

# Ottimizzazione Combinatoria

## Appunti

Giovanni "Qua' Qua' dancer" Palma e Alex Basta

# Contents

# Chapter 1

## Introduzione

Prova scritta e orale. Si studiano metodi algoritmici per ottimizzare problemi di flusso su reti e di programmazione lineare. In poche parole, impariamo come prendere decisioni. Appunti presi in base alle lezioni del prof Ugo Dal Lago, si consiglia una lettura con accento veneto

Questa introduzione è totalmente inutile, serve solo ad occupare byte. Ringraziate pure il bastianini per questo ☹

# Chapter 2

## Problemi e Modelli

### 2.1 Problemi di ottimizzazione

#### Definition 2.1.1: Ricerca operativa

La **ricerca operativa** è un ramo della matematica applicata che si occupa dello studio, della modellizzazione e della risoluzione dei cosiddetti *problemi decisionali* complessi mediante strumenti matematici, algoritmici e computazionali, con l'obiettivo di ottimizzare processi e risorse

Per evitare qualsivoglia fraintendimento fornirò anche la definizione di **ottimizzazione Combinatoria**

#### Definition 2.1.2: Ottimizzazione Combinatoria

Si definisce **Ottimizzazione Combinatoria** una branca della Ricerca Operativa che nel modellare matematicamente e risolvere problemi complessi di natura discreta unisce tecniche di calcolo combinatorio alla teoria degli algoritmi e ai risultati teorici e metodologici della programmazione lineare

Pertanto ricerca operativa e ottimizzazione combinatoria sono due cose diverse, MA cito testualmente

”Per tutti i nostri scopi ricerca operativa e ottimizzazione, sono sinonimi  
tuttavia non vedremo solo alcune tecniche di ottimizzazione combinatoria, ma anche altre tecniche che stanno nella ricerca operativa ma che trattano di valori non discreti”

– Ugo

Adesso, sotterrato questo problema di carattere unicamente terminologico con cui io non posso fare a meno di strizzarmi il cervello perché c'ho l'autismo, possiamo tornare a parlare di ricerca operativa/ottimizzazione combinatoria (tanto so' sinonimi per noi)

I problemi di cui si occupa la ricerca operativa, quindi, riguardano situazioni in cui occorra massimizzare i ricavi o minimizzare i costi, in presenza di risorse limitate. Detto in termini più matematici, data una funzione **vincolata** l'obiettivo è trovare una soluzione ottimale che massimizzi o minimizzi tale funzione.

È pertanto vero, quindi, che questa disciplina ha forte contenuto economico

La ricerca operativa si inserisce all'interno del processo decisionale, il quale può essere suddiviso in diverse fasi

- **Individuazione problema**
- **Raccolta dati**
- **Costruzione modello**, ovvero la Traduzione del problema in un modello matematico che descriva il sistema e i vincoli in modo formale
- **Determinazione di piu' soluzioni**: applicazione di algoritmi e tecniche di ottimizzazione per individuare la soluzione migliore
- **Analisi dei risultati**

La ricerca operativa, quindi, si occupa delle fasi 3 e 4 del processo, dato che sono le fasi che richiedono l'impiego di modelli matematici, algoritmi di ottimizzazione e strumenti computazionali. Adesso andiamo a definire per benino che cosa intendiamo per "modello"

**Definition 2.1.3: modello**

un **modello** è una descrizione astratta e scritta in linguaggio matematico, della parte di realtà utile al processo decisionale

I modelli ci permettono di inquadrare i problemi in una determinata "cornice" che ci permette di determinare quale tipo di algoritmo risolutivo usare.

Esistono tre tipi di modelli:

- **Teoria dei giochi:** ricerca di un equilibrio fra le componenti coinvolte in un'interazione reciproca, spesso con obbiettivi contrastanti. (non ce ne occupiamo)
- **Simulazione:** il problema viene studiato simulando la situazione senza studiarne la natura in modo analitico tramite generazione di istanze casuali. (anche questi modelli non ci interessano)
- **Analitici:** dal problema si costruisce un modello matematico rigoroso (senza perdere informazione sul problema reale) e risolto mediante tecniche analitiche, senza ricorrere a simulazioni. La natura stessa dello spazio matematico in cui è inserito il problema è in grado di garantire la soluzione ottima. Questo tipo approccio è particolarmente vantaggioso in quanto assicura l'esattezza della soluzione supponendo che il modello sia formulato correttamente.

È tuttavia richiesto un discreto livello di creatività

Definiamo, adesso, i problemi che andiamo a trattare

**Definition 2.1.4: Problema**

Definiamo **problema** una domanda, espressa in termini generali, la cui risposta dipende da *parametri* e *variabili*, soprattutto nei problemi analitici

Un problema  $\mathcal{P}$  è descritto tramite:

- I suoi parametri e variabili
- Le caratteristiche che una soluzione deve avere

Quando fissiamo un'istanza di un problema, vengono fissati i parametri ma non le variabili, che sono le incognite che devono essere definite. Distinguiamo un problema dalla sua istanza per generalizzarlo. Si presti attenzione alla differenza tra parametri e variabili che molti si confondono

**Example 2.1.1** (Problema con parametri e variabili)

Sia  $\mathcal{P}$  il seguente problema

$$ax^2 + bx + c = 0$$

Dove  $a, b$  e  $c$  sono i suoi parametri e  $x$  rappresenta le variabili, una possibile istanza di tale problema è:

$$5x^2 - 6x + 1 = 0$$

Un modo comune per descrivere un problema è dare l'insieme di soluzioni ammissibili  $\mathbb{F}_{\mathcal{P}} \subseteq G$ , dove  $G$  è un sovrainsieme generico noto, di solito contenente la collezione di tutte le possibili configurazioni o decisioni che si possono prendere, dando dei vincoli che un generico  $g \in G$  deve soddisfare per far parte di  $\mathbb{F}_{\mathcal{P}}$ , avremo così che  $G - \mathbb{F}_{\mathcal{P}}$  è l'insieme delle soluzioni non ammissibili

**Example 2.1.2**

Sia l'istanza di  $\mathcal{P}$  definita precedentemente

$$5x^s - 6x + 1 = 0$$

si ha che

$$\begin{aligned} \mathbb{G} &= \mathbb{R} \\ \mathbb{F}_{\mathcal{P}} &= \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\} \end{aligned}$$

### 2.1.1 Problemi di ottimizzazione

Iniziamo con una definizione preliminare.

#### Definition 2.1.5: Problema di ottimizzazione

In matematica e in informatica, un problema di ottimizzazione è il problema di trovare la migliore soluzione fra tutte le soluzioni fattibili.

Un problema di ottimizzazione  $\mathcal{P}$  viene descritto:

- Dando l'insieme  $\mathbb{F}_{\mathcal{P}}$  delle soluzioni ammissibili
- Specificando una funzione obiettivo  $c_{\mathcal{P}} : \mathbb{F}_{\mathcal{P}} \rightarrow \mathbb{R}$  che assegna ad ogni  $g \in \mathbb{F}_{\mathcal{P}}$  un valore reale  $c_{\mathcal{P}}(g)$ , che rappresenta il costo o il beneficio.

Un problema (di ottimizzazione) di massimo  $\mathcal{P}$  consiste nel determinare il valore

$$Z_{\mathcal{P}} = \max\{c_{\mathcal{P}}(g) \mid g \in \mathbb{F}_{\mathcal{P}}\},$$

mentre un problema (di ottimizzazione) di minimo  $\mathcal{P}$  consiste nel determinare il valore

$$Z_{\mathcal{P}} = \min\{c_{\mathcal{P}}(g) \mid g \in \mathbb{F}_{\mathcal{P}}\}.$$

Ci si può trastullare con la definizione, infatti ad ogni problema di massimo  $\mathcal{P}$  corrisponde un problema di minimo  $\mathcal{P}'$  tale che  $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$ , ovvero:

$$Z_{\mathcal{P}} = -\min\{c_{\mathcal{P}'}(g) \mid g \in \mathbb{F}_{\mathcal{P}} = \mathbb{F}_{\mathcal{P}'}\}.$$

#### Definition 2.1.6: Valore ottimo e soluzione ottima

Dato un problema di ottimizzazione  $\mathcal{P}$ , il valore  $Z_{\mathcal{P}}$  definito in precedenza è detto valore ottimo, mentre il  $g^* \in \mathbb{F}_{\mathcal{P}}$  tale che  $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$  è detto soluzione ottima.

Si può quindi constatare che la differenza reale tra i due è che il valore ottimo è inserito nel codominio della funzione obiettivo (cioè il mero valore reale “ottimizzato”), mentre la soluzione ottima appartiene al dominio (ovvero è l'elemento in  $\mathbb{F}_{\mathcal{P}}$  che ottimizza la funzione).

#### Example 2.1.3 (Esempio di problema di ottimizzazione)

Dati

$$\mathbb{G} = \mathbb{R}, \quad \mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\}, \quad c_{\mathcal{P}} : \mathbb{R} \rightarrow \mathbb{R} \text{ con } c_{\mathcal{P}}(g) = g^2,$$

e si ponga

$$Z_{\mathcal{P}} = \max\{x^2 \mid x \in \mathbb{F}_{\mathcal{P}}\}.$$

Innanzitutto, calcolo l'insieme delle soluzioni ammissibili (hold my soluzione parabolica):

$$x = \frac{6 \pm \sqrt{(-6)^2 - 4 \cdot 5 \cdot 1}}{2 \cdot 5} = \frac{6 \pm \sqrt{36 - 20}}{10} = \frac{6 \pm \sqrt{16}}{10} = \frac{6 \pm 4}{10}.$$

Quindi, le soluzioni sono:

$$x_1 = \frac{6+4}{10} = 1 \quad \text{e} \quad x_2 = \frac{6-4}{10} = \frac{1}{5}.$$

Pertanto, l'insieme delle soluzioni ammissibili è:

$$\mathbb{F}_{\mathcal{P}} = \{1, \frac{1}{5}\}.$$

Si procede poi nel calcolare la funzione obiettivo per ogni elemento:

$$c_{\mathcal{P}}(1) = 1^2 = 1, \quad c_{\mathcal{P}}\left(\frac{1}{5}\right) = \left(\frac{1}{5}\right)^2 = \frac{1}{25}.$$

Il massimo tra questi valori è:

$$Z_{\mathcal{P}} = \max\{1, \frac{1}{25}\} = 1.$$

fin casi dei problemi di decisione

Si hanno quattro casi principali in cui sono inseriti i problemi di decisione:

- **Problema vuoto:** non esistono soluzioni ammissibili, ovvero  $\mathbb{F}_{\mathcal{P}} = \emptyset$ , per cui l'ottimizzazione è impossibile e si assume che  $Z_{\mathcal{P}} = \infty$ .
- **Problema illimitato:** si ha quando non esiste un limite inferiore/superiore per i valori della funzione obiettivo tra le soluzioni ammissibili, ovvero quando  $Z_{\mathcal{P}} = \pm\infty$ . Ad esempio, nel caso del massimo, per ogni  $x \in \mathbb{R}$  esiste un  $g \in \mathbb{F}_{\mathcal{P}}$  con  $c_{\mathcal{P}}(g) \geq x$ , e quindi  $Z_{\mathcal{P}} = +\infty$ .
- **Valore ottimo finito ma non soluzione ottima finita:** un problema di ottimizzazione può presentare un valore ottimo finito, pur non ammettendo alcuna soluzione  $g \in \mathbb{F}_{\mathcal{P}}$  tale che  $c_{\mathcal{P}}(g)$  sia esattamente uguale a  $Z_{\mathcal{P}}$ . (Es.: nell'insieme  $\{x \mid x > 0\}$ , dove l'estremo inferiore è 0, ma non esiste una soluzione che raggiunga tale valore.)
- **Valore ottimo finito e soluzione ottima finita:** esiste almeno un  $g \in \mathbb{F}_{\mathcal{P}}$  tale che  $c_{\mathcal{P}}(g) = Z_{\mathcal{P}}$ , ed esso è ottimo. Notare che possono esistere più soluzioni ottime, ma un solo valore ottimo.

## 2.1.2 Problemi di ottimizzazione e di decisione

Verranno fornite due definizioni importanti:

### Definition 2.1.7: Problema di decisione

Un problema di decisione consiste nel determinare una qualunque soluzione ammissibile  $g \in \mathbb{F}_{\mathcal{P}}$ .

### Definition 2.1.8: Problema di certificato

Il problema di certificato consiste nel verificare se, per un dato  $g \in G$ , risulti che  $g \in \mathbb{F}_{\mathcal{P}}$ .

I problemi di decisione sono generalmente più semplici dei problemi di ottimizzazione, in quanto non richiedono la definizione di una funzione obiettivo  $c_{\mathcal{P}}()$ . Tuttavia, è possibile trasformarli in problemi di ottimizzazione fissando una funzione obiettivo triviale (ad esempio, costante), per cui tutte le soluzioni ammissibili risultano ottime.

In alternativa, possiamo considerare un problema decisionale  $R$  definito da

$$F_R = \{g \in \mathbb{F}_{\mathcal{P}} \mid c_{\mathcal{P}}(g) = Z_{\mathcal{P}}\},$$

oppure, per un dato  $k \in \mathbb{R}$ , il problema decisionale  $R_k$  con

$$F_{R_k} = \{g \in \mathbb{F}_{\mathcal{P}} \mid c_{\mathcal{P}}(g) \leq k\},$$

nel caso in cui  $\mathcal{P}$  sia un problema di minimo.

## 2.1.3 Aspetto algoritmico

- **Algoritmi esatti:** e' un algoritmo che preso un'istanza di  $P$  ( $P$  e' un modello di un problema), fornisce in output una soluzione ottima  $g^*$  di  $P$  (se esiste). Spesso pero' i problemi sono troppo complessi ed e' impossibile costruire algoritmi efficienti.
- **Algoritmi euristici:** non ci danno garanzia sulla soluzione trovata (e' sicuramente ammissibile), ma un'approssimazione.

Come possiamo valutare la correttezza di una soluzione euristica? Possiamo misurare l'errore (assoluto o relativo) fra il valore ottimo euristico e quello esatto.

Gli algoritmi euristici vengono detti anche greedy.

## 2.2 Modelli

Al posto di dare un'algoritmo per ogni specifico problema, possiamo definire classi di problemi che possono essere risolti con lo stesso algoritmo.

### 2.2.1 Programmazione Lineare

Fornisco Innanzitutto la def. di problema di Programmazione Lineare

#### Definition 2.2.1: Problema di Programmazione Lineare

Si definiscono **Problemi di programmazione lineare (PL)** tutti quei problemi di ottimizzazione in cui la funzione obiettivo  $c_P$  è lineare e i vincoli sono *tutti espressi da disequazioni lineari* ed anche, eventualmente, equazioni lineari (quest'ultime possono mancare le prime no).

In particolare in un problema  $\mathcal{P}$  di programmazione lineare si ha:

- $\mathbf{G} = \mathbb{R}$ , definendo poi un numero finito  $n \in \mathbb{N}$  di variabili reali, che nella realtà rappresentano delle quantità

$$x = (x_1, \dots, x_n) \in \mathbb{R}^n$$

- Una funzione obiettivo  $c_P$  definita  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  nella forma:

$$f(x) = cx$$

Dove  $c$  è un vettore riga e  $x$  è un vettore colonna. Si noti che  $c$  non è una variabile, bensì un parametro (definizione della funzione obiettivo)

- Un insieme di  $m$  vincoli lineari, tutti in una delle forme seguenti:

$$ax = b \quad ax \leq b \quad ax \geq b$$

Dove  $a \in \mathbb{R}^n$  e  $b \in \mathbb{R}$

$f$  è lineare.  $c_P$  è l'insieme di vettori di numeri reali.  $\mathbf{G} = \mathbb{R}^n$ . (definizione di  $G$ )

È talvolta utile assumere che  $x \in \mathbb{Z}^n$ , ovvero che le soluzioni ammissibili siano vettori di numeri interi. In questo caso si parla di *programmazione lineare intera*. Facendo così, stiamo restringendo il campo di ricerca (ed il dominio delle soluzioni possibili), ma si perdono alcune proprietà (geometriche) che in realtà possono rendere più difficile la ricerca della soluzione.

Esiste inoltre la *programmazione lineare mista* (PLM), dove si hanno variabili di natura mista (alcune variabili in  $\mathbb{R}$  ed alcune in  $\mathbb{Z}$ )

Si noti questo diagramma riassuntivo:



Un problema PL può sempre essere espresso nella seguente forma matriciale:

$$\max\{cx \mid Ax \leq b\}$$

dove  $A \in \mathbb{R}^{m \times n}$  e  $b \in \mathbb{R}^m$ . Grazie ad alcuni accorgimenti è possibile, inoltre, scrivere tutti i vincoli possibili di un problema di PL  $\mathcal{P}$  in un'unico sistema di disequazioni lineari. Infatti:

- Se  $\mathcal{P}$  è un problema di minimo, occorre considerare semplicemente la funzione  $f(x) = (-c)x$



- Ogni vincolo  $ax = b$  diventa la coppia di vincoli  $ax \leq b$  e  $ax \geq b$
- Ogni vincolo  $ax \geq b$  è equivalente a  $(-a)x \leq (-b)$ .

### Example 2.2.1 (Pianificazione della Produzione)

#### TESTO

La società Pintel deve pianificare la produzione della sua fabbrica di micro-processor. La Pintel possiede due diverse linee di prodotti: i processori Pintium, più potenti e destinati al mercato “server”, ed i Coloron, meno potenti e destinati al mercato “consumer”. L’impianto è in grado di realizzare 3.000 “wafer” alla settimana: su ogni wafer trovano posto o 500 Coloron oppure 300 Pintium. La resa di un wafer dipende anch’essa dal tipo di processore: i Coloron, di minori dimensioni, hanno una resa media del 60%, mentre i Pintium, più grandi e quindi maggiormente sottoposti a difetti, solamente del 50%. I processori Pintium si vendono a 500\$ al pezzo, mentre i Coloron si vendono a 200\$ al pezzo. La divisione commerciale della Pintel ha anche stabilito che la massima quantità di processori che possono essere messi sul mercato ogni settimana senza causare un calo dei prezzi è di 400.000 unità per i Pintium e di 700.000 unità per i Coloron. Si vuole determinare le quantità di ciascun tipo di processore da produrre settimanalmente in modo da massimizzare il ricavo totale.

#### SVOLGIMENTO

##### • VARIABILI

Prendiamo le prime due variabili ovvie:

$x_p$  = “numero di pintium prodotti”

$x_c$  = “numero di coloron prodotti”

Notare che a differenza delle dispenze,  $x_p$  e  $x_c$  sono il numero di processori che *proviamo* a produrre, senza tenere conto della resa del wafer.

Usiamo anche due variabili che poi possiamo anche rimuovere:

$w_p$  = “numero wafer utilizzati per pintium”

$w_c$  = “numero wafer coloron”

##### • VINCOLI

Mettiamo i vincoli di produzione settimanali:

$$0 \leq x_p \leq 400000$$

$$0 \leq x_c \leq 700000$$

Dobbiamo considerare anche il massimo numero di wafer, ipotizzo che un wafer può essere usato tutto solo per pintium o solo per coloron:

$$w_p + w_c \leq 3000$$

Queste nuove variabili devono essere legate in qualche modo a quelle principali che usiamo nella FO:

$$300w_p = x_p$$

$$500w_c = x_c$$

Possiamo quindi tornare sul vincolo sul numero di wafer e sostituire  $w_p, w_c$  con  $x_p, x_c$  per riscrivere il vincolo relativo alle ultime due variabili, riducendo quindi le variabili utilizzate e semplificando il problema:

$$5x_p + 3x_c \leq 4500000$$

L’insieme delle soluzioni ammissibili è quindi:

$$F = \{(x_p, x_c) | 0 \leq x_p \leq 400000, 0 \leq x_c \leq 700000, 5x_p + 3x_c \leq 4500000\}$$

- Funzione Obbiettivo

Vogliamo trovare  $\max\{c(x_p, x_c) | (x_p, x_c) \in F\}$

$$c(x_p, x_c) = 0.5x_p \cdot 500 + 0.6x_c \cdot 200$$

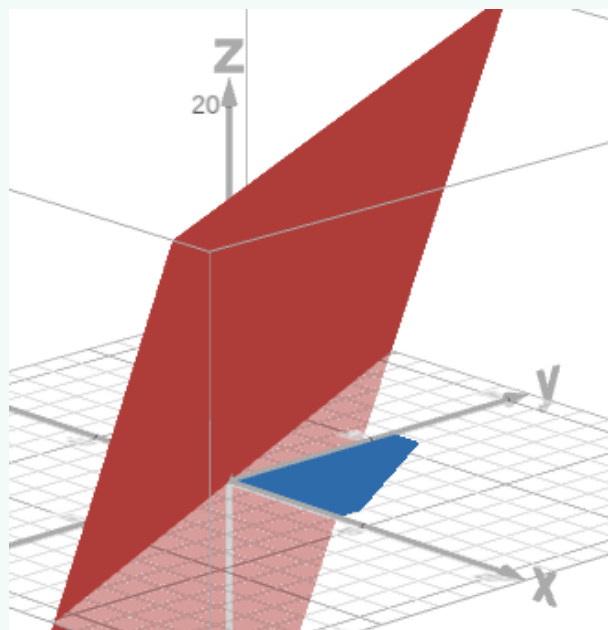
Vediamo su un grafico l'area delle soluzioni ammissibili data da  $F$ :



Figura 1.1: Insieme ammissibile per il problema della Pintel

Un'importante proprietà dei problemi lineari (reali) è che la soluzione ottimale si trova sempre su uno dei "vertici" della figura descritta dalle soluzioni ammissibili (in questo caso quella ottimale è  $(400000, 100000)$  per  $(x_p/0.5, x_c/0.6)$ ).

Grafico della funzione costo (rossa) con l'area delle soluzioni ammissibili (blu):



## OSSERVAZIONI

In realtà, con la soluzione trovata il numero di wafer per tipo di chip non è intero! In questo caso possiamo supporre che non sia troppo un problema, ma in altri casi può esserlo, quindi tocca studiare la *programmazione lineare intera*.

### 2.2.2 Programmazione lineare intera

Se nella *PL* le variabili rappresentano quantità nella *PLI* le variabili possono essere

- **Quantitative:** ovvero rappresentare quantità.
- **Logiche:** rappresentare valori booleani, ovvero scelte basate sulla possibilità di scegliere o meno una determinata opzione

Qui la definizione di variabile logica

#### Definition 2.2.2: variabile logica

Una **variabile**  $x$  si definisce **logica** se:

$$x \in \mathbb{N}(\mathbb{Z}) \quad 0 \leq x \leq 1$$

Un esempio tipico è rappresentato dalle variabili che associano una determinata risorsa a un compito specifico, oppure che determinano l'utilizzo di un particolare processo.

Questo è il bello dei problemi lineari

– Il Basta

Un giga esempietto bellino è il problema dello zaino:

#### Example 2.2.2 (Problema dello zaino)

Il problema dello zaino è un esempio classico di problema di ottimizzazione combinatoria piuttosto complesso.

Sia dato un insieme  $E = 1, 2, \dots, n$  di elementi, a ciascuno dei quali sia assegnato un peso  $a_i$  ed un costo  $c_i$ ,  $i = 1, \dots, n$ , interi e positivi: il problema dello zaino (KP, da Knapsack Problem) consiste nel determinare un sottoinsieme di elementi che abbia costo totale massimo ed il cui peso totale non superi un prefissato intero  $b$ . **Parametri:**

$$E = \{1, \dots, n\}$$

$$a_i = \text{peso dell'oggetto } i$$

$$c_i = \text{costo dell'oggetto } i$$

$$b = \text{peso massimo dello zaino}$$

#### Variabili:

Dobbiamo indicare usando le variabili quali elementi vogliamo includere nella soluzione e quali vogliamo scartare. Possiamo fare ciò usando  $n$  variabili logiche:

$$x_i = \begin{cases} 1 & \text{se l'oggetto } i \text{ è nello zaino} \\ 0 & \text{altrimenti} \end{cases}$$
$$x_i \in \{0, 1\} \quad \forall i \in \{1, \dots, n\}$$

#### Vincoli:

$$\sum_{i=1}^n x_i a_i \leq b$$

#### Funzione obiettivo:

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

## Relazioni logiche

Avendo introdotto le variabili logiche, è opportuno chiedersi se sia possibile esprimere, mediante programmazione lineare (PL), le regole di inferenza logica per le relazioni che intercorrono tra di esse. Grazie all'uso di opportuni vincoli lineari, la risposta è affermativa:

- **Negazione** ( $y = \neg x$ ):

$$x = 1 - y$$

- **Congiunzione** ( $z = x \wedge y$ ):

$$z \leq x$$

$$z \leq y$$

$$z \geq x + y - 1$$

- **Disgiunzione** ( $z = x \vee y$ ):

$$z \geq x$$

$$z \geq y$$

$$z \leq x + y$$

- **Implicazione** ( $z = x \implies y$ ):

$$x + z \geq 1$$

$$z \geq y$$

$$x + z \leq 1 + y$$

È possibile, tuttavia, dimostrare che il problema di ottimizzazione lineare è, in generale, *NP-hard*, dato che il problema di soddisfacibilità di una formula logica rientra nella classe NP-hard

## 2.3 Template per risolvere problemi

### 2.3.1 Vincoli di assegnamento

Un tipo di vincolo che la programmazione lineare intera (PLI) gestisce in modo molto efficace è rappresentato dai cosiddetti *vincoli di assegnamento*. Questi vincoli sono utilizzati per modellare problemi che riguardano l'assegnazione di "oggetti a luoghi".

Si considerano:

- Un insieme  $N = \{1, \dots, n\}$  di **oggetti** (che possono rappresentare automezzi, persone, ecc.)
- Un insieme  $V = \{1, \dots, m\}$  di **luoghi**

Per rappresentare le diverse condizioni di assegnazione degli oggetti ai luoghi, si introduce la variabile  $x_{ij} \in \{0, 1\}$  (dove  $1 \leq i \leq n$  e  $1 \leq j \leq m$ ). Questa variabile indica se l' $i$ -esimo oggetto è stato assegnato al  $j$ -esimo luogo ( $x_{ij} = 1$ ) o meno ( $x_{ij} = 0$ ), si può così procedere alla formale

### Definition 2.3.1: Vincoli di assegnamento

Sia  $N = \{1, \dots, n\}$  un insieme di oggetti e  $V = \{1, \dots, m\}$  di luoghi, si definiscono **vincoli di assegnamento** quei vincoli che impongono che *ogni oggetto sia assegnato a un solo luogo e ad ogni luogo è assegnato esattamente un oggetto*. Questi sono espressi mediante queste sommatorie:

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

Dove:

- $x_{ij}$  è una variabile binaria che vale 1 se l'oggetto  $i$  è assegnato al luogo  $j$ , e 0 altrimenti.
- La prima equazione assicura che ogni oggetto  $i$  sia assegnato esattamente a un luogo.
- La seconda equazione assicura che ogni luogo  $j$  riceva esattamente un oggetto.

il luogo può anche essere visto come uno slot temporale, quindi imponiamo un certo pipeline

### Vincoli di semi-assegnamento

I vincoli di semi-assegnamento sono una variante dei vincoli di assegnamento in cui non è necessario che ogni luogo riceva esattamente un oggetto, in particolare si noti la seguente definizione

### Definition 2.3.2: Vincoli di semi-assegnamento

Sia  $N = \{1, \dots, n\}$  un insieme di oggetti e  $V = \{1, \dots, m\}$  un insieme di luoghi. Si definiscono **vincoli di semi-assegnamento** quei vincoli che impongono che *ogni oggetto sia assegnato ad al massimo un luogo*. Questi vincoli sono espressi mediante la seguente sommatoria:

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

Dove  $x_{ij}$  è una variabile binaria che vale 1 se l'oggetto  $i$  è assegnato al luogo  $j$ , e 0 altrimenti.

### Example 2.3.1 (Problema del docente di informatica)

#### Testo

Si hanno dei progetti  $(t_1, t_2, \dots, t_n)$ , si hanno  $m$  PC, dove ogni pc può compilare qualunque progetto in modalità sequenziale. Le prestazioni del PC sono identiche

Vogliamo assegnare i progetti ai pc in modo da minimizzare il tempo complessivo parallelo di compilazione

#### Svolgimento

Variabili:

$$x_{ij} = \begin{cases} 1 & \text{se il progetto } i \text{ è compilato al pc } j \\ 0 & \text{altrimenti} \end{cases}$$
$$x_{ij} \in \mathbb{N}, \quad \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}$$

Vincoli:

$$\forall i \in \{1, \dots, n\} \quad \sum_{j=1}^m x_{ij} = 1 \quad (\text{normale vincolo di semi-assegnamento!!})$$

$$\forall j \in \{1, \dots, m\} \quad y \geq \sum_{i=1}^n x_{ij} t_i$$

Funzione obbiettivo:

$$\min \left( \underbrace{\max_j \sum_{i=1}^n \underbrace{x_{ij} t_i}_{\text{tempo di compilazione del progetto } i \text{ al pc } j}}_{\text{nonostante questa espressione matematica ha perfettamente senso, non è lineare!}} \right)$$

nonostante questa espressione matematica ha perfettamente senso, non è lineare!

Funzione obbiettivo sbagliata! Pertanto prenderemo  $y$  per "simulare" il massimo, di modo da rendere la funzione obbiettivo lineare:

$$\min y$$

### 2.3.2 Selezione di Sottoinsiemi

Sia  $N = \{1, \dots, n\}$  un insieme finito di elementi e sia  $F = \{F_1, \dots, F_n\}$  una famiglia di sottoinsiemi (non necessariamente disgiunti) dove  $F_i \subseteq N$  e sia  $c_j$  un costo che associamo ad ogni  $F_j$ . Si vuole determinare qual'è la scelta migliore di  $D \subseteq F$  di costo minimo fra tutti gli  $F_j \in F$ . Per formulare formalmente il problema, si può rappresentare l'appartenenza degli elementi di  $N$  in un sottoinsieme  $F_j$  come una matrice che ha un numero di  $n$  righe (numero di elementi) e un numero  $m$  di colonne (il numero di sottoinsieme). In dettaglio, sia  $A = (a_{ij} \in \{0, 1\}^{n \times m})$ , dove

$$a_{ij} = \begin{cases} 1 & \text{se } i \in F_j \\ 0 & \text{altrimenti} \end{cases}$$

Il vettore delle variabili  $x$  assumerà la forma  $x = (x_1, \dots, x_m)$  dove

$$x_j = \begin{cases} 1 & F_j \in D \\ 0 & \text{altrimenti} \end{cases}$$

Definendo  $D$  come  $D = \{F_j \mid x_j = 1\}$

La funzione obbiettivo è sempre:

$$\sum_{j=1}^m x_j c_j$$

I vincoli, invece, dipendono dal problema:

- **Problema di copertura:** ognuno degli elementi di  $N$  sta in almeno uno degli elementi di  $D$ :

$$\sum_{j=1}^m a_{ij} x_j \geq 1 \quad \forall i \in [1, n]$$

Ad esempio, i sottoinsiemi  $F$  di  $N$  possono essere dei curriculum dove c'è scritto quale dei linguaggi di programmazione  $l \in N$  conosce un candidato. Ogni candidato ha anche uno stipendio. Dobbiamo scegliere quali candidati assumere per coprire tutti i linguaggi in  $N$  minimizzando il costo

- **Partizione:** Ognuno degli elementi di  $N$  sta in *esattamente* uno degli elementi di  $D$ , ovvero  $D$  deve essere una partizione di  $N$ . Quindi:

$$\sum_{j=1}^m a_{ij} x_j = 1 \quad \forall i \in [1, n]$$

Ad esempio,  $N$  possono essere dei task da svolgere e gli insiemi di  $F$  sono offerte da fornitori rispetto alla risoluzione di alcuni dei task. Quindi gli insiemi di  $D$  devono essere disgiunti perché non vogliamo che due società risolvano lo stesso task. Tutto deve essere coperto una sola volta.

- **Riempimento:** si usa solo quando si vuole massimizzare il costo  $c_j$ .  $N$  non sono più incombenze o task, ma risorse da usare una volta per costruire un prodotto (elemento di  $F$ ). Al più perché possiamo scartare

qualche componente. Possiamo usare ogni elemento di  $N$  al massimo una volta. Quindi:

$$\sum_{j=1}^m a_{ij}x_j \leq 1 \quad \forall i \in [1, n]$$

### Example 2.3.2 (Problema delle Commesse)

#### Testo

Un'azienda deve decidere come impiegare i suoi  $n$  dipendenti  $1, \dots, n$ .

L'azienda, nell'intervallo di tempo desiderato deve evadere  $m$  commesse  $1, \dots, m$

Ciascuna commessa  $j$  deve essere svolta dal sottoinsieme  $D_j \subseteq \{1, \dots, n\}$  dei dipendenti dall'azienda.

Ogni commessa, se evasa darebbe luogo ad un ricavo pari a  $r_j$  euro.

Ogni dipendente può lavorare ad una singola commessa nell'unità di tempo

#### Svolgimento

Si consideri che questo è un Problema di selezione di sottoinsiemi, infatti:  $N = \{1, \dots, n\}$  elementi/dipendenti

$F = \{F_1, \dots, F_m\}$  dove  $F_j$  è la  $j$ -esima commessa che rappresentiamo con il sottoinsieme di dipendenti che la devono svolgere (quindi  $F_j = D_j$ ).

$$a_{ij} = \begin{cases} 1 & \text{se } i \in F_j \\ 0 & \text{altrimenti} \end{cases}$$

Variabili:

$$x_j = \begin{cases} 1 & \text{se la commessa } j \text{ è evasa} \\ 0 & \text{altrimenti} \end{cases}$$

$$x_j \in \mathbb{N} \quad y_j \in \mathbb{N}$$

Vincoli:

$$0 \leq x_j \leq 1 \quad \forall j \in \{1, \dots, m\} \quad 0 \leq y_j \leq 1$$

$$\forall i \in \{1, \dots, n\} \quad \sum_{j=1}^m a_{ij}x_j \leq 1 \quad y_j = 1 - x_j$$

Funzione obiettivo:

$$\max \sum_{j=1}^m r_j x_j - \sum_{j=1}^m y_j p_j$$

Nota: le scritte in rosso sono una possibile modifica all'esercizio dove per ogni commessa non fatta c'è una penalità  $p_j$ .

### 2.3.3 Variabili a valori discreti

Spesso, tuttavia, le variabili in gioco sono vincolate ad assumere un valore da un insieme che non è  $\{0, 1\}$  o un intervallo continuo (come  $\mathbb{N}$ ). Ad esempio si può essere interessati a vincolare  $x$  in un insieme finito di valori  $Z = \{z_1, \dots, z_k\}$  dove i vari  $z_i$  sono valori reali distinti

In questi casi, si possono utilizzare  $n$  variabili ausiliarie  $y_1, \dots, y_k$  che rappresentano l'appartenenza di  $z$  a  $Z$ . In particolare, si definiscono le variabili  $y_i$  come segue:

$$y_i = \begin{cases} 1 & \text{se } z = z_i \\ 0 & \text{altrimenti} \end{cases}$$

E sono vincolate come segue:

$$\sum_{i=1}^k y_i = 1 \quad x = \sum_{i=1}^k z_i y_i$$

### Example 2.3.3 (Progetto di reti)

#### TESTO

Si vogliono collegare le reti fognarie di alcuni centri residenziali e industriali ad un impianto di depurazione e smaltimento. Ogni centro produce una quantità nota di rifiuti (liquidi), ed è necessario dimensionare opportunamente l'impianto di collegamento al depuratore in modo da consentirne il trasporto. Si possono scegliere condotte di diversa portata e il costo di messa in opera di una condotta sarà una funzione della sua portata. Si consideri ad esempio il grafo  $G = (N, A)$  di figura 1.3(a) dove i nodi da 1 a 4 rappresentano i centri ed il nodo 5 rappresenta il depuratore. Accanto ad ogni nodo è indicata la quantità di liquido da depurare per unità di tempo prodotta dal centro ad esso corrispondente. Ogni arco  $(i, j) \in A$  rappresenta un possibile collegamento; fra di essi andranno scelti quelli che verranno effettivamente costruiti, e per ciascuno dei collegamenti costruiti si dovrà determinare la portata. Osserviamo che i collegamenti hanno una direzione prefissata di percorrenza. Una soluzione ammissibile è data da un insieme di collegamenti che garantiscano il fluire dei liquidi da tutti i centri fino al depuratore. In figura 1.3(b) è rappresentata una possibile soluzione.



**SVOLGIMENTO TODO:** svolgi

### 2.3.4 Minima quantità positiva

Quando una variabile  $x$  rappresenta un certo livello di produzione, possibile che il valore della variabile sia

$$x \in \{0\} \cup [l, u]$$

Dove:

- $l$  è il livello minimo di produzione
- $u$  è il livello massimo di produzione
- $0$  è l'assenza di produzione

Per modellare correttamente tale situazione si introduce una variabile ausiliaria logica  $y \in \{0, 1\}$  che indica la presenza o l'assenza di produzione, cosicché i vincoli siano:

$$x \geq ly \quad x \leq uy$$

Infatti se  $y = 0$  allora  $x = 0$  e se  $y = 1$  allora  $x \in [l, u]$

### 2.3.5 Funzione a carico fisso

#### Definition 2.3.3: Funzione a carico fisso

Una funzione  $f(x)$  è detta a carico fisso se:

$$f(x) = \begin{cases} b + cx & \text{se } 0 < x \leq u \\ 0 & \text{se } x = 0 \end{cases} \quad (2.1)$$

Dove  $b > 0$  e  $u$  è un opportuno reale positivo



Si noti il seguente grafico rappresentativo: TODO: inserire il grafico

Questo tipo di funzione è molto utile per modellare situazioni in cui si ha un costo di investimento fisso  $b$  per l'attivazione di un servizio, e un costo variabile  $c$  (coefficiente angolare) per l'unità di prodotto, mentre il costo è nullo se il servizio non è attivato ( $x = 0 \implies f(x) = 0$ )

Per rappresentare, analiticamente, questa funzione si introduce, al solito, una variabile ausiliaria logica  $y \in \{0, 1\}$  che indica l'attivazione del servizio, cosicché i vincoli siano:

$$0 \leq x \leq uy$$

Adesso è possibile sostituire la funzione a carico fisso tramite una nuova funzione  $g : \mathbb{R}^2 \rightarrow \mathbb{R}$  definita come segue:

$$g(x, y) = b(y) + cx$$

Si ha infatti che

$$g(0, 0) = 0 \quad g(x, 1) = b + cx$$

**Note:**

Si noti che se  $y = 0$  allora, per i vincoli introdotti poc'hanzi, si ha che  $x = 0$  (anche se non vale viceversa). In altre parole i vincoli introdotti non escludono la possibilità che sia contemporaneamente  $y = 1$  e  $x = 0$ , in questo caso avremo  $g(0, 1) \neq f(0)$  (nasty). Di conseguenza la funzione  $g$  non rappresenta perfettamente ed in modo univoco la funzione  $f(x)$ , ma dato che l'obiettivo è *minimizzare* la funzione  $f(x)$ , la soluzione  $(x, y) = (0, 1)$  viene esclusa essendo  $g(0, 1) = b > g(0, 0) = 0 = f(0)$

**Example 2.3.4** (Ordinamento di valori su macchine)

TODO: aggiungere questo bellissimo esempio

### 2.3.6 Rappresentazioni del valore assoluto

#### Vincoli di valore assoluto

Si può avere a che fare con un **vincolo** a "valore assoluto" del tipo:

$$|g(x)| \leq b$$

Dove  $g(x)$  è una funzione dell'insieme di variabili  $x$  e  $b$  è un valore reale positivo, in questo caso tale vincolo è espresso come la congiunzione di due vincoli:

$$g(x) \leq b \quad -g(x) \leq b$$

Dato che se  $g(x) \leq b$  il vincolo  $-g(x) \geq -b$  e viceversa. Inoltre, se  $g(x)$  è una funzione lineare, allora tale vincolo può essere espresso come un vincolo lineare.

#### Funzioni obiettivo a valore assoluto

Vediamo ora come trattare una funzione obiettivo espressa mediante un valore assoluto. Si consideri una funzione obiettivo  $|f(x)|$  da dover *massimizzare* con  $x \in X$ . È sufficiente risolvere i due diversi problemi:

$$\max\{f(x) \mid x \in X\} \quad \max\{-f(x) \mid x \in X\}$$

E scegliere come soluzione quella che fornisce il valore più alto della funzione obiettivo

Analiticamente, per i problemi di minimizzazione

### 2.3.7 Funzioni definite a tratti

Una funzione che è possibile riscontrare nella vita reale è il caso delle funzioni lineari a tratti

Si consideri la seguente funzione

$$f(x) = \begin{cases} b_1 + c_1x & x \in [a_1, a_2] \\ b_2 + c_2x & x \in (a_2, a_3] \end{cases} \quad (2.2)$$

Dove assumiamo

$$b_2 + c_2 a_2 \geq b_1 + c_1 a_2$$

Si noti il seguente grafico:

TODO: INSERIRE GRAFICO

Come per il carico fisso vengono introdotte due variabili logiche ausiliarie  $y_1, y_2$  con seguente significato:

$$y_1 = \begin{cases} 1 & \text{se } x \in [a_1, a_2] \\ 0 & \text{altrimenti} \end{cases} \quad y_2 = \begin{cases} 1 & \text{se } x \in (a_2, a_3] \\ 0 & \text{altrimenti} \end{cases} \quad (2.3)$$

Dovendo  $x$  appartenere ad uno ed uno solo degli intervalli, si ha il seguente vincolo

$$y_1 + y_2 = 1$$

Inoltre introduciamo variabili quantitative ausiliarie che ci dicono lo spostamento rispetto a un estremo dell'intervallo a cui appartiene  $x$ :

$$z_1 = \begin{cases} x - a_1 & \text{se } x \in [a_1, a_2] \\ 0 & \text{altrimenti} \end{cases} \quad z_2 = \begin{cases} x - a_2 & \text{se } x \in (a_2, a_3] \\ 0 & \text{altrimenti} \end{cases} \quad (2.4)$$

Infatti se  $x \in [a_1, a_2]$ , il valore  $z_1 (= x - a_1)$  non è che la porzione di valore di  $x$  che "supera"  $a_1$ , pertanto il suo valore è limitato dalle disuguaglianze  $0 \leq z_1 \leq a_2 - a_1$ , si ha quindi che un'altro vincolo:

$$0 \leq (z_1 \leq a_2 - a_1)y_1$$

Con  $y_1$  variabile definita prima utile a controllare se effettivamente  $x \in [a_1, a_2]$ . Analogamente è il caso di  $z_2$ , che ci conduce al seguente vincolo:

$$0 \leq z_2 \leq (a_3 - a_2)y_2$$

Riassumendo tutti i vincoli ottenuti, si ha:

$$\begin{aligned} y_1, y_2 &\in \{0, 1\} \\ y_1 + y_2 &= 1 \\ 0 &\leq z_1 \leq (a_2 - a_1)y_1 \\ 0 &\leq z_2 \leq (a_3 - a_2)y_2 \end{aligned} \quad (2.5)$$

**Note:**

Si noti che i vincoli ?? impongono che solo una delle due variabili  $z_1$  e  $z_2$  possa avere valore non nullo

## Rappresentazione di $f$ con funzione multivariabile

Poi il basta mi spiegherà la seguente quote

Non c'è niente che ci dà la funzione da ottimizzare, abbiamo solo delle definizioni che sono dei semplici commenti, i vincoli lineari sono solo gli ultimi, ma non bastano — ☹

Vabbuò, adesso è possibile sostituire alla funzione  $f(x)$  la seguente funzione  $g : \mathbb{R}^4 \rightarrow \mathbb{R}$ :

$$\begin{aligned} g(z_1, z_2, y_1, y_2) &= b_1 y_1 + c_1(a_1 y_1 + z_1) + b_2 y_2 + c_2(a_2 y_2 + z_2) \\ &= y_1(b_1 + c_1 a_1) + c_1 z_1 + y_2(b_2 + c_2 a_2) + c_2 z_2 \end{aligned}$$

Con le variabili soggette ai vincoli ?? In questo modo è stato possibile rappresentare la funzione  $f(x)$  (non lineare) come una funzione lineare a variabili continue definita su un opportuno insieme.

Si ha che il valore di  $f$  è rappresentato *univocamente* dalla quadrupla  $(z_1, z_2, y_1, y_2)$ , eccetto nel punto di non continuità  $a_2$  che può essere espresso con le due seguenti quadruple:

$$(a_2 - a_1, 0, 1, 0) \quad (0, 0, 0, 1)$$

Dei quali solo **primo** è accettabile per i vincoli dati. Se, però, supponiamo il problema sia un problema di *minimo*, possiamo considerare il problema come benigno, dato che:

$$g(0, 0, 0, 1) = b_2 + c_2 a_2 \geq b_1 + c_1 a_2 = g(a_2 - a_1, 0, 1, 0)$$

## Chapter 3

# Reti di Flusso

Innanzitutto iniziamo con alcune definizioni:

### Definition 3.0.1: Rete

definiamo **rete** un grafo *pesato*  $G = (N, A)$  dove

- $N$  è l'insieme dei nodi
- $A$  è l'insieme degli archi

Più precisamente:

- $\forall i \in N$  è associato un valore reale  $b_i$ , detto *sbilanciamento* che può essere:
  - **Positivo**: il nodo  $i$  è detto *pozzo* e rappresenta la quantità del bene che esce dalla rete al nodo  $i$ , mentre  $b_i$  è detto *domanda del bene*
  - **Negativo**: il nodo  $i$  è detto *sorgente* e rappresenta la quantità del bene che entra nella rete al nodo  $i$ , mentre  $-b_i$  è detto *offerta del bene*
  - **Nulla**: il nodo  $i$  è detto *transito* e rappresenta un nodo di passaggio
- $\forall (i, j) \in A$  è associato un valore reale  $c_{ij}$  detto *costo* che rappresenta il costo di trasportare una unità di bene lungo l'arco  $(i, j)$  ed una coppia di valori  $l_{ij}$  e  $u_{ij}$  detti *capacità inferiore* e *capacità superiore* rispettivamente, che rappresentano il minimo e il massimo numero di unità di bene che possono essere trasportate lungo l'arco  $(i, j)$



### 3.1 Problemi di Flusso

#### Definition 3.1.1: Problemi di flusso

Si definiscono *problemi di flusso* quei problemi che consistono nel determinare, lungo una rete, il flusso di un bene, ossia un assegnamento a ciascun arco  $(i, j) \in A$  un valore reale  $x_{ij} \in [l_{ij}, u_{ij}]$ , rispettando i vincoli di capacità e conservazione del flusso

**flusso**, quindi, è proprio la soluzione che vogliamo trovare. Tuttavia ad ogni unità di flusso assegnata ad un arco corrisponde un costo  $c_{ij}$ , si ha quindi che il costo complessivo del flusso è dato da:

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \quad (3.1)$$

### 3.1.1 Vincoli

I vincoli che nei problemi di flusso si deve rispettare sono:

- *Uguaglianza di domanda e offerta globale:*

$$\sum_{i \in D} b_i = - \sum_{i \in O} b_i \iff \sum_{i \in N} b_i = 0 \quad (3.2)$$

Dove  $D = \{b_i \in N \mid b_i > 0\}$  e  $O = \{b_i \in N \mid b_i < 0\}$ . Per quelli nulli, tanto vale non metterli da nessuna parte per non creare asimmetria inutile.

- *Conservazione di flusso:*

lol

Il Basta

$$\sum_{(i,j) \in BS(i)} x_{ij} - \sum_{(i,j) \in FS(i)} x_{ij} = b_i \quad \forall i \in N \quad (3.3)$$

dove

$BS(i) = \{(k, i) \mid (k, i) \in A\}$  detto stella entrante

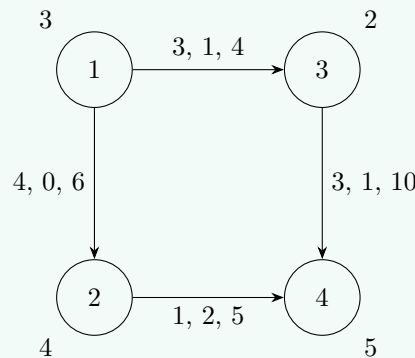
$FS(i) = \{(i, k) \mid (i, k) \in A\}$  detto stella uscente

Ovvero ciò che entra nel nodo e' uguale a cio' che esce piu' lo sbilanciamento.

- *Ammissibilità del flusso:*

$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

#### Example 3.1.1 (Esempio di rete)



Perche' mai dovremmo studiare sta roba?

- **Espressivita'**: permette di catturare un range di problemi concreti abbastanza grande
- **Complessita'**: esistono algoritmi che risolvono questo tipo di problemi con complessita' polinomiale abbastanza basso. A Ugo questo fa molto piacere :), perché ricordiamolo NON SIAMO INGEGNERI GESTIONALI, SIAMO INFORMATICI

### 3.1.2 Ipotesi semplificative

Le ipotesi semplificative sono supposizioni che non consentono di usare il caso generale, ma che semplificano il problema **senza** perdere espressivita', come ad esempio nel caso in cui le capacita' inferiori sono nulle ovvero  $l_{ij} = 0$ , che e' una situazione a cui possiamo arrivare **anche da grafi che non hanno questa caratteristica**. Infatti data una rete  $G$ , si può costruire una rete  $G'$  tale che  $G$  e  $G'$  hanno lo stesso flusso ottimo, ma  $G'$  ha capacita' inferiori nulle.  $\forall (i, j) \in A$ :

- Si sottrae  $l_{ij}$  a  $b_j$  e a  $u_{ij}$
- Si aggiunge  $l_{ij}$  a  $b_i$
- Occorre aggiungere la quantità

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

Alla funzione obiettivo

**Note:**

Si noti che non cambia la soluzione ottima, ma solo il valore ottimo della funzione obiettivo

È pertanto vero, quindi, che ad un flusso  $x_{ij} \in G$  corrisponde un flusso  $(x_{ij} + l_{ij}) \in G'$

### 3.1.3 Problema del Flusso di Costo Minimo

**Definition 3.1.2: Problema del Flusso di Costo Minimo**

Si definisce *problema del flusso di costo minimo* il problema di flusso in cui la funzione obiettivo è il costo di flusso da minimizzare e le cui capacità inferiori sono nulle

Questo problema è formalizzabile in programmazione lineare come segue:

$$\begin{aligned} \min & cx \\ Ex = b & \leq x \leq u \end{aligned} \quad (3.4)$$

dove

- $x \in \mathbb{R}^{|A|}$  è il vettore delle variabili di flusso
- $c \in \mathbb{R}^{|A|}$  è il vettore dei costi
- $E \in \mathbb{R}^{|N| \times |A|}$  è una matrice di incidenza fra nodi e archi i cui elementi possono solo assumere valori 0, -1 e 1
- $b \in \mathbb{R}^{|N|}$  è il vettore degli sbilanciamenti
- $u \in \mathbb{R}^{|A|}$  è il vettore delle capacità superiori

In questo modo abbiamo scritto funzione obiettivo e vincoli in forme matriciali molto semplici, adesso, senza terrorizzare nessuno, fornirò la formalizzazione in forma estesa:

$$\begin{aligned} \min & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ & \sum_{(j,i) \in BS(i)} x_{ji} - \sum_{(i,j) \in A} x_{ij} = b_i \quad \forall i \in N \\ & l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A \end{aligned} \quad (3.5)$$

#### Rilassamento di assunzioni

Può essere utile assumere che esiste un solo pozzo e una sola sorgente, che vengono poi collegati con archi fittizi (a costo nullo e capacità pari allo sbilanciamento dei pozzi/sorgenti effettivi!) a quelli effettivi.

Non si può, però, enunciare che i nodi non hanno una capacità, perché di fatto non succede nulla, perché può essere trasformata in una rete equivalente dove i nodi non ce l'hanno:

*TODO: spiegare meglio, che cazzo vuoi dire abbastianini?*

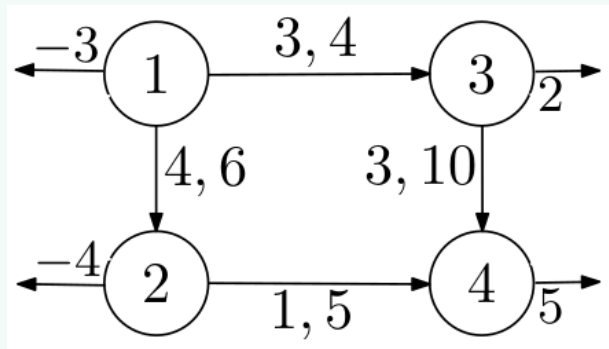
Mi spiego in modo più eloquente: Per poter utilizzare nodi che hanno una propria capacità, non è necessario perdere generalità cambiando la definizione del problema esaminato. Questo è vero, in quanto una tale rete può essere banalmente trasformata in una rete equivalente la quale presenta solo nodi con capacità nulla. Come accade tale trasformazione? Boh (c'è scritto dopo vedi dopo)

Per trasformare un generico problema MCF in un problema con una sola sorgente ed un solo pozzo...

- Si aggiungono due nodi fittizi  $s$  e  $t$ , uno sorgente e uno pozzo
- Si aggiungono archi fittizi da  $s$  ai nodi sorgenti di partenza, con un costo nullo e capacità pari all'inverso dello sbilanciamento del nodo sorgente, ovvero  $-b_s$
- Si aggiungono archi fittizi dai nodi pozzo ai nodi pozzo di arrivo, con un costo nullo e capacità pari allo sbilanciamento del nodo pozzo
- Lo sbilanciamento di  $s$  e' pari alla somma degli sbilanciamenti dei nodi sorgenti, mentre lo sbilanciamento di  $t$  e' pari alla somma degli sbilanciamenti dei nodi pozzo

### Example 3.1.2 (Esempio di Rilassamento)

Si parte da questa rete:

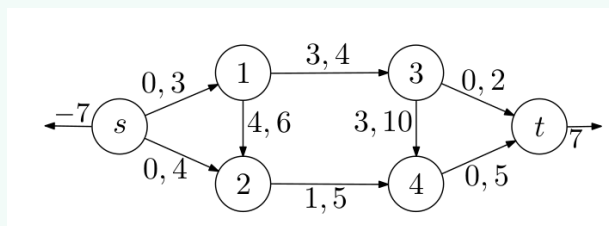


Si aggiungono i nodi fittizi  $s$  e  $t$  con sbilanciamento rispettivamente  $b_s = b_1 + b_2 = -3 - 4 = -7$  e  $b_t = b_3 + b_4 = 2 + 5 = 7$

Si aggiungono gli archi fittizi con costo nullo e con capacità:

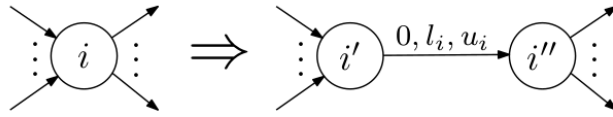
- $a_{s1} = -b_1 = 3$
- $a_{s2} = -b_2 = 4$
- $a_{3t} = b_3 = 2$
- $a_{4t} = b_4 = 5$

Si ottiene quindi la seguente rete rilassata:



Alle volte, inoltre, è utile che anche i nodi abbiano delle **capacità**, ossia che solo una quantità di flusso compresa nell'intervallo chiuso  $[l_i, u_i]$  possa passare per il nodo  $i \in N$ . Per fare ciò occorre *sdoppiare* ciascun nodo  $i$  in due nodi  $i', i''$ , in modo che:

- Tutti gli archi entranti in  $i$  entrino in  $i'$
- tutti gli archi uscenti da  $i$  escano da  $i''$
- Si aggiunge un arco da  $i'$  a  $i''$  con capacità  $[l_i, u_i]$



### 3.1.4 Problema di Flusso Massimo

#### Definition 3.1.3: Problema di Flusso Massimo

Si definisce *problema di flusso massimo* il problema di flusso in cui la funzione obiettivo è il flusso da massimizzare e le cui capacità inferiori sono nulle

Ciò che cambia, quindi, dal problema di flusso di costo minimo è la funzione obiettivo, che diventa, si vuole infatti *massimizzare* il flusso, e non minimizzare il costo.

Formalmente, data una rete  $G = (N, A)$ :

- Si fissano due nodi  $s$  e  $t$  rispettivamente sorgente e pozzo
- Si vuole trovare il massimo valore  $v$  tale che se  $b_s = -v$ ,  $b_t = v$  e  $b_i = 0$  allora esiste un flusso ammissibile  $x$  tale che  $Ex = b$

Formalizzato in programmazione lineare, il problema di flusso massimo diventa:

$$\begin{aligned}
 & \max \quad v \\
 & \sum_{(j,s) \in BS(s)} x_{js} + v = \sum_{(s,j) \in FS(s)} x_{sj}; \\
 & \sum_{(j,i) \in BS(i)} x_{ji} - \sum_{(i,j) \in FS(i)} x_{ij} = 0, \quad i \in N - \{s, t\}; \\
 & \sum_{(j,t) \in BS(t)} x_{jt} = \sum_{(t,j) \in FS(t)} x_{tj} + v; \\
 & 0 \leq x_{ij} \leq u_{ij}, \quad (i, j) \in A.
 \end{aligned} \tag{3.6}$$

#### Note:

Si noti che il problema di flusso massimo è un caso particolare di problema di flusso di costo minimo, di fatti partendo da un problema di flusso massimo si può costruire un problema di flusso di costo minimo:

- Si nullificano i costi
- Si pone  $l_{ij} = 0$  e  $u_{ij} = \infty$

## 3.2 Problema del flusso massimo: algoritmi

### 3.2.1 Tagli

#### Definition 3.2.1: Taglio

Si definisce **taglio** in una rete  $G = (N, A)$  una coppia  $(N', N'')$  di sottoinsiemi di  $N$  tali che  $N' \cup N'' = N$  e  $N' \cap N'' = \emptyset$

Si noti, inoltre anche questa definizione

#### Definition 3.2.2: $(s, t)$ -taglio

Si definisce  $(s, t)$ -**taglio** in una rete  $G = (N, A)$  un taglio  $(N_s, N_t)$  dove  $s \in N_s$  e  $t \in N_t$

Si prendi in considerazione inoltre i confini il confine fra gli  $(s, t)$  - tagli, dove:

- $A^+(N_s, N_t) = \{(i, j) \in A \mid i \in N_s \wedge j \in N_t\}$  sono gli archi che vanno dalla partizione  $s$  a  $t$
- $A^-(N_s, N_t) = \{(i, j) \in A \mid i \in N_t \wedge j \in N_s\}$  sono gli archi che vanno da  $t$  a  $s$

Adesso introduciamo un importante lemma sui tagli:

### Lemma 3.2.1

$\forall (s, t)$  – taglio  $(N_s, N_t)$  e  $\forall$  flusso ammissibile  $x$  con valore  $v$ , si ha che:

- $v = \sum_{(i,j) \in A^+(N_s, N_t)} x_{ij} - \sum_{(i,j) \in A^-(N_s, N_t)} x_{ji}$   
Si ha, quindi, che  $v$  la somma del flusso uscente da  $s$  meno il flusso entrante in  $s$ . Questa quantità è detta **flusso del taglio**  $(N_s, N_t)$  e la si denota  $x(N_s, N_t)$
- $v \leq \sum_{(i,j) \in A^+(N_s, N_t)} u_{ij}$   
Ovvero Il flusso massimo è sempre minore o uguale alla capacità totale del taglio. Questa quantità è detta **capacità del taglio** ed è indicata con  $u(N_s, N_t)$

**Dimostrazione:** • Dimostro la prima parte: Per definizione di  $v$ , si ha che:

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

Questo valore è uguale alla somma netta in uscita  $\forall i \in N_s$  verso la partizione  $N_t$ , quindi:

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(i,j) \in A} x_{ji} = \sum_{k \in N_s} \left( \sum_{(k,j) \in A} x_{kj} - \sum_{(i,k) \in A} x_{ik} \right)$$

E questa somma è ovviamente uguale al flusso uscente dal confine  $A^+(N_s, N_t)$  meno il flusso entrante dal confine  $A^-(N_s, N_t)$ , quindi:

$$\sum_{k \in N_s} \left( \sum_{(k,j) \in A} x_{kj} - \sum_{(i,k) \in A} x_{ik} \right) = \sum_{(i,j) \in A^+(N_s, N_t)} x_{ij} - \sum_{(i,j) \in A^-(N_s, N_t)} x_{ij}$$

Per riassumere quindi:

$$v = \sum_{(i,j) \in A^+(N_s, N_t)} x_{ij} - \sum_{(i,j) \in A^-(N_s, N_t)} x_{ij} \quad (3.7)$$

- Dimostro la seconda parte:

Ovvio perché è una semplice derivazione del punto 1



#### Note:

si noti che la conclusione di questo Lemma è

$$v = x(N_s, N_t) \leq u(N_s, N_t)$$

Ovvero *il valore di un flusso ammissibile è sempre minore o uguale della capacità di qualunque taglio*

Adesso tratteremo il caso in cui si voglia determinare se esiste un taglio con capacità **identica** al valore di un flusso ammissibile (ovvero massimo)

## 3.2.2 Algoritmo di Ford-Fulkerson

Innanzitutto dobbiamo presentare alcuni concetti



## Grafi residui

### Definition 3.2.3: grafi residui

Dati una rete  $G = (N_G, A_G)$  ed un flusso ammissibile  $x$  si definisce il **grafo residuo**  $G_x$  il multigrafo  $(N_{G_x}, A_{G_x})$  tale che:

- $N_{G_x} = N_G$
- Gli archi in  $A_{G_x}$  sono di due tipi:
  - $\forall (i, j) \in A_G$  tale che  $x_{ij} < u_{ij}$  esiste un arco da  $i$  a  $j$  in  $G_x$  detto **arco concorde**  
Significa che il flusso non ha saturato la capacità dell'arco e si può ancora inviare flusso in quella direzione
  - $\forall (i, j) \in A_G$  tale che  $x_{ij} > 0$  esiste un arco da  $j$  a  $i$  in  $G_x$  detto **arco discorde**

## Cammino aumentante

### Definition 3.2.4: Cammino aumentante

Un **cammino aumentante**  $P$  in una rete  $G$  rispetto a  $x$  è un cammino semplice e orientato da  $x$  a  $t$

Inoltre introduciamo anche alcune notazioni

#### Note:

Denotiamo con  $P^+$  l'insieme degli archi *concordi* in  $P$ , e  $P^-$  l'insieme degli archi *discordi*

Sia poi la capacità di un cammino aumentante

### Definition 3.2.5: Capacità di un cammino aumentante

Dato un cammino aumentante  $P$  in una rete  $G$  rispetto a un flusso  $x$ , la **capacità** di  $P$  rispetto a  $x$  è definita come:

$$\theta(P, x) = \min \{ \min \{ u_{ij} - x_{ij} \mid (i, j) \in P^+ \}, \min \{ x_{ij} \mid (j, i) \in P^- \} \} \quad (3.8)$$

Da cui discende la definizione di flusso  $x(P, \theta)$ :

### Definition 3.2.6: Flusso $x(P, \theta)$

Dato un cammino aumentante  $P$  in una rete  $G$  rispetto a un flusso  $x$  e una capacità  $\theta$ , il flusso  $x(P, \theta)$  è definito come:

$$(x(P, \theta))_{ij} = \begin{cases} x_{ij} + \theta & (i, j) \in P^+ \\ x_{ij} - \theta & (i, j) \in P^- \\ x_{ij} & \text{altrimenti} \end{cases} \quad (3.9)$$

## Algoritmo Ford-Fulkerson

TODO: cambiare il titolo della sottosotto sezione

Miglioriamo il flusso corrente usando il grafo residuo con cammino aumentante. Quindi visitiamo il grafo residuo partendo da  $s$ , e se c'è un cammino che arriva a  $t$  significa che è aumentante, quindi si aggiorna  $x$  il meglio possibile. fare questo finché non si trova il cammino aumentante, quindi abbiamo ottimizzato :)

---

**Algorithm 1** Algoritmo di Ford-Fulkerson

---

**Input:** Grafo  $G$  con sorgente  $s$  e sink  $t$ **Output:** Flusso massimo  $x$ 

```
1  $x \leftarrow 0$ 
2 while True do
3    $\text{Make}(G_x)$ ; // Costruisci il grafo residuo  $G_x$ 
4   if  $\exists$  un cammino aumentante  $P \in G_x$  then
5      $x \leftarrow x(P, \theta(P, x))$ 
6   else
7     return  $x$ 
```

---

**Correttezza****Lemma 3.2.2**

Se  $x$  e' ammissibile, allora anche  $x(P, \delta(P, x))$  e' ammissibile

**Lemma 3.2.3**

Se  $x$  e' un flusso ammissibile massimo, allora  $G_x$  non ha cammini aumentanti.

: Uso l'eliminazione del not e ipotizzo che  $G_x$  abbia cammini aumentanti. Ma in questo caso  $x$  non sarebbe massimo dato che sarebbe possibile aumentare la capacita' del flusso. Quindi dimostrato per il falso.  $\square$

**Lemma 3.2.4**

Se  $G_x$  non ha cammini aumentanti, allora esiste un taglio di capacita' pari a  $v$ .

# Chapter 4

## Esercitazioni

### 4.1 Problemi e modelli

#### 4.1.1 Problema dei data center

##### Testo

Abbiamo  $n$  server  $1, \dots, n$ , ogni server può lavorare in  $m_i \in \mathbb{N}$  modalità operative diverse. Nella modalità  $j \in \{1, \dots, m_i\}$ , il server  $i$  riesce ad eseguire un numero di istruzioni  $s_{ij}$ .

##### Svolgimento

*Variabili:*

$$x_{ij} = \begin{cases} 1 & \text{se il server } i \text{ è utilizzato in modalità } j \in \{1, \dots, m_i\} \\ 0 & \text{altrimenti} \end{cases}$$
$$x_{ij} \in \mathbb{N}, \quad \forall i \forall j$$

*Vincoli:*  $\forall i \in \{1, \dots, n\}, \quad \forall j \in \{1, \dots, m_i\}$  con  $0 \leq x_{ij} \leq 1$  si ha:

$$\forall i \in \{1, \dots, n\} \quad \sum_{j=1}^{m_i} x_{ij} = 1$$
$$\sum_{i=1}^n \sum_{j=1}^{m_i} x_{ij} s_{ij} \geq k$$

*funzione obiettivo*

$$\min \sum_{i=1}^n \sum_{j=1}^{m_i} x_{ij} w_{ij}$$

#### 4.1.2 Problema del docente di informatica

##### Testo

Si hanno dei progetti  $(t_1, t_2, \dots, t_n)$ , si hanno  $m$  PC, dove ogni pc può compilare qualunque progetto in modalità sequenziale. Le prestazioni del PC sono identiche

Vogliamo assegnare i progetti ai pc in modo da minimizzare il tempo complessivo parallelo di compilazione

## Svolgimento

*Variabili:*

$$x_{ij} = \begin{cases} 1 & \text{se il progetto } i \text{ è compilato al pc } j \\ 0 & \text{altrimenti} \end{cases}$$
$$x_{ij} \in \mathbb{N}, \quad \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}$$

*Vincoli:*

$$\forall i \in \{1, \dots, n\} \quad \sum_{j=1}^m x_{ij} = 1 \quad (\text{normale vincolo di semi-assegnamento!!})$$
$$\forall j \in \{1, \dots, m\} \quad y \geq \sum_{i=1}^n x_{ij} t_i$$

*Funzione obiettivo:*

$$\min \left( \max_j \sum_{i=1}^n \underbrace{x_{ij} t_i}_{\text{tempo di compilazione del progetto } i \text{ al pc } j} \right)$$

nonostante questa espressione matematica ha perfettamente senso, non è lineare!

Funzione obiettivo sbagliata! Pertanto prenderemo  $y$  per "simulare" il massimo, di modo da rendere la funzione obiettivo lineare:

$$\min y$$

### 4.1.3 Problema delle commesse

#### Testo

Un'azienda deve decidere come impiegare i suoi  $n$  dipendenti  $1, \dots, n$ .

L'azienda, nell'intervallo di tempo desiderato deve evadere  $m$  commesse  $1, \dots, m$

Ciascuna commessa  $j$  deve essere svolta dal sottoinsieme  $D_j \subseteq \{1, \dots, n\}$  dei dipendenti dall'azienda.

Ogni commessa, se evasa darebbe luogo ad un ricavo pari a  $r_j$  euro.

Ogni dipendente può lavorare ad una singola commessa nell'unità di tempo

#### Svolgimento

Si consideri che questo è un Problema di selezione di sottoinsiemi, infatti:  $N = \{1, \dots, n\}$  elementi/dipendenti  
 $F = \{F_1, \dots, F_m\}$  dove  $F_j$  è la  $j$ -esima commessa

$$a_{ij} = \begin{cases} 1 & \text{se } i \in F_j \\ 0 & \text{altrimenti} \end{cases}$$

*Variabili:*

$$x_j = \begin{cases} 1 & \text{se la commessa } j \text{ è evasa} \\ 0 & \text{altrimenti} \end{cases}$$
$$x_j \in \mathbb{N} \quad y_j \in \mathbb{N}$$

*Vincoli:*

$$0 \leq x_j \leq 1 \quad \forall j \in \{1, \dots, m\} \quad 0 \leq y_j \leq 1$$
$$\forall i \in \{1, \dots, n\} \quad \sum_{j=1}^m a_{ij} x_j \leq 1 \quad y_j = 1 - x_j$$

*Funzione obiettivo:*

$$\max \sum_{j=1}^m r_j x_j - \sum_{j=1}^m y_j p_j$$

#### 4.1.4 Problema della minimizzazione del massimo

##### Testo

Una ditta di costruzioni edilizie ha deciso di subappaltare  $n$  diverse opere ad  $n$  diversi artigiani. Ad ogni artigiano  $i = 1, \dots, n$  chiede di fornire il costo preventivo  $c_{ij}$  che richiede per effettuare l'opera  $j$ , per ogni  $j = 1, \dots, n$ . Si vuole assegnare un'opera a ciascun artigiano in modo che tutte le opere siano effettuate e il costo massimo dei subappalti assegnati sia minimizzato. Formulare il problema

##### Svolgimento

*Variabili:*

$$x_{ij} = \begin{cases} 1 & \text{se l'opera } j \text{ è assegnata all'artigiano } i \\ 0 & \text{altrimenti} \end{cases}$$

*Vincoli:*

$$\forall j \sum_{i=1}^n x_{ij} = 1 \quad \forall i \sum_{j=1}^n x_{ij} = 1$$

Inoltre:

$$\forall i, j \quad c_{ij} x_{ij} \leq M$$

*Funzione obiettivo:*

$$\min M$$

#### 4.1.5 Problema della massimizzazione del minimo

##### Testo

Si provi ora a massimizzare il costo minimo dei subappalti assegnati

##### Svolgimento

*Variabili:*

$$x_{ij} = \begin{cases} 1 & \text{se l'opera } j \text{ è assegnata all'artigiano } i \\ 0 & \text{altrimenti} \end{cases}$$

*Vincoli:*

$$\forall j \sum_{i=1}^n x_{ij} = 1 \quad \forall i \sum_{j=1}^n x_{ij} = 1$$

Inoltre:

$$\forall i, j \quad c_{ij} x_{ij} \geq m$$

*Funzione obiettivo:*

$$\max m$$

#### 4.1.6 Esercizio valore assoluto

##### testo

Formulare il problema  $\min\{|3 - 4x| : |x| \leq 2\}$

##### Svolgimento

*Vincoli*

$$x \leq 2 \quad -x \leq 2$$

*Funzione obiettivo*

$$\min\{3 - 4x\} \quad \min\{3 - 4x\}$$

#### 4.1.7 Esercizio 1.26

*Variabili*

$x_1, x_2$  = quantita di V1, V2

$y_1, y_2, y_3$  = quantita di N1, N2, N3

$z$  = totale degli oli acquistati

*Vincoli*

$$0 \leq x_1 + x_2 \leq 200$$

$$0 \leq y_1 + y_2 + y_3 \leq 250$$

$$z \leq d_1 x_1 + d_2 x_2 + d_3 y_1 + d_4 y_2 + d_5 y_3 \leq z_6$$

$z = x_1 + x_2 + y_1 + y_2 + y_3$  Dobbiamo fare in modo che  $z$  sia effettivamente il valore che gli vogliamo dare

*Funzione Obiettivo*

$$\max 150z - (110x_1 + 120x_2 + 130y_1 + 110y_2 + 115y_3)$$

#### 4.1.8 Esercizio 1.27

*Variabili*

$x_1, x_2, x_3$  = numero di dolci A, B o C

$x_1, x_2, x_3 \in \mathbb{N}$

*Vincoli*

$$x_1 + x_2 + x_3 \leq 10000$$

$$x_1 \leq 0.5x_1 + 0.5x_2 + 0.5x_3$$

$$x_3 \leq 0.25x_2$$

*Funzione Obiettivo*

$$0.2x_1 + 0.1x_2 + 0.4x_3$$

#### 4.1.9 Esercizio 1.28

*Variabili*

$x_1, x_2, x_3$  = numero di beni prodotti con  $P_1, P_2, P_3 \in \mathbb{N}$

*Vincoli*

$$2x_1 + x_2 + 3x_3 \leq 50$$

$$4x_1 + 2x_2 + 3x_3 \leq 50$$

$$3x_1 + 4x_2 + 2x_3 \leq 50$$

$$15x_1 + 18x_2 + 10x_3 \geq 200$$

*Funzione Obiettivo*

$$\min 4x_1 + 2x_2 + 3x_3$$

#### 4.1.10 Esercizio 1.29

*Variabili*

$x_1, x_2, x_3, x_4, x_5, x_6$  = numero ostetriche nuove a turno  $\in \mathbb{N}$

*Vincoli*

$$x_1 \geq 70$$

$$x_1 + x_2 \geq 80$$

$$x_2 + x_3 \geq 50$$

$$x_3 + x_4 \geq 60$$

$$x_4 + x_5 \geq 40$$

$$x_5 + x_6 \geq 30$$

*Funzione Obiettivo*

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

#### 4.1.11 Esercizio 1.31

*Variabili*

$\forall i = 1, \dots, 7$   $x_i$  = variabili logiche che indicano il fatto che un certo abitante  $a_i$  sia o meno membro del consiglio

*Vincoli*

$$\sum_{i=1}^7 x_i = 4$$

$$\forall i = 1, \dots, 4 \sum_{x \in C_i} x = 1$$

#### 4.1.12 Esercizio 1.32

*Variabili*

$x_i$  = variabili logiche che mi dicono se scelgo i

*Funzione Obiettivo*

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

*Vincoli*

$$\sum_{i=1}^n x_i = 11$$

$$1 \leq \sum_{i \in C} x_i \leq 5$$

$$1 \leq \sum_{i \in P} x_i \leq 1$$

$$1 \leq \sum_{i \in D} x_i \leq 6$$

$$1 \leq \sum_{i \in A} x_i \leq 4$$

$$\forall i = 1, \dots, m. \quad 0 \leq \sum_{i \in L_i} x_i \leq 1$$

#### 4.1.13 Esercizio 1.35

*Variabili*

$x_{ik}$  = variabile logica che dice se teniamo l'impianto i per il mercato k

$y_i$  = variabile logica che dice se l'impianto i rimane aperto

*Funzione obiettivo*

$$\min \sum_{k \in K} \sum_{i \in I \cup J} x_{ik} c_{ik} b_k$$

*Vincoli*

$$\forall k \in K. \quad \sum_{i \in I \cup J} x_{ik} = 1$$

$\sum_{i \in I} x_{ik} \geq \frac{|I|}{2}$  non va bene! possiamo contare piu' volte lo stesso impianto che pero' funziona su diversi mercati - $j$  variabili ausiliarie

$\forall i \in I \cup J. \quad y_i \leq \sum_{k \in K} x_{ik}$  se tutti 0, allora  $y_i = 0$   $y_i \geq x_{ik}$  disgiunzione logica generalizzata a k elementi

$$2 \sum_{i \in I} y_i \geq |I|$$

$$2 \sum_{i \in J} y_i \geq |J|$$

#### 4.1.14 Esercizio 1.7

**Testo**

Un'azienda di trasporti ha bisogno di acquistare n litri di carburante ogni mese. Affinché gli autoveicoli funzionino a dovere, occorre che il numero di ottani del carburante utilizzato sia almeno k. L'azienda acquista il suo carburante da due fornitori A e B, che vendono carburante da kA e kB ottani, rispettivamente. Sia A che B applicano un prezzo al litro piuttosto alto (pA e pB, rispettivamente) per i primi s litri di carburante acquistati dall'azienda (ogni mese). Se il numero di litri acquistati ogni mese supera s, i due fornitori applicano invece un prezzo al litro più basso (ossia bA e bB, rispettivamente), ma solo ai litri eccedenti s. Si formuli in PL il problema di determinare da chi acquistare il carburante necessario ogni mese in modo da minimizzare il costo complessivo e da rispettare il requisito sugli ottani. *Variabili*

$x_i$  = "litri da acquistare di tipo i"

$$y_i = \begin{cases} 1 & \text{se } x_i \geq s \\ 0 & \text{altrimenti} \end{cases}$$

$$z_{i1} = \begin{cases} x_i & \text{se } 0 < x_i < s \\ 0 & \text{altrimenti} \end{cases}$$

$$z_{i2} = \begin{cases} x_i - s & \text{se } x_i \geq s \\ 0 & \text{altrimenti} \end{cases}$$

*Vincoli*

$$\begin{aligned} x_A + x_B &= n \\ \frac{k_A \cdot x_A + k_B \cdot x_B}{n} &\geq k \\ 0 &\leq z_{i1} \leq () \end{aligned}$$

*Funzione Obiettivo*

$$\min \sum_{i \in \{A, B\}} z_{i1} p_i + z_{i2} b_i + s p_1 y_i$$

#### 4.1.15 Esercizio 1.6

**Testo**

Una radio privata deve fare arrivare il suo segnale in una città che si trova al di là di una catena montuosa rispetto alla città da cui trasmette. Per fare ciò decide di installare  $n$  ripetitori sulla cima di altrettante colline, ciascuna delle quali si trova ad un'altitudine pari a  $a_i$ . Il costo di costruzione di ogni ripetitore dipende dalla sua altezza: ogni metro costa  $c_i$ . Occorre soddisfare solo un vincolo: il dislivello, in metri, tra la cima di un ripetitore e la cima del successivo non può superare una soglia pari a  $k$ . Si formuli, in PL, il problema di decidere le altezze dei ripetitori in modo che il relativo costo sia minimo.

**Svolgimento**

*Variabili*

$x_i$  = "altezza del ripetitore i"

*Vincoli*

$$\begin{aligned} \forall i \in [1 \dots (n-1)] \quad x_{i+1} + a_{i+1} - x_i - a_i &\leq k \\ \forall i \in [1 \dots (n-1)] \quad -x_{i+1} - a_{i+1} + x_i + a_i &\leq k \end{aligned}$$

*Funzione Obiettivo*

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

#### 4.1.16 Esercizio 1.4 - Problema di Manutenzione degli Aerei

**Testo**

Una compagnia aerea dispone di  $n$  aerei  $1, \dots, n$ , e deve decidere in quale aeroporto, tra gli  $m$  in cui opera, svolgere le operazioni di manutenzione di ciascun aereo. Ogni aereo  $i$  è normalmente basato sull'aeroporto  $a_i \in \{1, \dots, m\}$ . Svolgere le operazioni di manutenzione di un aereo presso l'aeroporto  $j$  costa  $c_j$  euro. Se le operazioni di manutenzione si svolgono in un aeroporto diverso da quello in cui ogni aereo è basato, occorre poi sostenere un costo fisso pari a  $s$ . Si formuli in PLI il problema di minimizzare i costi complessivi di manutenzione



## Svolgimento

Bho, non ho capito, magari chiedere al prof

### 4.1.17 Esercizio 1.3

#### Testo

Una compagnia deve affittare dei computer per soddisfare il lavoro dei prossimi quattro mesi.

Mese	Gennaio	Febbraio	Marzo	Aprile
Requisito	9	5	7	9

Table 4.1: Requisiti mensili di computer.

Il costo dell'affitto dipende dalla lunghezza dell'affitto stesso.

Lunghezza	1 mese	2 mesi	3 mesi
Costo	200	350	450

Table 4.2: Costo dell'affitto in base alla lunghezza.

Si dia una formulazione in programmazione lineare intera per il problema di trovare un piano di affitto di minimo costo.

## 4.2 Tutor

### 4.2.1 Esercizio 1.24

NON SO DOVE SIA IL RESTO DELL'ESERCIZIO

*Variabili*

$x_{ij}$  = viene usato l'oleodotto che collega il giacimento  $i$  alla raffineria  $j$

$y_{ij}$  = l'oleodotto  $ij$  viene utilizzato variabile logica

*FO*

*Vincoli*

$$\forall i. \sum_{j=1}^m x_{ij} = p_i$$

$$\forall j. \sum_{i=1}^n x_{ij} \leq r_j$$

$$\forall i, j. x_{ij} \leq y_{ij} \cdot a_{ij}$$

### 4.2.2 Esercizio 1.33

*Dati*

$n$  = num camion

$\forall i = 1, \dots, n :$

$m_i$  = chilometri massimi percorribili per il camion  $i$

$k_i$  = limite di chilometri per polizza bassa

$c_i$  = costo polizza quando  $m_i \leq k_i$

$d_i$  = polizza quando  $m_i > k_i$

*Variabili*

$p_i$  = chilometri percorsi dal camion  $i$

$b_i$  = il camion  $i$  utilizza la polizza bassa variabile logica

*Vincoli*

$\sum_{i=1}^n p_i \geq 2000000$  (possiamo mettere  $\geq$  e non  $=$  dato che minimizzando il valore ottimale sarà quello più basso)

$$m_i \leq b_i k_i + (1 - b_i) m_i$$

$$m_i > (1 - b_i) k_i$$

*FO*

$$\min \left\{ \sum_{i=1}^n b_i c_i + (1 - b_i) d_i \right\}$$