Ottimizzazione Combinatoria
Appunti

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

Contents

Chapter 1 **Introduzione** Page

Chapter 2 **Problemi e Modelli** Page

2.1 Problemi di ottimizzazione
2.1.1 Problemi di ottimizzazione — • Problemi di ottimizzazione e di decisione — • Aspetto algoritmico —

2.2 Modelli
Programmazione Lineare — • Programmazione lineare intera —

2.3 Template per risolvere problemi
Vincoli di assegnamento — • Selezione di Sottoinsiemi — • Variabili a valori discreti — • Minima quantità positiva — • Funzione a carico fisso — • Rappresentazioni del valore assoluto — • Funzioni definite a tratti —

Chapter 3 **Reti di Flusso** Page

3.1 Problemi di Flusso
Vincoli — • Ipotesi semplificative — • Problema del Flusso di Costo Minimo — • Problema di Flusso Massimo —

3.2 Problema del flusso massimo: algoritmi
Tagli — • Cammini aumentanti e Grafo residuo — • Algoritmo Ford-Fulkerson — • Algoritmo di Edmonds-Karp — • Algoritmo di Goldberg-Tarjan —

3.3 Il Problema del Flusso di Costo Minimo: Algoritmi
nozioni preliminari — • Cammini aumentanti — • minimalità —

3.4 Problemi di Accoppiamento
Massima Cardinalita' — • Costo Minimo —

Chapter 4 **Programmazione Lineare** Page

4.1 Geometria della PL
Nozioni Preliminari —

4.2 Inviluppi Convessi

Chapter 5 **Esercitazioni** Page

5.1 Problemi e modelli
Problema dei data center — • Problema del docente di informatica — • Problema delle commesse — • Problema della minimizzazione del massimo — • Problema della massimizzazione del minimo — • Esercizio valore assoluto — • Esercizio 1.26 — • Esercizio 1.27 — • Esercizio 1.28 — • Esercizio 1.29 — • Esercizio 1.31 — • Esercizio 1.32 — • Esercizio 1.35 — • Esercizio 1.7 — • Esercizio 1.6 — • Esercizio 1.4 - Problema di Manutenzione degli Aerei — • Esercizio 1.3 —

5.2 Tutor
Esercizio 1.24 — • Esercizio 1.33 —
5.3 Esercitazioni reti di flusso con Luca
Ex 1 — • Ex 2 —

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 accetto 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*, sopratutto 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 paramteri 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'instanza di $\mathcal{P}$ definita precedentemente

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

si ha che

$$\mathbb{G} = \mathbb{R}$$
$$\mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} | 5x^2 - 6x + 1 = 0\}$$

2.1.1 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 $c_{\mathcal{P}} : \mathbb{F}_{\mathcal{P}} \to \mathbb{R}$ 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 "ottimizato"), 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

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

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:

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

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

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

Il massimo tra questi valori è:

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

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 ch epreso 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 efficenti.
- 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

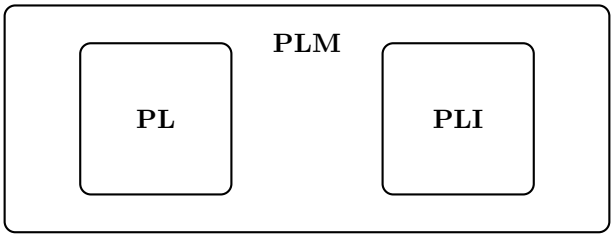
Fornisco Innanzitutto la def. di problema di problema di Programmazione Lineare

Definition 2.2.1: Problema di Programmazione Lineare

Si definiscono **Problemi di programmazione lineare (PL)** tutti quei problemi di ottimizzazione in cui la funzione obbiettivo $c_{\mathcal{P}}$ 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:

- $\mathbb{G} = \mathbb{R}$, definendo poi un numero finito $n \in \mathbb{N}$ di variabili reali, che nella realtà rappresentano delle *quantità*
$$x = (x_1, \dots, x_n) \in \mathbb{RN}$$
- Una *funzione obiettivo* $c_{\mathcal{P}}$ definita $f : \mathbb{R}^n \to \mathbb{R}$ 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:
$$ax = b \quad ax \leqslant b \quad ax \geqslant b$$
Dove $a \in \mathbb{R}^n$ e $b \in \mathbb{R}$

f e' lineare. c_p e' l'insieme di vettori di numeri reali. $\mathbb{G} = \mathbb{R}^n$. (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:

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

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 $ax \leqslant b$ e $ax \geqslant b$
- Ogni vincolo $ax \geqslant b$ è equivalente a $(-a)x \leqslant (-b)$.

Example 2.2.1 (Pianificazione della Produzione)

TESTO

La societ‘a Pintel deve pianificare la produzione della sua fabbrica di micro- processori. La Pintel possiede due diverse linee di prodotti: i processori Pin- tium, pi‘u potenti e destinati al mercato “server”, ed i Coloron, meno potenti e destinati al mercato “consumer”. L’impianto ‘e 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, pi‘u 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 quantit‘a di processori che possono essere messi sul mercato ogni settimana senza causare un calo dei prezzi ‘e di 400.000 unit‘a per i Pintium e di 700.000 unit‘a per i Coloron. Si vuole determinare le quantit‘a di ciascun tipo di processore da produrre settimanalmente in modo da massimizzare il ricavo totale.

SVOLGIMENTO

- VARIABILI

  Prendiamo le prime due variabili ovvie:

  $$x_p = \text{"numero di pintium prodotti"}$$
  $$x_c = \text{"numero di coloron prodotti"}$$

  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:

  $$w_p = \text{"numero wafer utilizzati per pintium"}$$
  $$w_c = \text{"numero wafer coloron"}$$

- VINCOLI

  Mettiamo i vincoli di produzione settimanali:

  $$0 \leqslant x_p 0.5 \leqslant 400000$$
  $$0 \leqslant x_c 0.6 \leqslant 700000$$

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

  $$w_p + w_c \leqslant 3000$$

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

  $$300 w_p = x_p$$
  $$500 w_c = x_c$$

  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:

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

  L’insieme delle soluzioni ammissibili e’ quindi:

  $$F = \{(x_p, x_c) | 0 \leqslant x_p 0.5 \leqslant 400000, 0 \leqslant x_c 0.6 \leqslant 700000, 5x_p + 3x_c \leqslant 4500000\}$$

- 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:

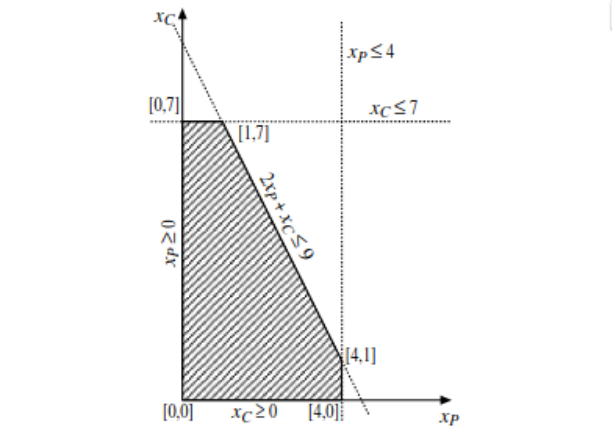
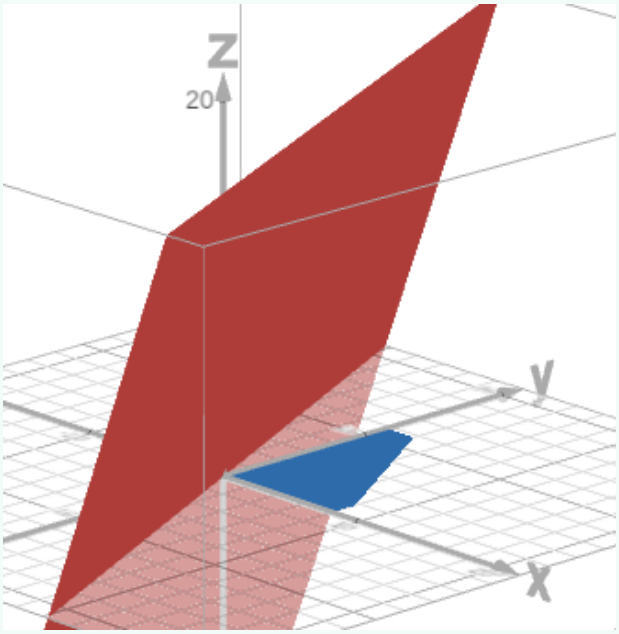


Figura 1.1: Insieme ammissibile per il problema della Pintel

Un'importante proprieta' dei problemi lineari (reali) e' che la soluzione ottimale si trova sempre su uno dei "vertici" della figura descritta dalle soluzioni ammissibili (in questo caso quella ottimale e' (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 realta', con la soluzione trovata il numero di wafer per tipo di chip non e' intero! In questo caso possiamo supporre che non sia troppo un problema, ma in altri casi puo' 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:

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

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 e' 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, . . . , n di elementi, a ciascuno dei quali sia assegnato un peso ai ed un costo ci, i = 1, . . . , 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:**

$$\begin{aligned} E &= \{1, \dots, n\} \\ a_i &= \text{peso dell'oggetto } i \\ c_i &= \text{costo dell'oggetto } i \\ b &= \text{peso massimo dello zaino} \end{aligned}$$

Variabili:
Dobbiamo indicare usando le variabili quali elementi vogliamo includere nella soluzione e quali vogliamo scartare. Possiamo fare cio 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:

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

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** ($z = x \wedge y$):
$$\begin{aligned} z &\leqslant x \\ z &\leqslant y \\ z &\geqslant x + y - 1 \end{aligned}$$
- **Disgiunzione** ($z = x \vee y$):
$$\begin{aligned} z &\geqslant x \\ z &\geqslant y \\ z &\leqslant x + y \end{aligned}$$
- **Implicazione** ($z = x \implies y$):
$$\begin{aligned} x + z &\geqslant 1 \\ z &\geqslant y \\ x + z &\leqslant 1 + y \end{aligned}$$

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

Per rappresentare le diverse condizioni di assegnazione degli oggetti ai luoghi, si introduce la variabile $x_{ij} \in \{0, 1\}$ (dove $1 \leqslant i \leqslant n$ e $1 \leqslant j \leqslant m$). 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:

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

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 particole 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:

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

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:

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

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

Funzione obbiettivo:

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

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 $N = \{1, ..., n\}$ un insieme finito di elementi e sia $F = \{F_1, ..., F_n\}$ 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'e' 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 $x = (x_1, ..., x_m)$ 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 e' sempre:

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

I vincoli, invece, dipendono dal problema:

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

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

  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 si 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 iniemi di F sono offerte da fornitori rispetto alla risoluzione di alcuni dei task. Quindi gli inisiemi di D devono essere disgiunti perche' non vogliamo che due societa' 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 piu' incombenze o task, ma risorse da usare una volta per costruire un prodotto (elemento di F). Al piu' perche' possiamo scartare

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

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

Example 2.3.2 (Problema delle Commesse)

Testo

Un'azienda deve decidere come impiegare i suoi n dipendenti $1, \ldots, n$.
L'azienda, nell'intervallo di tempo desiderato deve evadere m commesse $1, \ldots, m$
Ciascuna commessa j deve essere svolta dal sottoinsieme $D_j \subseteq \{1, \ldots, n\}$ 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: $N = \{1, \ldots, n\}$ elementi/dipendenti
$F = \{F_1, \ldots, F_m\}$ 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:

$$0 \leqslant x_j \leqslant 1 \quad \forall j \in \{1, \ldots, m\} \quad 0 \leqslant y_j \leqslant 1$$

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

Funzione obbiettivo:

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

Nota: le scritte in rosso sono una possibile modifica all'esercizio dove per ogni commessa non fatta c'e' una penalita 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 inseime finito di valori $Z = \{z_1, ..., z_k\}$ dove i vari z_i sono valori reali distinti
In questi casi, si possono utilizzare n variabili ausiliarie $y_1, ..., y_k$ che rappresentano l'appartenenza di z a Z. In particolare, si definiscono le variabili y_i come segue:

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

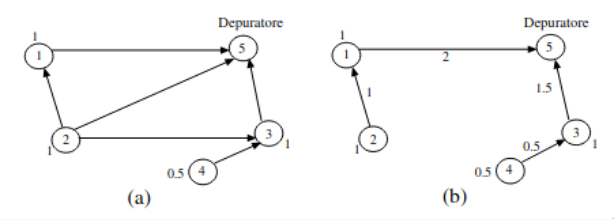
E sono vincolate come segue:

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

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 quantit‘a nota di rifiuti (liquidi), ed ‘e necessario dimensionare opportuna- mente 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 sar‘a 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 ‘e indicata la quantit‘a di liquido da depurare per unit‘a 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 dovr‘a determinare la portata. Osserviamo che i collegamenti hanno una direzione prefissata di percorrenza. Una soluzione ammissibile ‘e data da un insieme di collegamenti che ga- rantiscano il fluire dei liquidi da tutti i centri fino al depuratore. In figura 1.3(b) ‘e 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 e' il livello minimo di produzione
- u e' il livello massimo di produzione
- 0 è l'assenza di produzione

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

$$x \geqslant ly \quad x \leqslant uy$$

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:

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

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:

$$0 \leqslant x \quad x \leqslant uy$$

Adesso è possibile sostituire la funzione a carico fisso tramite una nuova funzione $g : \mathbb{R}^2 \to \mathbb{R}$ 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 eslcudono 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'obbiettivo 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 esempip

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:

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

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:

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

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

Funzioni obbiettivo 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 obbiettivo
*Anal*olgalmente, 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 lineare a tratti
Si consideri la seguente funzione

$$f(x) = \begin{cases} b_1 + c_1 x & x \in [a_1, a_2] \\ b_2 + c_2 x & x \in (a_2, a_3] \end{cases} \tag{2.2}$$

Dove assumiamo

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

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:

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

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:

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

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

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

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

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

Riassumendo tutti i vincoli ottenuti, si ha:

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

Note:
Si noti che i vincoli 2.5 impongono che solo una delle due variabili z_1 e z_2 possa avere valore non nullo

Rappresentazione di f con funzione mutlivariabile

Poi il basta mi spiegherà la sequente quote

> Non c'e' niente che ci da 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 sostuire alla funzione $f(x)$ la seguente funzione $g : \mathbb{R}^4 \to \mathbb{R}$:

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

Con le variabili soggette ai vincoli 2.5. In questo modo è stato possibile rappresentare la funzione $f(x)$ (non lineare) come una funzione lineare a variabili continue definita du un opportuno insieme.
Si ha che il valore di f e' rappresentato *univocamente* dalla quadrupla (z_1, z_2, y_1, y_2), eccetto nel punto di non continuita' 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 prblema come benigno, dato che:

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

Chapter 3

Reti di Flusso

Innanzi tutto inziamo con alcune definzioni:

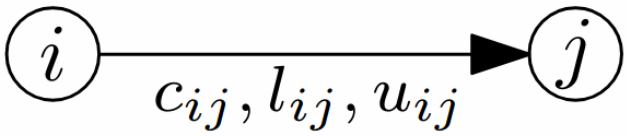
Definition 3.0.1: Rete

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

- N e' l'insieme dei nodi
- A e' 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 qunatità del bene che esce della rete al nodo i, mentre b_i è ldetto *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*
  - **Nullo**: 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, e' 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 e' dato da:

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

3.1.1 Vincoli

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

- *Uguaglianza di domanda e offerta globale*:

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

  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*

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

  dove

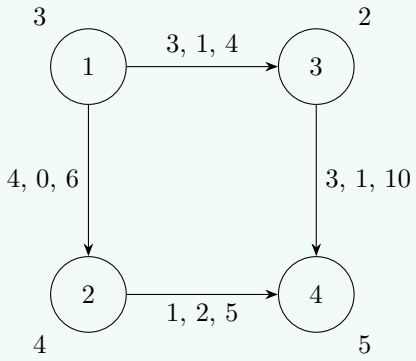
$$BS(i) = \{(k,i) | (k,i) \in A\} \text{ detto stella entrante}$$
$$FS(i) = \{(i,k) | (i,k) \in A\} \text{ detto stella uscente}$$

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

- *Ammissibilità del flusso*:

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

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 INGENERI 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 quantita'

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

Alla funzione abbiettivo

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

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 obbiettivo è il costo di flusso da minimizzare e le cui capacità inferiori sono nulle

Questo problema è formalizzabile in programmazione lineare come segue:

$$\begin{aligned} &\min cx \\ &Ex = b \quad \leqslant x \leqslant u \end{aligned} \tag{3.4}$$

dove

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

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

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

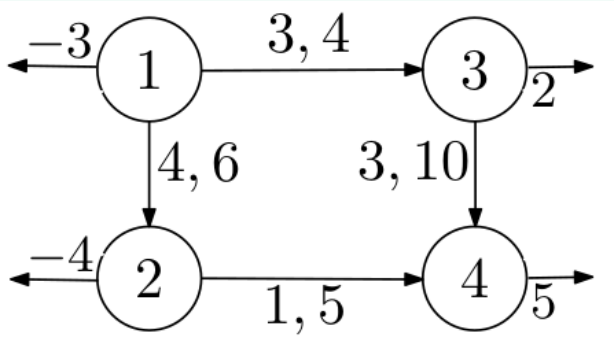
Rilassamento di assunzioni

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

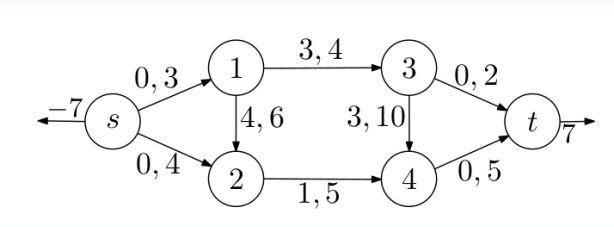
Si parte da questa rete:



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:



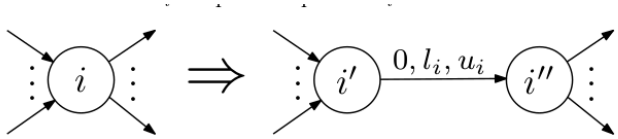
Non si può, però, enunciare che i nodi non hanno una capacita', perche' di fatto non succede nulla, perche' puo' essere trasformata in una rete equivalente dove i nodi non ce l'hanno:

TODO: spiegare meglio, che cazzo vuoi dire ibbastianini?

Mi spiego in modo piu' eloquente: Per poter utilizzare nodi che hanno una propria capacita', non e' necessario perdere generalita' cambiando la definizione del problema esaminato. Questo e' vero, in quanto una tale rete puo' essere banalmente trasformata in una rete equivalente la quale presenta solo nodi con capacita' nulla. Come accade tale trasformazione? Facciamocelo spiegare da GioPalmi:

Alle volte, inoltre, è utile che anche i nodi abbiano delle **capacità**, ossia che solo 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

Dato un grafo $G = (N, A)$ orentato su cui definiamo:

- il vettore di capacita' superiori $u = [u_{ij}]$ associate agli archi
- s (*origine* o *sorgente*) e t (*destinazione* o *pozzo*) due nodi distinti

si definisce *problema di flusso massimo* il problema di massimizzazione del flusso da s a t.
Ovvero, il problema consiste nel trovare il **massimo** valore v tale che se $b_s = -v$, $b_t = v$ e per tutti gli altri casi $b_i = 0$, allora esiste un flusso $x = [x_{ij}]$ ammissibile. Tale v si dice **valore** del flusso x.

Ciò che cambia, quindi, dal problema di flusso di costo minimo è la funzione obbiettivo, che diventa, si vuole infatti *massimizzare* il flusso, e non minimizzare il costo (che non ci interessa).
Formalizzato in programmazione lineare, il problema di flusso massimo diventa:

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

Note:

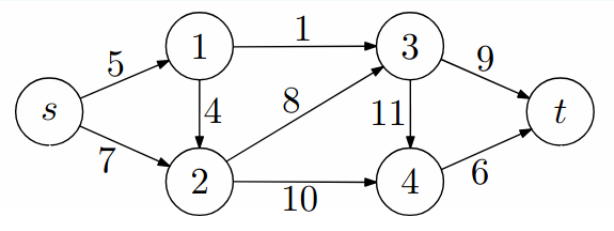
Si noti che il problema di flusso massimo e' un caso particolare di problema di flusso di costo minimo. Difatti la formulazione 3.6 puo' essere vista come la descrizione di un problema di MCF su un grafo G' ottenuto da G con delle modifiche:

- Si aggiunge un arco fittizio (t, s), detto *arco di ritorno*, con capacita' infinita.
- Tutti i nodi sono di trasferimento ($\forall i \in N.\ b_i = 0$), si tratta quindi di un problema di *circolazione*.
- Tutti i costi degli archi sono nulli tranne (t, s) che ha costo -1.

Di conseguenza, risolvendo il problema di MCF su G', stiamo minimizzando il valore $-1 \cdot x_{ts}$ rispettando i vincoli imposti, che per definizione corrisponde a $-v$. Ma minimizzare $-v$ significa massimizzare il valore v del flusso fra t ed s, che e' proprio l'obbiettivo del problema di MF.

Example 3.1.3 (Risolvere un problema MF come MCF)

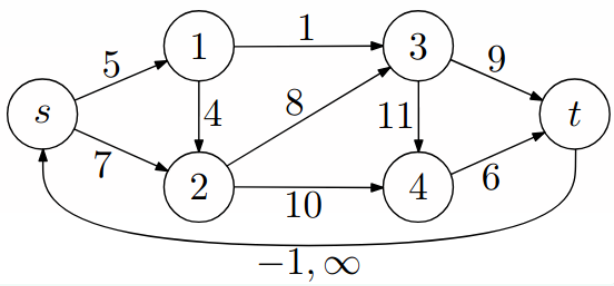
Dato il seguente problema di MF, con capacita' superiori indicate in figura e dove i nodi diversi da s e t hanno sbilanciamento nullo (come per definizione del problema), trasformarlo in un problema MCF:



Il problema di MF consiste nel determinare il massimo valore di flusso v, ovvero il massimo valore di sbilanciamento uguale per s (ma con segno meno) e t (devono essere per forza uguali per il vincolo di *uguaglianza di domanda e offerta globale*) per cui esiste un flusso ammissibile $x = [x_{ij}]$ (che quindi rispetta i vincoli di *ammissibilita'* e di *conservazione*).
A prima occhiata, possiamo subito dire che il valore limite per v e' sicuramente $5 + 7 = 12$, dato che per il vincolo di conservazione $b_s = x_{s1} + x_{s2} \leqslant u_{s1} + u_{s2}$ (facendo lo stesso ragionamento con t, otteniamo un limite meno stringente, quindi prevale questo). Siamo sicuri che $v = 0$ sara' sempre una soluzione ammissibile, dato che $x = \underline{0}$ e' sempre un flusso ammissibile, ma non e' quasi mai il valore massimo. Facciamo i passi con $v = 12$:

- Dobbiamo mettere $x_{s1} = 5$ e $x_s2 = 7$
- Dobbiamo mettere $x_{13} = 1$ e $x_{12} = 4$
- Ora abbiamo che $\sum_{(j,i)\in BS(2)} x_{ji} = 7 + 4 = 11$ e dobbiamo decidere come dividere il flusso entrante fra quelli uscenti, dato che questa volta non vengono saturati. Possiamo usare l'euristica e guardare i nodi di destinazione: il 3 ha capacita' superiori per i flussi uscenti rispetto al nodo 4, quindi gli mandiamo piu' flusso possibile (questa e' solo un'intuizione e non e' affatto una regola generale). Quindi $x_{23} = 8$ e $x_{24} = 3$.
- Poniamo $x_{3t} = 9$ e $x_{4t} = 3$: esiste un flusso x ammissibile per $v = 12$! E dato che 12 e' il limite massimo, siamo sicuri che e' la soluzione al problema.

Non sempre sara' cosi' facile trovare la soluzione corretta. Vediamo il problema MCF corrispondente e convinciamoci che e' equivalente:



Le capacita' massime sono uguali, tutti i nodi hanno sbilanciamento nullo e abbiamo aggiunto l'arco di ritorno. La funzione obbiettivo e:

$$min - x_{ts} \equiv max x_{ts}$$

dato che tutti gli altri costi sono nulli. Dato che s e t hanno sbilanciamento nullo, per i vincoli di conservazione e di ammissibilita':

$$x_{ts} = x_{3t} + x_{4t} \leqslant u_{3t} + u_{4t} = 15 \quad \wedge \quad u_{s1} + u_{s2} = 12 \geqslant x_{s1} + x_{s2} = x_{ts}$$

da qui otteniamo che $x_{ts} \leqslant 12$ e' il limite superiore. Da questo punto, controllare se con $x_{ts} = 12$ possiamo costruire un flusso ammissibile segue lo stesso identico procedimento di prima. Quindi il flusso che minimizza il costo e' proprio x il cui valore di flusso associato $v = x_{ts}$ e' il valore della soluzione MF.

3.2 Problema del flusso massimo: algoritmi

Abbiamo capito quindi che il problema di Flusso Massimo e' un problema di MCF con due caratteristiche specifiche:

- Il vettore dei bilanci b e' nullo
- Il vettore dei costi c e' nullo tranne per c_{ts} che vale -1

Vediamo come queste caratteristiche permettono di utilizzare algoritmi specifici al problema, che sono molto piu' veloci di quelli generali.

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 defnizione

Definition 3.2.2: (s,t)-taglio

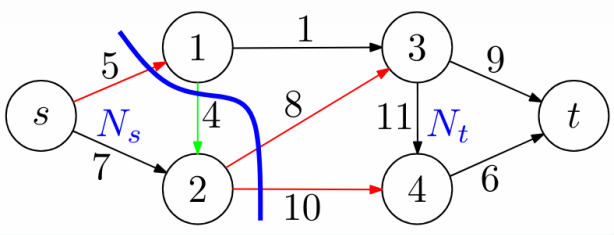
Si definisce (s,t)**-taglio** in una rete $G = (N, A)$ un taglio (N_s, N_t) dove $s \in N_s$ e $t \in N_t$

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

Example 3.2.1 (Taglio (s,t) di un problema di flusso massimo)

Ecco un esempio di taglio (rappresentato dalla riga blu) che partiziona in due sottoinsiemi i nodi di un problema di flusso massimo:



Dal grafo si deduce che:

- $N_s = \{s, 2\}$
- $N_t = \{1, 3, 4, t\}$
- $A^+ = \{(s,1), (2,3), (2,4)\}$
- $A^- = \{(1,2)\}$

Adesso introduciamo un importante lemma sui tagli:

Lenma 3.2.1

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

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

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 $x(N_s, N_t)$

- $v \leqslant \sum_{(i,j)\in A^+(N_s,N_t)} u_{ij}$

  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:

$$v = \sum_{(s,j)\in A} x_{sj} - \sum_{(i,s)\in A} x_{is}$$

Questo valore è uguale alla somma netta in uscita $\forall i \in N_s$ verso la partizione N_t, quindi: [1]

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

Riscrivendo usando la proprieta' distributiva, posso dividere l'esspressione in due parti:

$$\underbrace{\sum_{k\in N_s} \sum_{(k,j)\in A} x_{kj}}_{\text{somma archi uscenti } S^+} - \underbrace{\sum_{k\in N_s} \sum_{(i,k)} x_{ik}}_{\text{somma archi entranti } S^-}$$

Dato che $k \in N_s$, consideriamo tutti gli archi (p, q) che hanno *almeno* un nodo dentro a N_s, ci sono quindi tre casi:

$$\begin{cases} p \in N_s \wedge q \in N_t & \rightarrow S^+ \mathrel{+}= x_{pq} \\ p \in N_t \wedge q \in N_s & \rightarrow S^- \mathrel{+}= x_{pq} \\ p, q \in N_s & \rightarrow S^+ \mathrel{+}= x_{pq} \wedge S^- \mathrel{+}= x_{pq} \end{cases}$$

Ma aggiungere lo stesso valore in S^+ e in S^- non ha effetto sul valore finale dato che si annullano, quindi possiamo anche non considerare gli archi fra nodi che appartengono a N_s.

Consideriamo quindi solo gli archi che attraversano il taglio: S^+ e' la somma dei flussi da N_s a N_t e S^- i flussi da N_t a N_s. Quindi la loro somma e' uguale al flusso uscente dal confine $A^+(N_s, N_t)$ meno il flusso entrante dal confine $A^-(N_s, N_t)$, allora:

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

Per riassumere quindi:

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

- Dimostro la seconda parte:

  Ovvio perché è una semplice derivazione del punto 1

[1] Possiamo dimostrare questa cosa dato che, per il vincolo di conservazione e per definizione del problema:

$$\forall k \in N_s. \sum_{(k,j)\in A} x_{kj} - \sum_{(i,k)} x_{ik} = \begin{cases} v & k = s \\ 0 & k \neq s \end{cases}$$

Note:

si noti che la conclusione di questo Lemma è

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

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

Note:

Abbiamo utilizzato il secondo punto del lemma all'esempio 3.1.4, quando abbiamo imposto un limite superiore al valore di v sommando le capacita' massime degli archi uscenti da s e poi con gli archi entranti a t. Quindi e' come se avessimo costruito un taglio con $N_s = \{s\}$ e uno con $N_t = \{t\}$, per poi imporre il secondo punto del lemma, scegliendo quello minore come limite.
Pero' abbiamo controllato solo due tagli specifici, quello che aggiunge il lemma e' che tale proprieta' vale per tutti i tagli, quindi per trovare un limite migliore tocca calcolare la capacita' di tutti i tagli possibili.

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

3.2.2 Cammini aumentanti e Grafo residuo

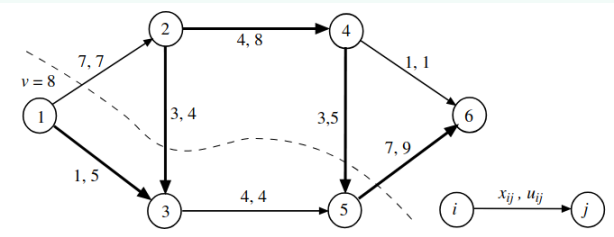
Innanzi tutto dobbiamo presentare alcuni concetti

Cammino aumentante

Dato una rete di un problema di MF e un suo flusso ammissibile, noi vogliamo sapere se questo flusso e' ottimo. Se non fosse ottimo, cio' significa che sarebbe possibile inviare altro flusso da s a t, che quindi dovra' seguire un certo cammino P. Tale cammino non segue per forza l'orientamento del grafo, quindi e' possibile che un arco (i, j) venga attraversato al contrario. Provo a dare l'intuizione del perche' con un esempio:

Example 3.2.2

Consideriamo il seguente grafo (ignorare il taglio):



Attualmente, il valore del flusso vale 8. Vogliamo vedere se e' possibile aumentare il flusso. Proviamo ad aumentare il flusso di 2 ($v = 10$):

- Dato che l'arco $(1, 2)$ e' saturo, devo aumentare x_{13} di 2
- L'unico arco uscente da 3 e' saturo, quindi non possiamo far proseguire il flusso. Puo' sembrare di essere in un vicolo cieco, perche' dove lo mettiamo questo flusso in piu'? Ma, come abbiamo detto prima, il cammino che stiamo cercando *non e' per forza orientato*, quindi sarebbe possibile "proseguire" lungo l'arco $(2, 3)$ al contrario. Questo e' perche' in questa situazione possiamo fare una giocata: il nodo 3 e' sovraccaricato, dato che il flusso non puo' uscire da nessuna parte, quindi dobbiamo diminuire quello in entrata. Il flusso x_{13} appena aumentato lo vogliamo tenere, dato che fa parte del percorso, quindi *diminuiamo* il flusso x_{12} della stessa quantita' di quanto abbiamo aumentato il flusso dell'arco entrante. In questo modo il totale di flussi entranti al nodo 3 rimane uguale, ma il nodo 2 ha ora ben 2 di flusso in piu' che puo' utilizzare! Quindi, il flusso in piu' che vogliamo far arrivare a 6 si e' spostato da 3 a 2, ed e' proprio per questo che possiamo attraversare gli archi al contrario. Ma attenzione, al posto di aumentare il valore del flusso x_{23}, l'abbiamo diminuito, cosa che definiremo meglio dopo.

- Ora che il nodo 2 ha del flusso extra, lo inviamo al nodo 4 ($x_{24}+ = 2$)
- Il nodo 4 lo manda al nodo 5 che infine lo manda al nodo 6.

Notare che se avessimo scelto di aumentare v di 3, dal nodo 4 non sarebbe stato possibile rispettare i vincoli di ammissibilita', quindi il valore 2 non era scelto a caso.

Ora che e' chiaro cosa stiamo cercando e perche', iniziamo a formalizzare le intuizioni. Per prima cosa, distinguiamo gli archi attraversati col verso "giusto" da quelli attraversati con verso "opposto":

Definition 3.2.3: Archi concordi e discordi

Dato un cammino P su una rete G, chiamiamo:

- L'insieme degli archi *conocrdi* P^+ l'insieme degli archi del percorso P che hanno lo stesso orientamento degli archi corrispondenti in G
- L'insieme degli archi *discordi* P^- l'insieme degli archi del percorso P che hanno verso opposto rispetto agli archi corrispondenti in G

Example 3.2.3

Quindi, nell'esempio di prima:

- $P^+ = \{(1,3),(2,4),(4,5),(5,6)\}$
- $P^- = \{(3,2)\}$

Definiamo ora la capacita' di un percorso, che dimostreremo essere la quantita' in piu' di flusso che tale percorso puo' portare da s a t:

Definition 3.2.4: Capacità di un cammino

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

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

Note:

Quindi abbiamo definito $\theta(P,x)$ in questo modo per fare in modo che corrisponda alla quantita' massima di flusso che possiamo mandare in piu' rispetto al flusso OG x in modo che rimanga sempre ammissibile. Notare che e' possibile che tale valore sia nullo, nel caso in cui uno degli archi concordi e' saturo o uno di quelli discordi e' nullo, in questo caso P non e' aumentante.

Dalla definizione di capacita' discende la definizione di flusso $x(P,\theta)$:

Definition 3.2.5: Flusso $x(P,\theta)$

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

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

Lenma 3.2.2

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

: TODO: dimostra (facile)

Possiamo ora definire un cammino aumentante come:

Definition 3.2.6: Cammino aumentante

Data una rete G di un problema MF, un cammino da s a t *non necessariamente orientato* si dice **aumentante** se $\theta(P, x) > 0$.

Grafi residui

Per definire un algoritmo che riesca a determinare se esisti o meno un cammino aumentante dato un grafo G e un flusso ammissibile x, ci e' piu' comodo lavorare su sul cosidetto *grafo residuo* G_x associato:

Definition 3.2.7: 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**
    Signifca che il flusso non ha saturato la capicità 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**

Note:

Osserviamo come in A_{G_x} ci possano essere al piu' due archi per ogni arco $(i,j) \in A_G$ (uno concorde e l'altro discorde), quindi se in A_G esistono gli archi (i,j) e (j,i), e' possibile che in A_{G_x} ci siano due archi (i,j) e due archi (j,i) con capienze diverse.

Introduciamo ora due lemmi che legano i cammini aumentanti di G a cammini semplici e orientati in questo nuovo grafo:

Lenma 3.2.3

Per ogni cammino aumentante da s a t rispetto a x in G, esiste uno ed un solo cammino *orientato* da s a t in G_x.

: Per come abbiamo costruito il grafo residuo G_x, $\forall (i,j) \in A$ se l'arco non e' saturo, allora esiste l'arco orientato $(i,j) \in A_x$ che puo' far parte degli archi *conocordi* di un cammino aumentante. Se l'arco $(i,j) \in A$ e' non nullo, allora esiste l'arco orientato in modo opposto $(j_x, i_x) \in A_x$, che puo' fare parte degli archi *discordi* di un cammino aumentante.
Quindi, se esiste un cammino aumentante P su G dato x, allora e' possibile seguire un cammino semplice orientato da s a t sul grafo dei residui G_x, proprio perche' sappiamo che se P e' aumentante allora tutti gli archi che percorre saranno non saturi (se attraversti in modo orientato) e non nulli (se attraversati in modo opposto), quindi esisteranno per forza i corrispondenti archi orientati in G_x.

Lenma 3.2.4

Se G_x non ha cammini semplici orientati da s a t, allora x e' un flusso massimo.

: Sappiamo che x e' un flusso massimo di valore v se determiniamo un taglio (N_s, N_t) la cui capacita' e' pari a v (TODO: introdurre e dimostrare prima questo lemma). Ci riduciamo quindi a dimostrare che esiste un tale taglio sul grafo residuo G_x se questo non ha visite che raggiungono t.
Siano N_s tutti i nodi raggiungibili da s in G_x e $N_t = N \setminus N_s$ i restanti (contiene almeno t per ipotesi). Siccome non ci sono archi che collegano nodi da N_s a N_t, significa che nel grafo G tutti gli archi di $A^+(N_s, N_t)$ sono saturi e tutti gli archi di $A^-(N_s, N_t)$ sono nulli, altrimenti ci sarebbe stato almeno un arco in G_x che collega N_s a N_t.

Quindi il flusso del taglio (che equivale al valore del flusso x per il lemma) avra' il valore:

$$x(N_s, N_t) = v = \underbrace{\sum_{(i,j)\in A^+} x_{ij}}_{\text{saturo}} - \underbrace{\sum_{(i,j)\in A^-} x_{ij}}_{\text{nullo}} = \sum_{(i,j)\in A^+} u_{ij} = u(N_s, N_t)$$

Abbiamo quindi dimostrato il lemma.

Note:

In altre parole, l'ultimo lemma ci dice che se non esistono cammini aumentanti su G per un dato flusso x, allora nel grafo G_x non esistono cammini semplici orientati da s a t.

Quindi per torvare un cammino aumentante, basta far partire una visita da s nel grafo G_x e vedere se t e' raggiungibile!

Note

Volendo, era possibile partire dalla definizione:

Definition 3.2.8: Cammino aumentante (usando G_x)

Un **cammino aumentante** P in un grafo residuo G_x è un cammino semplice e orientato da s a t.

Nelle slide e' cosi', ma penso che sia piu' corretto partire dal concetto di cammino aumentante sul grafo G (che e' piu' intuitivo) per poi mostrare l'equivalenza con un cammino orientato sul suo grafo residuo.

3.2.3 Algoritmo Ford-Fulkerson

Miglioriamo il flusso corrente usando il grafo residuo con cammino aumentante. Quindi visitiamo il grafo residuo partendo da s, e se c'e' un cammino che arriva a t significa che e' aumentante, quindi si aggiorna x il meglio possibile. fare questo finche 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 ← 0;
2 while True do
3     Make(G_x);                                   // Costruisci il grafo residuo G_x
4     if ∃ un cammino aumentante P ∈ G_x then
5         x ← x(P, θ(P, x));
6     else
7         return x;
```

TODO: usa lemmi dati in precedenza per dimostrare che sta roba funzia

3.2.4 Algoritmo di Edmonds-Karp

L'algoritmo di Ford-Fulkerson è corretto ma può avere complessità esponenziale nel caso pessimo. L'algoritmo di Edmonds-Karp nasce proprio per rendere la complessità di Ford-Fulkerson polinomiale nel caso pessimo, l'idea è mantenere quindi l'algoritmo invariato, ma cambiare il modo in cui si scelgono i commini aumentanti nel grafo residuo G_x, imponendo che la ricerca del cammino aumentante venga fatta usando **BFS (Breadth - First Search)**

Note:

Grazie alla visita in ampiezza i cammini aumentanti saranno sempre cammini di lunghezza minima

Correttezza

EK è trivialmente

Complessità

Per valutarne la complisstià occorre procedere in questo modo, prima si procede osservando che, se in FF i cammini aumentanti sono di lunghezza minima, allora la distanza di un generico nodo i dalla sorgente s in G_x non può diminuire, da ciò si deducie che il numero di iterazione di EK non può essere asintoticamnte più grande di $N \cdot A$. In particolare:

Definition 3.2.9: Distanza fra nodi di un grafo residuo

Data una rete G, un flusso ammissibile x e due nodi i, j in G, indiachiamo con $\delta_x(i,j)$ la distanza tra i e j nel grafo residuo G_x

Lenma 3.2.5

Se, durante l'esecuzione di EK il flusso y è ottenuto da x tramite un'operazione di aumento del flusso in un cammino aumentante, allora per ogni nodo $i \in N$, vale che

$$\delta_x(s,i) \leqslant \delta_y(s,i)$$

: oH

Theorem 3.2.1 Teorema sulla complessità di Edmonds-Karp

Il numero di iterazioni di EK è $O(NA)$, quindi la sua complessità è $O(NA^2)$

Prima della dimostrazione si noti questa definizione

Definition 3.2.10: arco critico

Un arco (i,j) è detto **critico** per un cammino aumentante P se la sua capacità (ossia $u_{ij} - x_{ij}$ se è concorde o x_{ji} se è discorde) è uguale a $\theta(P,x)$

Note:

In ongi cammino aumentante, esiste almeno un arco critico (e ce ne possono essere piu' di uno). Sono quindi gli archi che spariscono quando viene aggiunto $\theta(P,x)$.

Dimostrazione: Si procede nel contare quante volte un arco (i,j) può diventare critico prima di scomparire definitivamente da grafo

- Quando (i,j) diventa critico per la prima volta significa che nel grafo residuo:

  $$\delta_x(s,j) = \delta_x(s,i) + 1$$

  Cioè, j si trova a distanza 1 in più di i dalla sorgente nel BFS.[2]

- Perché (i,j) possa **riapparire** nel grafo residuo, deve succedere che il flusso da i a j diminuisca+ , cioè:

  $$\delta_y(s,i) = \delta_y(s,j) + 1 \tag{3.10}$$

  Ma allora, per il lemma 3.2.4:

  $$\delta_y(s,i) = \delta_y(s,j) + 1 \geqslant \delta_x(s,j) + 1 = \delta_x(s,i) + 2 \tag{3.11}$$

  Quindi la distanza da s aumenta di almeno 2 ogni volta che un arco diventa critico di nuovo.

[2] Questo e' perche' stiamo usando proprio la BFS, che garantisce che (s,i) e' un cammino minimo per arrivare a i e che il nodo j sia sicuramente piu' lontano da s rispetto a i (altrimenti il cammino aumentante non passerebbe da i). Quindi, dato che i e j sono collegati, il cammino minimo per arrivare da s a j non sara' maggiore di $\delta_x(s,i)+1$ e deve essere che $\delta_x(s,j) \geqslant \delta_x(s,i)$. Dato che la distanza e' un numero intero, l'unico valore ammissibile e' questo.

- **Limite massimo sul numero di volte in cui un arco può diventare critico**:
  - La distanza di un nodo da s nel BFS **non può superare** $|N|$ (il numero di nodi).
  - Poiché la distanza aumenta **di almeno 2** ogni volta che (i, j) diventa critico, un arco può essere critico **al più** $O(N)$ **volte**.

3.2.5 Algoritmo di Goldberg-Tarjan

Un'altro approccio possibile che permette di scendere sotto la soglia di una complessità $O(V A^2)$ consiste nell'utilizzare sempre un algoritmo iterativo, ma locale rispetto al grafo, in quanto il flusso viene modificato solo in una parte del grafo (in opposizione con FF ed EK, in cui ad ogni iterazione occorre procedere con un'analisi globale), un concetto fondamentale che sta dietro all'algoritmo è il concetto di **preflusso**

Definition 3.2.11: preflusso

Un **preflusso** è un vettore x tale che:

$$\sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in BS(i)} x_{ij} \geqslant 0, \quad i \in N\backslash\{s,t\} \tag{3.12}$$

In altre parole, i vincoli di capacità sono soddisfatti, al contrario, quelli di bilanciamento possono non esserlo. **ciò che entra in un nodo è $\geqslant$ a quello che esce**.

Note:

Usiamo questa definizione di flusso piu' "lasca" dato che modificamndo il flusso di un arco solo difficilmente porta ad un flusso totale ammissibile.

che è collegato all'idea di nodo attivo

Definition 3.2.12: nodo attivo/bilanciato

Un nodo si dice **attivo** se il suo eccesso

$$e_i = \sum_{(j,i)\in BF(i)} x_{ji} - \sum_{i,j} x_{i,j} \tag{3.13}$$

L'idea centrale dell'algoritmo è quella di eliminare gli sbilanciamenti presenti nel prefusso corrente. In altre parole cercando di rendere i nodi attivi in nodi bilanciati in modo eliminando l'eccedenza "spostandola" verso i nodi più vicini, in modo *iterativo* e *locale*.
Lo sbilanciamento presente in un nodo i viene eliminato spostando parte del flusso in eccesso, mediante due diverse azioni

- **push forward**: viene in avanti, ossia attraverso un arco (i, j), ogniqualvolta $x_{ij} < u_{ij}$ (quindi se x_{ij} ha capacità residua)
- **backward.**: viene in avanti, ossia attraverso un arco (j, i), ogniqualvolta $x_{ij} > 0$

Tuttavia queste operazioni di push non possono essere sempre eseguite per evitare situazioni di loop o di complessità troppo elevata. Si utilizza un sistema di etichettature: intuitivamente si può trasferire del flusso in avanti o in dietro sse il nodo i ha un'altitudine (una quota) maggiore rispetto agli altri nodi verso i quali si vuole trasferire del flusso (1 o 0)
Adesso fornirò l'algoritmo

```
1  x ← 0;
2  x_sj ← u_sj ∀(s,j) ∈ FS(s);        // Faccio uscire tutto quello che posso da s, quindi, quello che esce da s è
     almeno pari al massimo
3  d ← EtichettaturaValida(G);
4  d_s ← n;                                  // la quota di esse assume il numero di nodi della rete
5  if ∀N ∈ G\{s,t}, N è bilanciato then
6      return x
7  foreach N ∈ G : N sbilanciato do
8      v ← N;
9      if ∃(v,j) ammissibile per v then
10         PushForward(v,j);
11     if ∃(i,v) ammissibile per v then
12         PushBackward(i,v);
13     Relabel(v);
```

Correttezza e complessità

Theorem 3.2.2

L'algoritmo di Goldberg e Tarjan è corretto, e la sua complessità in tempo è $O(N^2A)$

3.3 Il Problema del Flusso di Costo Minimo: Algoritmi

3.3.1 nozioni preliminari

La teoria dei cammini minimi successivi si trasmette bene da problemi di flusso massimo a problemi di costo minimo. Grafi residui e capacita' rimangono uguali, ma e' diverso come determinimao il cammino aumentante di costo minimo dato che dobbiamo tenere in consideraizone il costo.
Partiamo con la definizione di pseudoflusso

Definition 3.3.1: pseudoflusso

Uno **pseudoflusso** è un vettore x che soddisfa i vincoli di capacità, ossia tale che

$$0 \leqslant x_{ij} \leqslant u_{ij} \quad (i,j) \in A \tag{3.14}$$

Da cui discende

Definition 3.3.2: sbilanciamento

Se x è uno pseudoflusso, definiamo sbilanciamento di un nodo i rispetto a x la quantità

$$e_x(i) = \sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in FS(i)} x_{ij} - b_i \tag{3.15}$$

In pratica $e_x(i)$ può essere visto come un vettore degli sbilanciamenti

Il nostro abbiettivo sarà trasformare lo pseudoflusso in un'altro pseudoflusso che risolva le "anomalie" del flusso di partenza, queste anomalie sono dei valori molto alti nel vettore di sbilanciamento

Definition 3.3.3

Si ha inoltre che dato uno pseudoflusso x, i nodi sbilanciati rispetto a x fanno parte di uno dei seguenti due insiemi

$O_x = \{i \in N \mid e_x(i) > 0\}$ quindi lo sbilanciamento è strettamente positivo

$D_x = \{i \in N \mid e_x(i) < 0\}$ quindi lo sbilanciamento è strettamente positivo

I nodi in O_x sono detti **nodi con eccesso di flusso**, mentre quelli in D_x sono detti **nodi con difetto di flusso**

Con O_x si va a segnalare che ciò che entra è inferiore a ciò che esce, al contrario D_x

Note:

Si noti che che se $O_x = N_x = \emptyset$ si ha che x è un flusso, si può quindi intuire che occorrerà procedere sbarazzandoci di O_x e D_x

Un indice ancora più precise che indica quanto è lontano uno pseudoflusso con un flusso ammissibile è:

Definition 3.3.4: sbilanciamento complessivo

Lo **sbilanciamento complessivo** di x è definito come:

$$g(x) = \sum_{i \in O_x} e_x(i) = -\sum_{j \in D_x} e_x(j) \tag{3.16}$$

3.3.2 Cammini aumentanti

Per introdurre concetti futuri occorre "generalizzare" il concetto di grafi residui e cammini aumentanti, si ha che:

Definition 3.3.5: grafo residuo per pseudo flusso

Si definisce **grafo residuo per uno pseudoflusso** x il grafo residuo G_x definito in precedenza dove ogni arco ha un costo:

- $\forall (i,j) \in G_x$ il costo è c_{ij}
- $\forall (i,j) \in G_x$ il costo è $-c_{ij}$

Poi si definisce il cammino aumentate

Definition 3.3.6: cammino aumentante

Un **cammino aumentante** P tra i e j in G_x è un cammino semplice nell'arco residuo (tra i e j)

ovviamente i suoi archi possono essere partizionati in P^+ e P^- e la sua capiacità $\delta(P,x)$ è definita come al solito

Definition 3.3.7: ciclo aumentante

un cammino aumentante tra i e j viene anche detto **ciclo aumentante**

Vorremmo quindi utilizzare questi cammini aumentati per diminuire lo sbilanciamento
Dato uno pseudoflusso x e un cammino aumentante P , è possibile inviare $0 \leqslant \theta \leqslant \theta(P,x)$ (capacità residua lungo un cammino P) unità di flusso lungo attraverso l'operazione $x(P,\theta)$, denoteremo in questo contesto $x(P,\theta)$ (il flusso che ottengo da x pompando θ unità di flusso lungo P) verrà spesso indicato anche con $x \oplus P\theta$. Se P è un cammino aumentante da i a j in Gx , allora lo pseudoflusso $x(P,\theta)$ avrà gli stessi sbilanciamenti di x, tranne in i e j, infatti se io amunto il flusso che va da i a j, tutti gli sbilanciamenti tutti i nodi non cuivolti rimarrano tali e quali, nei nodi che sono attraversati dal cammino tra i e j, ciò entrerà nei nodi sara la setessa quantità di ciò che uscire per definizione dell'operazione di update, ciò che entra viene aumentato e ciò che esce viene aumentato. pertanto lo sbilanciamento avrà solo effetti tra i e j (se sono uguali il vettore degli sbilanciamenti sarà inalterato).
Si passi adesso alla def di costo

Definition 3.3.8: costo

il costo di un cammino aumentate P è definito come

$$c(P) = \sum_{(i,j)\in P^+} c_{ij} - \sum_{(i,j)\in P^-} c_{ij} \tag{3.17}$$

Si verifica facilmente che

$$c(x(P,\theta)) = c(x \oplus P\theta) = cx + \theta c(P)$$

Theorem 3.3.1 struttura degli Psudoflussi

Siano x e y due pseudoflussi qualunque. Allora esistono $k \leqslant n + m$ cammini aumentanti P1 , . . . , Pk , tutti per x, di cui al più m sono cicli, tali che:

$$\begin{aligned} z1 &= x \\ z_{i+1} &= zi \oplus \theta iPi \\ z_{k+1} &= y \quad 1 \leqslant i \leqslant k \\ 0 &\leqslant \theta_i \leqslant \theta(P_i, z_i) \end{aligned} \tag{3.18}$$

Inoltre, tutti i Pi hanno come estremi dei nodi in cui lo sbilanciamento di x è diverso da quello di y.

3.3.3 minimalità

Definition 3.3.9: pseudoflusso minimale

si definisce pseduofluddo minimale è uno pseudoflusso x che abbia costo minimo tra tutti gli pseudoflussi aventi lo stesso vettore di sbilanciamento ex

Occorre tra gli algoritmi che vedremo preservare la minimalità evitando di aumentare il costo flusso indiscriminatamente, Un problema centrale è quindi quello di determinare quali siano le operazioni di aumento lecite e quali siano le proprietà sui flussi che esse garantiscano.

Lenma 3.3.1

Uno pseudoflusso (rispettivamente, un flusso ammissibile) è minimale (rispettivamente, ottimo) sse non esistono cicli aumentanti di costo negativo

Dimostrazione:

- $\Rightarrow$ Per contrapposizione: se esiste un ciclo aumentante di costo negativo in G_x, applicarlo fa diminuire il costo senza alterare lo sbilanciamento, in contraddizione con la minimalità di x.
- $\Leftarrow$ Ancora per contrapposizione: supponiamo che x non sia minimale, ossia che esista y con $c_y < c_x$ e $e_y = e_x$. Allora per il teorema sugli pseudoflussi possiamo scrivere $y = x \oplus \theta_1 P_1 \oplus \ldots \oplus \theta_n P_n$, dove $\theta_i > 0$ e ciascun P_i è un ciclo. Da $c_y < c_x$ discende però che:

  $$c_x > c_x + \theta_1 c(P_1) + \ldots + \theta_n c(P_n)$$

  e quindi, è ovvio, che $c(P_i) < 0$ per qualche i.

Si può inoltre dimostrare un altro teorema

Theorem 3.3.2

Sia x uno pseudoflusso minimale e sia P un cammino aumentante rispetto ad x avente costo minimo tra tutti i cammini che uniscono un nodo di O_x ad un nodo di D_x. Allora, qualunque sia $\theta \leqslant \theta(x,P)$, abbiamo che $x(\theta,P) = x \oplus \theta P$ è ancora pseudoflusso minimale.

Dimostrazione:

- Siano s e t i vertici che P collega. Supponiamo che $\theta \leqslant \theta(x, P)$ e che y sia un qualunque pseudoflusso con vettore di sbilanciamento $e_{x(\theta,P)}$.
- Per il Teorema sulla struttura degli pseudoflussi esistono:
  - k cammini aumentanti $P_1, \ldots, P_k$ rispetto a x, tutti da s a t;
  - h cicli aumentanti $C_1, \ldots, C_h$ rispetto a x.

  tali che $y = x \oplus \theta_1 P_1 \oplus \ldots \oplus \theta_k P_k \oplus \mu_1 C_1 \oplus \ldots \oplus \mu_h C_h$, (dove tutti gli θ_i, μ_j sono positivi).
- Deve essere, per ragioni che hanno a che fare con lo sbilanciamento, che $\sum_{1 \leqslant i \leqslant k} \theta_i = \theta$.
- Poiché x è minimale, $c(C_i) \geqslant 0$.
- Siccome P ha costo minimo, $c(P_i) \geqslant c(P)$.
- Di conseguenza:

$$cy = cx + \theta_1 c(P_1) + \ldots + \theta_k c(P_k) + \mu_1 c(C_1) + \ldots + \mu_h c(C_h) \geqslant cx + \theta c(P) = cx(\theta, P).$$

3.4 Problemi di Accoppiamento

- **Accoppiamento di Massima Cardinalita'**
- **Accoppiamento di Costo Minimo**

3.4.1 Massima Cardinalita'

Possiamo vederlo come un problema di flusso massimo con piu' sorgenti e piu' pozzi (ma questo l'abbiamo gia risolto). Bisogna un po' riflettere su come attribuire le capacita', in questo caso abbiamo due valori: 0 o 1 (non sceglere o scegliere).
Ogni volta che abbiamo un flusso ammissibile intero, abbiamo un accoppiamento. Dato che non abbiamo flussi frazionari, la proprieta' di unico arco e' rispettata perche' sarebbe impossibile avere un grafo bilanciato con piu' archi per nodo.
Il valore massimo teorico sara' $min\{|O|, |D|\}$, dato che non hanno necessariamente la stessa cardinalita'. Non e' necessario usare EK, dato che non c'e' il rischio delle situaizoni patologiche che rallentano FF.

3.4.2 Costo Minimo

In questo caso, vediamo il problema come uno di flusso di costo minimo, dove i nodi in O hanno sbilanciamento -1 e quelli in D 1, dove ogni arco ha un certo costo

Chapter 4

Programmazione Lineare

4.1 Geometria della PL

Ci concentriamo su algoritmi *senza* vincoli di iterezza. Come accennato in precedenza, questi sono piu' facili da risolvere rispetto alle varianti intere dato che, anche se l'insieme di soluzioni e' infinito, e' possibile guardare le caratterisctiche geometriche delle soluzioni ammissibili in modo da non dover guardarle tutte, ma solo alcune principali.
I vincoli lineari definiscono un'"area" di soluzioni ammissibili, che sara' definita da un poliedro (che puo' anche essere infinito) in quanto i vincoli sono lineari. Anche la funzione obbiettivo e' lineare, quindi guardando ogni valore che puo' assumere, forma una "retta" che puo' intersecare la regione ammissibile. L'ultima retta che forma la funzione obbiettivo che interseca quest'area (che ha quindi il valore ottimo) interseca per forza uno dei vertici che la delimitano.
Quindi ci basta controllare il valore della FO ai vertici per decidere. Ovviamente, il piano non e' per forza bidimensionale, quindi questa intuizione ci abbandona quando abbiamo piu' variabili. Dobbiamo dimostrarlo quindi usando la MATEMATICA

4.1.1 Nozioni Preliminari

Natura matriciale di oggetti geometrici in piu' dimensioni

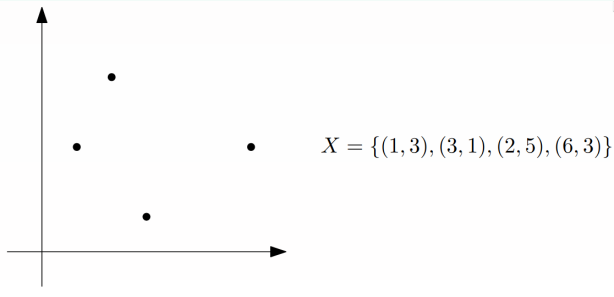
- Iperpiano: generalizzazione di una retta (in 3 dimensioni e' un piano)
- Semispaizo: zona dello spazio delineata da un iperpiano
- Poliedro: intersezione di un numero finito di semispazi
- Iniseme Convesso: ne fanno parte tutti i poliedri e i semispazi, tracciando una retta fra ogni due punti, questa attraversa solo punti dell'insieme. Ci servera' per il teorema fondamentale.

Le facce ci permettono di devinire i vertici

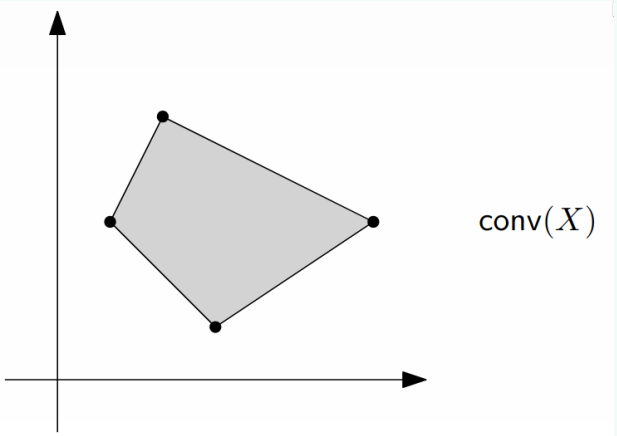
4.2 Inviluppi Convessi

Rappresentazione diveras dei poliedri: per punti. Si parte dai vertici nel costruire il poliedro. Come facciamo a costruire il poliedro minore contenente tutti i punti (che non sono necessariamente vertici). I punti sono vettori nello spazio n-dimensionale e ne prendiamo le combinazioni lineari che sommano a 1 e che siano tutti non negativi (media pesata sei punti, dove i pesi sono positivi e sommano a uno). L'insieme costruito in questo modo si chiama **Invuluppo Convesso** di x ed e' il poliedro piu' piccolo che contenga tutti i punti di x. Non e' solo un poliedro, ma un politopo in quanto e' limitato.

Example 4.2.1
L'inviluppo convesso dei seguenti punti



E' il quadrilatero che ha i punti come vertici. Ha fatto un discorso con triangoli e piani, va beh viene questo:



Chapter 5

Esercitazioni

5.1 Problemi e modelli

5.1.1 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 $j \in \{1, \dots, m\}$, il server 1 riesce ad eseguire un numero di istruzioni..

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 $0 \leqslant x_{ij} \leqslant 1$ si ha:

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

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} x_{ij} s_{ij} \geqslant k$$

funzione obbiettivo

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

5.1.2 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:

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

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

Funzione obbiettivo:

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

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

$$\min y$$

5.1.3 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:

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

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

Funzione obbiettivo:

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

5.1.4 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:

$$\forall i, j \; c_{ij} x_{ij} \leqslant M$$

Funzione obbiettivo:

$$\min M$$

5.1.5 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:

$$\forall i, j \; c_{ij} x_{ij} \geqslant m$$

Funzione obbiettivo:

$$\max m$$

5.1.6 Esercizio valore assoluto

testo

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

Svolgimento

Vincoli

$$x \leqslant 2 \quad -x \leqslant 2$$

Funzione obbiettivo

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

5.1.7 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 aquistati
Vincoli
$0 \leqslant x_1 + x_2 \leqslant 200$
$0 \leqslant y_1 + y_2 + y_3 \leqslant 250$
$z3 \leqslant d_1x_1 + d_2x_2 + d_3y_1 + d_4y_2 + d_5y_3 \leqslant z6$
$z = x_1 + x_2 + y_1 + y_2 + y_3$ Dobbiamo fare in modo che z sia effetivamente il valore che gli vogliamo dare
Funzione Obbiettivo

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

5.1.8 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
$x_1 + x_2 + x_3 \leqslant 10000$
$x_1 \leqslant 0.5x_1 + 0.5x_2 + 0.5x_3$
$x_3 \leqslant 0.25x_2$
Funzione Obbiettivo

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

5.1.9 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
$2x_1 + x_2 + 3x_3 \leqslant 50$
$4x_1 + 2x_2 + 3x_3 \leqslant 50$
$3x_1 + 4x_2 + 2x_3 \leqslant 50$
$15x_1 + 18x_2 + 10x_3 \geqslant 200$
Funzione Obbiettivo

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

5.1.10 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
$x_1 \geqslant 70$
$x_1 + x_2 \geqslant 80$
$x_2 + x_3 \geqslant 50$
$x_3 + x_4 \geqslant 60$
$x_4 + x_5 \geqslant 40$
$x_5 + x_6 \geqslant 30$
Funzione Obbiettivo

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

5.1.11 Esercizio 1.31

Variabili
$\forall i = 1, ..., 7 \quad x_i$ = variabili ogiche che indicano il fatto che un certo abitante a_i sia o meno membro del consiglio
Vincoli
$\sum_{i=1}^{7} x_i = 4$
$\forall i = 1, ..., 4 \sum_{x \in C_i} x = 1$

5.1.12 Esercizio 1.32

Variabili
x_i = variabili logiche che mi dicono se scelgo i
Funzione Obbiettivo

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

Vincoli
$\sum_{i=1}^{n} x_i = 11$
$1 \leqslant \sum_{i \in C} x_i \leqslant 5$
$1 \leqslant \sum_{i \in P} x_i \leqslant 1$
$1 \leqslant \sum_{i \in D} x_i \leqslant 6$
$1 \leqslant \sum_{i \in A} x_i \leqslant 4$
$\forall i = 1, ..., m. \quad 0 \leqslant \sum_{i \in L_i} x_i \leqslant 1$

5.1.13 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 obbiettivo
$min \quad \sum_{k \in K} \sum_{i \in I \cup J} x_{ik} c_{ik} b_k$
Vincoli
$\forall k \in K. \quad \sum_{i \in I \cup J} x_{ik} = 1$
$\sum_{i \in I} x_{ik} \geqslant \frac{|I|}{2}$ non va bene! possiamo contare piu' volte lo stesso impianto che pero' funziona su diversi mercati -¿ variabili ausiliarie
$\forall i \in I \cup J. \quad y_i \leqslant \sum_{k \in K} x_{ik}$ se tutti 0, allora y 0 $y_i \geqslant x_{ik}$ disgiunzione logica generalizzata a k elemetni
$2 \sum_{i \in I} y_i \geqslant |I|$
$2 \sum_{i \in J} y_i \geqslant |J|$

5.1.14 Esercizio 1.7

Testo

Un'azienda di trasporti ha bisogno di acquistare n litri di carburante ogni mese. Affinch´e 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 pi'u 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*

$$x_i = \text{"litri da acquistare di tipo i"}$$

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

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

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

Vincoli

$$x_A + x_B = n$$

$$\frac{k_A \cdot x_A + k_b \cdot x_b}{n} \geqslant k$$

$$0 \leqslant z_i 1 \leqslant ()$$

Funzione Obbiettivo

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

5.1.15 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

$$x_i = \text{"altezza del ripetitore i"}$$

Vincoli

$$\forall i \in [1 \ldots (n-1)] \quad x_{i+1} + a_{i+1} - x_i - a_i \leqslant k$$

$$\forall i \in [1 \ldots (n-1)] \quad -x_{i+1} - a_{i+1} + x_i + a_i \leqslant k$$

Funzione Obbiettivo

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

5.1.16 Esercizio 1.4 - Problema di Manutenzione degli Aerei

Testo

Una compagnia aerea dispone di n aerei $1, \ldots, n$, 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 $a_i \in \{1, \ldots, m\}$. 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

5.1.17 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 5.1: 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 5.2: 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.

5.2 Tutor

5.2.1 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
$\forall i.\ \sum_{j=1}^{m} x_{ij} = p_i$
$\forall j. \sum_{i=1}^{n} x_{ij} \leqslant r_j$
$\forall i, j.\ x_{ij} \leqslant y_{ij} \cdot a_{ij}$

5.2.2 Esercizio 1.33

Dati
n = num camion
$\forall i = 1, ..., n$:
m_i = kilometri massimi percorribili per il camion i
k_i = limite di kilometri per polizza bassa
c_i = costo polizza quando $m_i \leqslant k_i$
d_i = polizza quando $m_i > k_i$
Variabili
p_i = kilometri percorsi dal camion i
b_i = il camion i utilizza la polizza bassa variabile logica
Vincoli
$\sum_{i=1}^{n} p_i \geqslant 2000000$ (possiamo mettere $\geqslant$ e non = dato che minimizzando il valore ottimale sara' quello piu' basso)
$m_i \leqslant b_i k_i + (1 - b_i) m_i$
$m_i > (1 - b_i) k_i$
FO

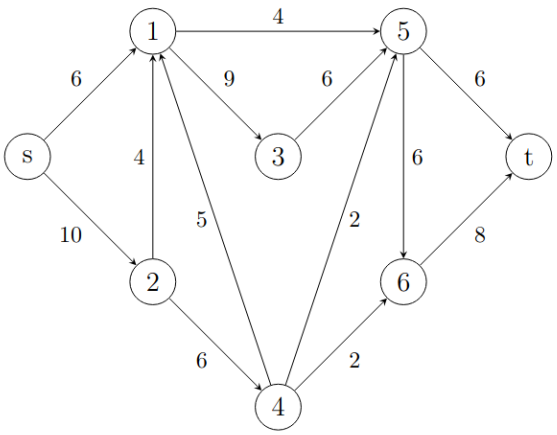
$$min\{\sum_{i=1}^{n} b_i c_i + (1 - b_i) d_i\}$$

5.3 Esercitazioni reti di flusso con Luca

5.3.1 Ex 1

Testo

Si risolva, tramite l'algoritmo di Edmonds e Karp il seguente problema di flusso massimo. Si determini altres'ı un taglio di capacit'a minima



Soluzione

In modo molto pratico, i passi sono:

1. Metti il flusso iniziale a 0 ovunque
2. Costruisci il grafo residuo (nota: il primo grafo residuo sara' uguale al grafo stesso)
3. Trova il cammino di lunghezza minima da s a t sul grafo residuo. Se non esiste, fine.
4. Identifica il valore minore fra gli archi del cammino. Poi, per tutti questi archi, aumentare o diminuire di questo valore il flusso dell'arco corrispondente sul grafo originale, in base a se l'arco residuo e' concorde o discorde all'arco associato.
5. Ritorna al punto 2

Quindi, il meccanismo e':

1. Copia il grafo originale e trova su di esso un cammino, se non c'e' fine
2. Se c'e', calcola il valore minimo fra gli archi e aggiorna correttamente il valore dei flussi sul grafo originale
3. Col grafo OG aggiornato, costruire da questo un altro grafo residuo e ripetere da 2

Note:

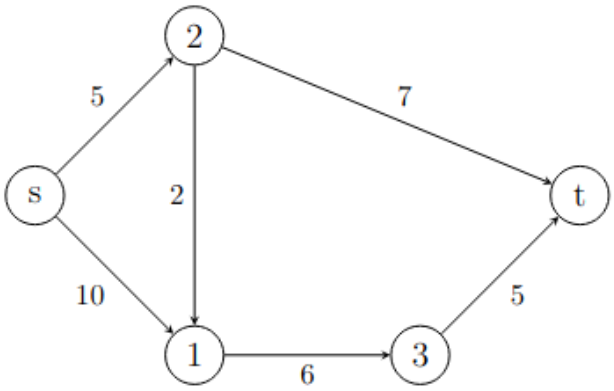
E' possibile risparmiarsi di etichettare i valori degli archi sui grafi residui, usando semplicemente le capacita' ed i flussi del grafo OG per trovare il valore minimo. Se l'arco e' concorde, il suo valore e' capienza - flusso, altrimenti e' semplicemente il flusso.

Facendo correttamente sti passi, arrivi a fine esecuzione con il valore del flusso $v = 14$.

5.3.2 Ex 2

Testo

Si risolva, tramite l'algoritmo di Goldberg-Tarjan il seguente problema MF.



Soluzione

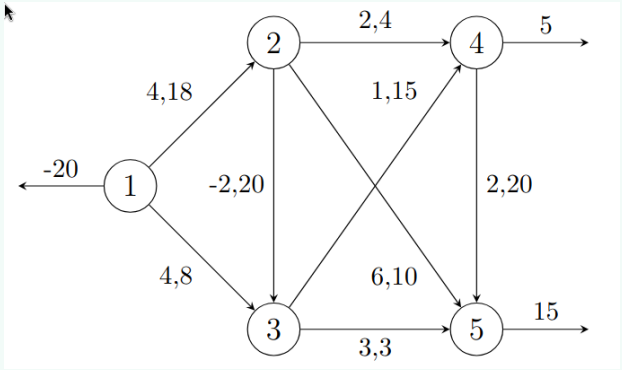
Con Goldberg, si lavora a livello locale e non ci serve il grafo residuo. I passi dell'algoritmo sono:

1. Inizializzazione: pongo tutti i flussi a 0 tranne per gli archi uscenti da s, che metto al massimo della capienza. Poi etichetto tutti i nodi con la loro distanza da t, tranne per il nodo s che prende il numero totale di nodi (la massima distanza possibile)
2. Se non ci sono nodi sbilanciati, fine. Altrimenti, scegli un nodo a caso v fra quelli sbilanciati
3. Se c'e' un arco uscente da v non saturo tale che l'altro nodo ha etichetta di 1 minore rispetto a v, metti tutto il flusso possibile su quell'arco
4. Altrimenti, se c'e' un arco entrnte a v con flusso non nullo tale che l'altro nodo ha etichetta minore di 1 rispetto a v, togliere tutto il flusso possibile da quell'arco
5. Se il nodo e' sbilanciato ma non e' possibile fare nessuna delle mosse sopra, allora tocca aumentare l'etichetta del valore **minimo** finche' esiste un arco (entrante o uscente) tale che l'altro nodo ha etichetta minore di 1.
6. Torna a 2

Se fatto correttamente, il valore del flusso massimo $v = 10$.

Exercise 5.3.1

Si risolva il seguente problema MCF tramite l'algoritmo dei cammini minimi successivi.



Soluzione

I passi dell'algoritmo dei cammini minimi successivi sono:

1. Assegna uno pseudoflusso minimale: per ogni arco, se il costo e' positivo metti 0, se e' negativi metti il massimo possibile.
2. Se tutti i nodi sono bilanciati, fine. Altrimenti vai avanti.

3. Costruisci il grafo residuo, indicando per ogni nodo il valore del suo sbilanciamento e per ogni arco il suo costo. Poi identifica qual'e' il cammino di costo minore che va da un nodo con sbilanciamento positivo a uno con sbilanciamento negativo, se non c'e' il problema non ha soluzioni valide (e' vuoto).

4. Invia il massimo flusso possibile su questo cammino senza cambiare di segno lo sbilanciamento dei due nodi. Quindi ci assicuriamo o di saturare l'arco o di bilanciare uno dei due nodi (nota che lo sbilanciamento dei nodi di mezzo rimane uguale).

5. Aggiornare flussi e sbilanciamenti sul grafo OG e torna a punto 2.

Se fatto tutto bene, penso il costo venga 100.

Exercise 5.3.2

Si risolva il precedente problema tramite l'algoritmo di cancellazione dei cicli.
Soluzione
I passi dell'algoritmo di cancellazione dei cicli sono:

1. Fai EK per trovare il flusso massimo ammissibile. Se questo fallisce, fine.

2. Crea il grafo residuo e cerca un ciclo con valore negativo, se non c'e' allora il flusso e' ottimo.

3. Altrimenti, vedere di quando possiamo aumentare il flusso sugli archi del ciclo (concordi aumento, discordi diminuisco) e cambia il flusso (notare che gli sbilanciamenti rimangono uguali)

4. Vai al passo 2.

Se fatto correttamente, dovrebbe venire lo stesso costo minimo di prima.