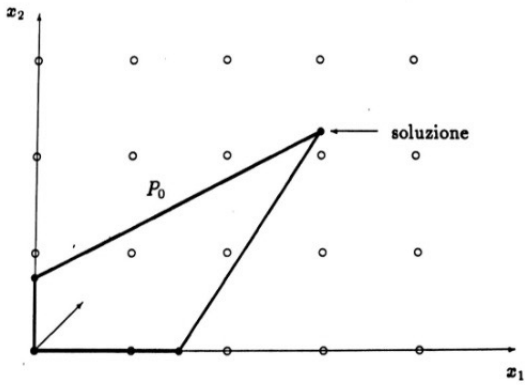

Figura 39. Regione ammissibile

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 $x_2 \geq 3$:

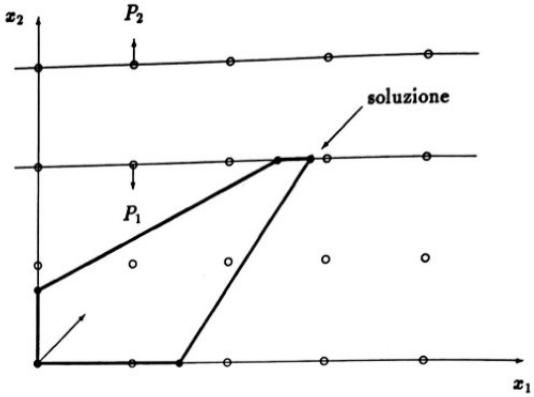

Figura 40. Regione ammissibile

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 $x_1 \geq 3$:

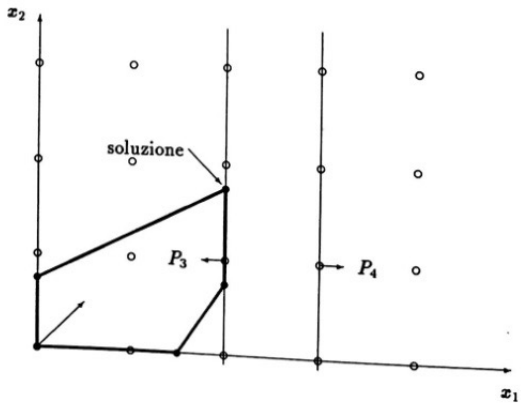

Figura 41. Regione ammissibile

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 $x_2 \geq 2$:

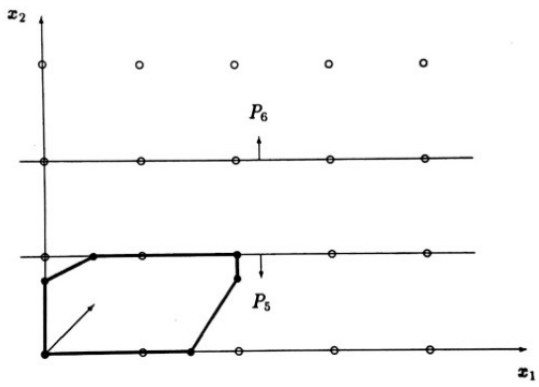

Figura 42. Regione ammissibile

dove:

- P_6 sarà inammissible
- 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.

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 ammisibile
- **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. Travellig 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 possiabili 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 produttivita' della macchina tipo un robot che fa n fori e conclusi gli n fori finisce oil cilo.

Un vincolo ulteriore potrebbero essere le finestre temporali che potrebbero esserci come nei casi di consegna dei pacchi amazon, il che rende computazionalmente piu' difficile o anche inammissibile il problema. Notare come cambiando anche solo un dato potremmo avere una soluzione ammsisibile o una crescita esposnenziale dell'infattibilità dala quele 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 adattamenteo sull'algoritmo del Travellig Salesman Problem (TSP):

last = 1 (last = ultimo punto toccato)
$S = \{2, 3, ..., n\}$
while ($S \neq$ insieme vuoto)
    estrarre da S un punto
        $i = \text{argmin}_{i \in S} c_{\text{last},i}$
        $\text{succ}_{last} = i$
        last $= i$
$\text{succ}_{last} = 1$

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 ; $\rho =< 2, 4 >$; $\text{succ}_1 = 3$
- last = 2; $\rho =< 4 >$; $\text{succ}_3 = 2$
- last = 4; $\rho =<>$; $\text{succ}_2 = 4$
- $\text{succ}_4 = 1$

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 j 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)

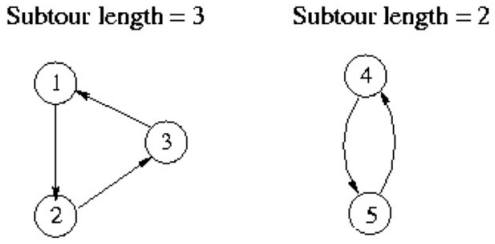

Figura 43. Soluzione disconnessa

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

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

$u_1 = 1 -> x_{1,3} = 1$
$u_3 = 2 -> x_{3,4} = 1$
$u_4 = 3 -> x_{4,2} = 1$
$u_2 = 4 -> x_{2,1} = 1$

altri vincoli da imporre su u_i:

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

Avremo quidni 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 piu' periodi temporali:

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

s.a.

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

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

abbiamo alloradelle applicazioni ndove le varaibili naturalemte si dividono in gruppi datoun osviluppo temporale, il orblema 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 varibili intere. Posso allora definire un problema con variaibli x_1 del primo gurppo intere, per le restanti faccio un rilassamento. Potremo avere che:

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

Quindi per risolvere il problema dobbiamo:

1) $\underline{x}_1$ diventa **intera**
   - **rilasso** le altre variabili
   - **risolvo** il problema e **fissiamo** $x_1 = \overline{x}_1$ che viene "rimossa" dal modello
2) ecc, ecc...

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

Generallizzando:

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

s.t

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

Lo pseudocodice sarà:

found = true
for (k = 1 $\rightarrow$ n):
solve $P_k(\overline{x}_1, ..., \overline{x}_{k-1})$
if è inammissibile:
found = false
break;
else: $(x'_{k+1}, ..., x'_n)$ è soluzione fissare $\overline{x}_k = x'_k$

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 priorita' allora spesso vanno in conflitto, allora cerchiamo di trovare le migliori soluzioni che devono rispettare alcuni vincoli:

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

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'attivtà j è prerequinistito dell'attività i
- $p_{ij} = 0$ altrimenti

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

$$\min T$$

s.t.

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

Per poter formulare un'attivita' 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 accellerare il completamento dell'attività i

allora:

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

abbiamo da minimizzare 2 obiettivi:

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

$$minT$$

s.t.

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

I. Superior solutions

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

Presa 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.

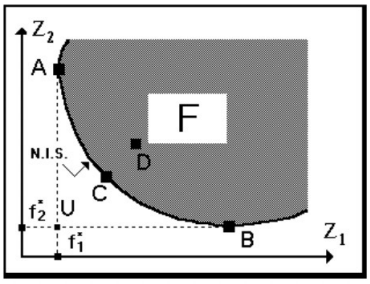

Figura 44. Utopia

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:

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

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

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

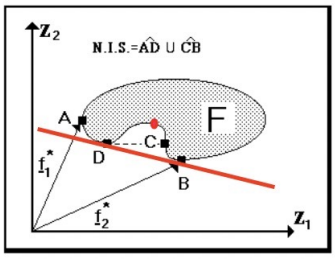

Figura 45. Zona convessa

per trovare le soluzioni efficenti 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:

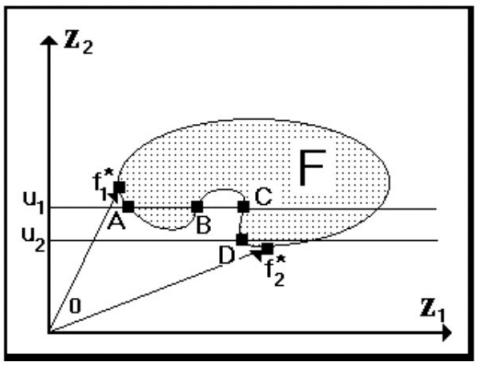

Figura 46. Metodo dei vincoli

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

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

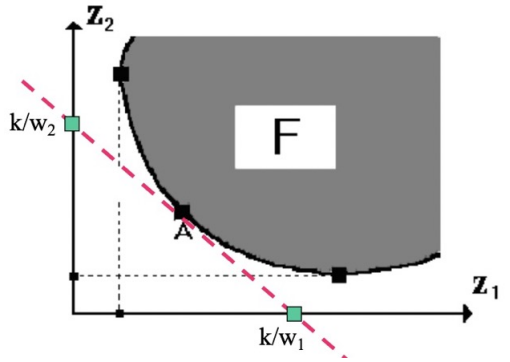

Figura 47. Metodo dei pesi

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

J. Metodo a priori

Per risolvere il singolo problema usiamo:

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

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...

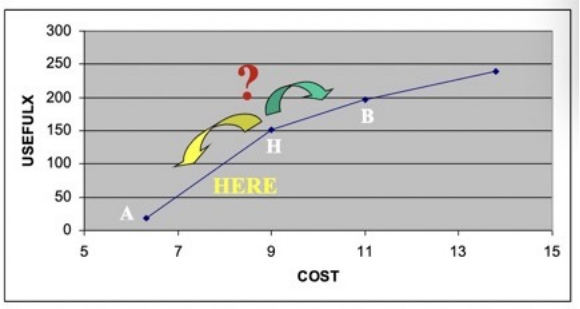

Figura 48. Metodo interattivo

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

VIII. Local Search

A. Introduzione

L'algoritmo di local search è di tipo euristico migliorativo, cioè **presuppone che sia stata già prodotta una soluzione ammissibile**.

Le procedure da utilizzare si dividono in:

1) single population: **migliorare in un intorno (localmente) la situazione**, tenendo in conto che potrebbe rimanere bloccata in minimi locali
2) multiple population: ad ogni iterazione si ha un **insieme di soluzioni**

Non essendo algoritmi precisi vanno adattati al particolare problema che si intende affrontare.

B. Progetto di ricerca locale

Si parte da una soluzione $x^{(0)}$ e si considera il suo intorno $N(x^{(0)})$, cioè un insieme di soluzioni vicine a $x^{(0)}$ dove la definizione di "vicine" viene data dal progettista, aggiungiamo allora il vincolo:

$$\underline{x} \in N(\underline{x}^{(0)})$$

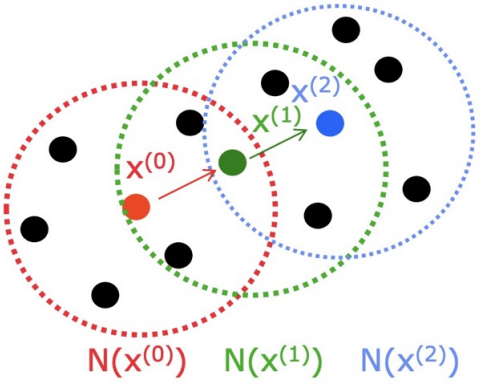

Figura 49. Local search

Avremo allora un problema con f.o:

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

s.t.

- $\underline{\underline{A}}\ \underline{x} \leq \underline{b}$
- $\underline{x} \geq 0$
- $\underline{x}$ integer
- $\underline{x} \in N(\underline{x}^{(0)})$

trovando così una soluzione $\underline{x}^{(1)}$ nell'intorno di $\underline{x}^{(0)}$, ecc ecc...

Sarà un algoritmo di discesa se minimizziamo e di ascesa se massimizziamo, ma entrambi si fermeranno in un ottimo locale.

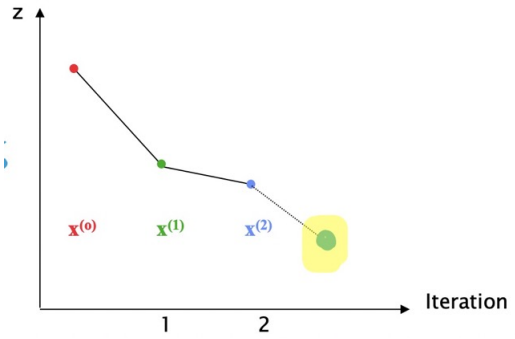

Figura 50. Esempio di ottimo locale

In questo caso un problema:

- concavo: avrà un ottimo locale di **scarsa qualità**
- convesso: avrà un ottimo **uguale a quello globale**

Lo pseudocodice nel caso di minimizzazione è:

INPUT: $\underline{x}^{(0)}$
OUTPUT: $\underline{x}^{(k)}$
$k = 0$
repeat:
  solve (P) + constraint $\forall \quad \underline{x} \in N(\underline{x}^{(k)})$
  let $\underline{x}^{(k+1)}$ be the optimal solution
  $k = k + 1$
until: $z(x^{(k)}) < z(x^{(k-1)})$

Riprendendo il problema del TSP dovremo definire un introno sui punti:

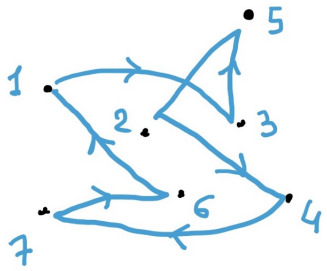

Figura 51. Problema del TSP

Dobbiamo definire una perturbazione della soluzione quindi usiamo un approccio destroy and repair:

1) destroy: **cancellare 2 archi** come x_{13}, x_{25}, creando una soluzione disconnessa con punti $(1, 6, 7, 4, 2)$ e una con $(3, 5)$
2) repair: **creo una giunzione più efficiente** tra i due segmenti con x_{15}, x_{23}

Questo potrebbe portare ad una variazione di costo:

$$\text{var. costo} \;=\; \text{archi aggiunti} \;-\text{archi tolti}$$

$$\Delta z = c_{15} + c_{23} - c_{13} - c_{25}$$

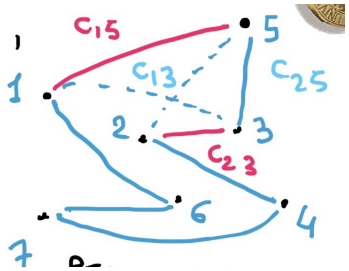

Figura 52. Ottimizzazione del problema TSP

L'intorno sarabbe l'insieme delle soluzioni ottenute dopo il dele and repair. Quanto troviamo $\Delta z = 0$ ci fermiamo.

C. *Local branching*

Se troviamo un problema a grandi dimensioni con variabili binarie:

$$x1, ..., \in < 0, 1 >$$

Essendo un problema che richiederebbe una quantita' elevata di tempodefiniamo un intorno di un soluzione. Volendolo dare in pasto ad un solver dobbiamo prima tradurre il vincolo $N(\underline{x}^{(k)})$ in forma di disequazione lineare, usando la distanza di Hamming (conto i bit diveri tra due stringhe):

$$\underline{x}^{(k)} = (0, 1, 1, 0, 1, 1)$$

$$\underline{x} = (1, 1, 1, 0, 0, 1)$$

con distanza di Hamming: $d(\underline{x}, x^{(j)}) = 2$bits.
Allora data una soluzione $\underline{x}^{(k)}$ avremo che:

$$\underline{x} \in N(\underline{x}^{(k)}) \Leftrightarrow d(\underline{x}, \underline{x}^{(k)}) \leq m$$

allora dovremo delle soluzioni $\leq m$. La distanza di Hamming tra $\underline{x}$ e $\underline{x}^{(k)}$ sarà:

$$\sum_{j=1,\dots,n:x_j^{(k)}=0} x_j + \sum_{j=1,\dots,n:x_j^{(k)}=1} 1 - x_j \leq m$$

o ancora meglio con: $\underline{x}k = (0, 1, 1, 0, 1, 1)$ possiamo contarli con:

$$x_1 + (1 - x_2) + (1 - x_3) + x_4 + (1 - x_5) + (1 - x_6) \leq m$$

IX. Multiperiod Lot Sizing model

A. Introduzione

Discretizziamo il tempo e quindi divido la pianificazione in slot di tempo esistenti come ore, settimane, mesi. Avremo allora una domanda d_t che fa rferimento al periodo $t = 1, ..., T$ con T lunghezza dell'orizzonte di pianificazione.

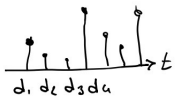

Figura 53. Discretizzazione del tempo

Dobbiamo definire:

1) varaibili generali:
   - f_t: **costi fissi di produzione**, potrebbero dipendere dal periodo ma **non dipendono dalla quantità prodotta** (es: gas, elettricità)
   - c_t: **costi variabili** per la produzione di un prodotto, potrebbe dipendere dal periodo (es: vagetali)
   - h_t: **costi di stockaggio** per un periodo t
   - Q: **capacità di magazzino**
2) variabili decisionali:
   - y_t: **variabili di accenzione**

$$y_t = \begin{cases} = 1 \text{se è acceso} \\ = 0 \text{se è spento} \end{cases}$$

   - x_t: indica la **produzione nel periodo** t
   - I_t: indica il **livello di scorte nel periodo** t
3) funzione obiettivo:

$$z = \text{ costi fissi } + \text{ costi variabili } + \text{ costi di inventario}$$

$$z = \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} c_t x_t + \sum_{t=1}^{T} h_t I_t$$

4) vincoli:
   - di conservazione delle scorte: cioè il livello di scorte a fino al periodo precedente e la merce prodotta, togliendo ciò che diamo al mercato:

$$I_{t-1} + x_t + -d_t = I_t$$

   - di capacita': dove **il livello di scorte non deve superare la capacità di magazzino**

$$I_t \leq Q$$

   - lega x_t cno y_t: **lega accenzione e spegnimento** dei macchinari

$$x_t \leq (\sum_{i=t}^{T}) d_i \quad \forall \quad t = 1, .., T$$

   essendo y_t binaria, avremo:

$$y_t = \begin{cases} = 1 \rightarrow x_t \leq \sum_{i=t}^{T} d_i \\ = 0 \rightarrow x_t \leq 0 \rightarrow x_t = 0 \end{cases}$$

   - sull'inventario: $I_0, I_T = 0$

B. Aggiornamento Beer Game - Backlog

In un problema reale non sono certo di soddisfare la domanda, infatti si potrebbe andare sotto scorta per cercare di soddisfarla con un incremento del costo. Il livello delle scorte si divide in una parte positiva ed una negativa:

$$I_t = I_t^+ - I_t^- \leq \geq 0$$

con I_t^+ livello di inventario e I_t^- backlog.
avremo allora:

1) funzione obiettivo:

$$z = \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} c_t x_t + \sum_{t=1}^{T} h_t I_t^+ + \sum_{t=1}^{T} b_t I_t^-$$

2) vincoli:
   - di conservazione delle scorte:

$$I_{t-1}^+ - I_{t-1}^- + x_t - d_t = I_{t-1}^+ - I_{t-1}^- \quad \forall \quad t = 1, ..., T$$

   - di capacita':

$$I_t^+ \leq Q$$

   - lega x_t cno y_t:

$$x_t \leq (\sum_{i=t}^{T}) d_i \quad \forall \quad t = 1, .., T$$

C. Aggiornamento Beer Game - Lead time $l \in \{0, 1, 2, ...\}$

Durante il periodo di arrivo degli ordini le prime l settimate non posso sperare di ricevere rifornimenti, infatti arrivera' nel periodo $t + l$. Abbiamo i vincoli:

- prime l settimane:

$$I_{t-1}^+ - I_{t-1}^- - d_t = I_{t-1}^+ - I_{t-1}^- \quad \forall \quad t = 1, ..., l$$

- le altre settimane:

$$I_{t-1}^+ - I_{t-1}^- + x_{t-l} - d_t = I_{t-1}^+ - I_{t-1}^- \quad \forall \quad t = l+1, ..., T$$

X. Network Flow Model

A. Introduzione

dove abbimao un grafo con dei sui parametri che assieme formano un network. dobbiamo allora capire con=me un flusso di dati transitra nellarete. primo porblema che vediamo PROBLEMA DI FLUSSO A COSTO MINIMO:

ha la varainte lineare variante single commodity

in questa classe di problemi abbiamo un grafo G(V, A) con v vertici e A archi

img

inq uestso caso ogni vetice $i \in V$ o generea o assorbe un flusso di materiali, dati, ecc...

parametro per i nodi: questo $\forall i \in V : d_i$ ¿ 0 per i = sorgente (souce) = 0 per i = transito ¡ 0 per i = pozzo (sink)

se $d_i > 0 indicaunafornitura < 0 indicaunadomanda$

caristiche: single commodity: flusso omogeneo multiple comodity: varie tipologie di flusso

parametri per gli archi prendo un parametro $c_{ij} cioeunarcochecolegainodiiej . c_{ij} = costo(ditraspost)unitario(perunitadiflusso)altroparamentroq_{ij} : \quad capacitamassima(qunatoflussopuotransitrardaiaj)$

decisione da prendere: allocaizone del flusso, cioe qunto flusso deve transitare su ogni arco le varaibili saranno: $x_{ij} >= 0 flusochetransitadaiajperunitaditempo$

affinche si possa avere una soluzione ammissibile una condizione necessaria è che se sommiamo su utti i vertici le quntita $d_i devoavere 0 : \sum_{i \in V} d_i = 0$

quindi non sara sufficiente per via delle capacita degli archi

misure di prestazione della f.o.: vigliamo allora minimizzare il costo totale di trasporto su tutti gli archi:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

notaione epr definire un insieme di vincoli: dato uun nodo i al quale abbiamo archi uscenti ed entranti allora tutti li archi entranti associamo un insieme $\delta^-(i)$ per quellio uscenti: $\delta^+(i)$

definiiamo allora i vincoli, dobbiamo tenere in conto dei vincoli di conservazione del flusso: cioe che dtutto quello cheentra nel nodo deve uscirci:

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = d_i \quad \forall \quad i \in V$$

altri vincoli per avere una soluzione ammissibile abbiamo ance dei vincoli di capacita: avremo un vincolo associato ad ogni arco:

$$x_{ij} \leq q_{ij} \quad \forall \quad i,j \in A$$

vincoli di non negativita:

$$x_{ij} \geq 0 (\geq l_i) \quad \forall \quad i,j \in A$$

dove $l_i quantitaminimiaincasodievenienza$

la formulazione compatta dice:

$$\min z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t - $\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = d_i \quad \forall \quad i \in V$ - $x_{ij} \leq q_{ij} \quad \forall \quad i,j \in A$ - $x_{ij} \geq 0 (\geq l_i) \quad \forall \quad i,j \in A$

formulaizone estesa: in una topologia di rete cone questa:

img

ad ognmi arco indico $c_i jeq_i jeperogninodod_i$

allora la fomulazione estesa è: (prendendo il parametro del costo)

$\min z = 2x_{12} + 5x_{13} + 3x_{23} + 3x_{24} + 3_{x32} + 4x_3 4 + 3x_{42} s.tperi = 1 :$

$x_{12} + x_{13} = 10$ per i = 2: $x_2 3 + x_{24} - x_{12} - x_{42} - x_{32} = 0$ per i = 3: $x_{32} + x_{34} - x_{13} - x_{23} = -3$ per i = 4: $x_4 2 - x_{24} - x_{34} = -7$

guardando la capacita scriviamo anche: $x_{12} \leq 8 \; x_{13} \leq 2 \; x_{23} \leq 4 \; x_{24} \leq 7 \; x_{32} \leq 4 \; x_{34} \leq 5 \; x_{42} \leq 7$

$x_{12}, x_{13}, x_{23}, x_{24}, x_{32}, x_{34}, x_{42} \geq 0$

se un instanza del problema è ammissiibile, cioe se lo sono i dati del problema, allora: - puo esistere una soluzione ottima - puo esere unbounded se esiste qualeche arco (i, j) $\in A con costonegatico_{ij} < 0$

img

poiche il costo totale negativo nel loop 2,4,5 allora abbiamo:

$x_{12} = 1, x_{23} = 1, x_{24} = x_{45} = x_{52} = +\infty$

siuponendo capacita infinita

importante condiione è quella di interezza: dice che se i dati $d_i elecapacita q_{ij} sonointeri, alloraesisteunasoluizoneottimanonunbou$

abbiamo dei casi speciali del problema di flusso a socoto minimo: - problem a di trasprto (rtansportation problem) - prob di assegnamento (assignment problem) - problema di cammino piu breve/rapido (shortesr/quickest path) - problema di massimo flusso (aximum flow problem)

PROBLEMA DI TRASPORTO: consideriamo la variante di simple commodity(supponiamo di avere un grafo diaprtito dove l'insieme dei vertici è composto da due insiemi disgiunti: sorgenti (fino a m) e pozzi (fino a n). ad ogni soegente è associata un $s_1, .., s_m \geq 0 chedefinisconolafornitureaeaipozziinveceabbiamounaqunatitab_1, .. pozzoeperogniarcoijèassociatouncostocij(costoitrasportoperunitadi$

img

allora il grafo G è dato da: G = (V=$v_1 \cup V_2, A) con V_1 \cap V_2 = insiemevuotoevienedettografobipartitoedèquestoproblemauncasosp - nonabbiamovincolidicapacita - nonabbiamonodiditransito$

il problema sara allora ammissiible se:

$$\sum_{i \in V_1} s_i = \sum_{i \in V_2} b_i$$

allora le varaibili abbiamo che:

$$x_{ij}, i_1, j \in V_2$$

che indica qunate unuta di flusso ("merce") vengono gtraspostate da i a j

l'obiettivo è minimizzare al f.o. quindi la misura di prestazione sara: $\min z = \sum_{(i,j) \in A} \sum_{j_2} c_{ij} x_{ij}$ alora minimizziamko il costo totale.

vincoli:

$$\sum_{j \in V_2} x_{ij} = s_i \quad \forall \quad i \in V_1$$

per ongi vertice sorgente

$$\sum_{i \in V_1} x_{ij} = b_i \ \ forall \quad j \in V_2$$

per ongi vertice pozzo

$$x_{ij} \geq 0 \quad \forall \quad i \in V_1, j \in v_2$$

PROBLEMA DI ASSEGNAMENTO LINAERE: abbiamo sempre un siseme di vertici diviso in due grupi: n risorse e n tasks obiettvo assengare le risorse ai tasks

img

assegnamo iun costo sll'arco ij con c_{ij}.

ad una risorsa possiamo associare un solo task. es: magazzini (impianti) automatizzati: n = 4 con n risorse = ABV (automated guided vehicle) abbiamo dei task da compiere come lo spostamente di un veicolo da un punto ad un altro

img

avremo allora i costi di oni veicolo per spostarsi allinizio del task costi = tempo di viaggio dal puno di prelievo al punto di inizio

quindi questa tipologi a di problemi è un acso speciale del prtiblema di trasposto dove cambia solo che: - m= n - $s_i = 1 \forall i - b_i = 1 \forall j$

le variabili saranno allora: $x_{ij} = 1 selarosorsaièaasegnataaduntaskj = 0 altrimenti$

vorremo allora minimizzare la f.o.: $\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ s.t. - $\sum_{j=1}^{n} x_{ij} = 1 \quad \forall \quad \imath = 1, ..., n$ cioe ogni risrosa deve essere assegna ad un solo task - $\sum_{i=1}^{n} x_{ij} = 1 \quad \forall \quad j = 1, ..., n$ cioe ongi task deve essere assegnato ad una sola risorsa

dato che $s_i = 1 con i = 1, ..., n e b_j = 1 con j = 1, ..., n$

possiamo imporre che $x_{ij} \geq 0 impedendochelavariabilesia <= 0 qustograzieall; interezzadeidatieaivincoliquisopracheimpedisconochelevaraibiliassumanovalore > 1 quidnipotremoavereconvalore 0 o 1.$