# Ottimizzazione

*Corso di Laurea in Informatica*

**Prima Parte**

Ugo Dal Lago

Anno Accademico 2017-2018

# Sezione 1

## Introduzione al Corso

# Questo Corso

- Questo è un corso pensato per gli studenti del secondo anno della **Laurea Triennale in Informatica**.
  - Può anche essere seguito da studenti di Matematica e Fisica che abbiano i prerequisiti descritti di seguito.
- Il **carico di lavoro** complessivo è di 6 ECTS, che corrispondono a circa 40 ore di lezioni frontali.
  - Le **esercitazioni** non sono formalmente previste, ma alcune lezioni frontali saranno dedicate alla risoluzione di esercizi, della stessa difficoltà di quelli previsti per la prova scritta.

# Logistica

- **Quattro ore** di lezione a settimana.
  - Il lunedì, dalle 13.30 alle 15.30.
  - Il mercoledì, dalle 16.30 alle 18.30.
- La lezione **inizia** all'orario prestabilito, per poi finire un po' prima.
- Il **ricevimento studenti** non ha un orario prefissato.
  - Previo appuntamento via email (`ugo.dallago@unibo.it`).
- Se avete **dubbi** sui contenuti del corso, potete anche inviarmi una mail.
- Vi invito a fare tutte le **domande** che volete durante la lezione, ma anche offline.

# Modalità d'Esame — I

- L'esame consiste in una **prova scritta**, non in itinere, seguita da una **prova orale**.
- La prova scritta e la prova orale devono essere sostenute nello **stesso appello** d'esame.
- Spesso, il docente può proporre allo studente la registrazione del voto dello scritto **senza la necessità** di svolgere la prova orale. Ovviamente lo studente può in tal caso chiedere di sostenere comunque la prova orale.
- Il **mancato superamento** della prova scritta o della prova orale dà luogo comunque alla registrazione.

# Modalità d'Esame — II

- La **prova scritta** consiste in un certo numero di semplici esercizi e domande.
- Porrò la massima attenzione nel rendere le prove **diverse** le une dalle altre.
  - Si studia il **corso**, non si studia **come superare l'esame**.
  - Di anno in anno, i contenuti del corso possono **cambiare**, quindi prendete le prove degli anni precedenti *cum grano salis*.
- Nella **prova orale**, l'enfasi si sposta un po' verso la teoria: ci saranno più domande di teoria e meno esercizi.

# Contenuti del Corso

- Questo è un corso introduttivo sull'**ottimizzazione combinatoria** e la **ricerca operativa**.
- Più nello specifico, il corso è suddiviso in tre parti:
  1. Una parte introduttiva, in cui si studia la terminologia di base, e si danno tecniche per la **modellizzazione** dei problemi concreti.
  2. Si studieranno poi i principali problemi e algoritmi per la risoluzione di problemi di **flusso** su reti.
  3. Infine, si studierà in dettaglio l'algoritmica di uno dei problemi di ottimizzazione più importanti, ossia la **programmazione lineare** (nelle sue due varianti).
- Se il tempo lo permette, parleremo anche di uno **specifico software** per la risoluzione di problemi di ottimizzazione, imparando ad usarlo e studiandone l'architettura.

# Prerequisiti

- Fondamentale per affrontare questo corso è una conoscenza delle nozioni di base dell'**algoritmica** e dell'**algebra lineare**.
  - Sconsiglio quindi a tutti voi di seguire questo corso senza aver seguito (anche se non superato) i corsi:
    - **Algoritmi e Strutture Dati**;
    - **Algebra e Geometria**.
- Occorre, ad esempio:
  - Sapere cos'è un *algoritmo*, come si misura la *complessità* di un algrotmo, cos'è un *grafo*, come si risolvono i più semplici problemi combinatorici sui grafi (*visita*, *cammini minimi*);
  - Sapere cosa sono i *vettori* e le *matrici*, cosa sia un *sistema di (dis)equazioni lineari*, quando due vettori si dicono *linearmente (in)dipendenti*.
  - ...

# Libri di Testo e Materiale Didattico

- La pagina web del corso è

  `http://www.cs.unibo.it/~dallago/OTT1718/`

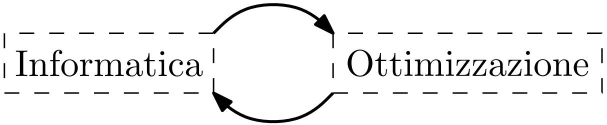
  A partire da essa trovate:
  - Il syllabus del corso;
  - Alcuni riferimenti bibliografici;
  - le prove scritte degli anni passati.
- Durante il corso, seguirò abbastanza fedelmente le note di un corso simile a questo che alcuni colleghi svolgono presso l'Università di Pisa.
  - Le note sono liberamente scaricabili, e comunque accessibili dalla pagina web di **questo** corso.
- Sono disponibili delle **trasparenze**, non necessariamente per tutte le parti del corso.
  - Sicuramente non per le esercitazioni.

# Perché questo corso?

- La teoria dell'ottimizzazione offre tutta una serie di **metodologie** di supporto alle **decisioni**.
  - Tali decisioni possono avere natura **quantitativa**, ma anche **qualitativa**.
- Di conseguenza, l'ottimizzazione combinatoria è un utilissimo strumento per l'informatico, che si trova molto spesso a dover prendere decisioni.

# Perché questo corso?

- La teoria dell'ottimizzazione offre tutta una serie di **metodologie** di supporto alle **decisioni**.
  - Tali decisioni possono avere natura **quantitativa**, ma anche **qualitativa**.
- Di conseguenza, l'ottimizzazione combinatoria è un utilissimo strumento per l'informatico, che si trova molto spesso a dover prendere decisioni.
- I problemi di cui si occupa l'ottimizzazione sono in genere complessi. Di conseguenza è naturale cercare dei meccanismi **automatici** per la sua risoluzione.
  - L'informatica diventa quindi anche **strumento**, oltre che **fine**.



Un'insegnante di informatica ha a sua disposizione 3 personal computers e deve tramite essi compilare i progetti presentati dai suoi 8 studenti. In ciascun PC è istallato il software necessario a compilare ciascun progetto. L'insegnante conosce già i tempi necessari alla compilazione dei progetti, che sono rispettivamente di 3, 4, 4, 5, 5, 6 e 9 minuti. Si formuli il problema di allocare i progetti sulle tre macchine in modo da minimizzare il tempo necessario a completare la compilazione degli 8 progetti. Si tenga ovviamente conto del fatto che i 3 PC possono compilare i progetti in parallelo.

Un'azienda deve strutturare una rete di comunicazione in modo da garantire una banda di 100 Mbps tra una macchina $A$ e una macchina $B$, che si trovano in due sedi diverse dell'azienda. Per mettere in comunicazione $A$ e $B$, l'azienda può far passare i dati attraverso i router $R_1, R_2, R_3, R_4$ e alcune linee dati esistenti tra di essi, che però devono essere affittate. $A$ si può supporre adiacente a $R_1$, mentre $B$ è adiacente a $R_4$ Le capacità (in Mbps) $u_{ij}$ e i costi di affitto (in Euro al Mbps) $c_{ij}$ di ciascuna linea dati *monodirezionale* fra il router $i$ e il router $j$ (dove $i, j \in \{1, 2, 3, 4\}$) sono riassunti di seguito:

$$\begin{array}{cccccc} u_{12} = 70 & u_{13} = 80 & u_{14} = 40 & u_{32} = 70 & u_{24} = 40 & u_{34} = 50 \\ c_{12} = 10 & c_{13} = 20 & c_{14} = 15 & c_{32} = 8 & c_{24} = 12 & c_{34} = 10 \end{array}$$

Si formuli il problema di minimizzare il costo complessivo di affitto delle linee di comunicazione.

Sezione 2

# Problemi di Ottimizzazione

# La Ricerca Operativa

- La **ricerca operativa** e l'**ottimizzazione combinatoria** hanno come oggetto lo studio di **metodologie** a supporto delle **decisioni**.
- I problemi di cui si occupa la ricerca operativa riguardano sempre situazioni in cui occorra **massimizzare** ricavi e profitti o **minimizzare** costi o perdite, in presenza di **risorse limitate**.
- In questo senso, la ricerca operativa è una disciplina a forte contenuto **economico**.

# Il Processo Decisionale

- Il **processo decisionale** si compone delle seguenti cinque fasi:
  1. Individuazione del **problema**;
  2. Raccolta dei **dati**;
  3. Costruzione del **modello**;
  4. Deteminazione di una o più **soluzioni**;
  5. Analisi dei **risultati**.
- Non necessariamente le cinque fasi vengono svolte in sequenza.
- La ricerca operativa e l'ottimizzazione combinatoria si occupano in particolare delle fasi 3 e 4.
  - Sono le fasi che richiedono l'impiego del linguaggio e degli strumenti dell'**informatica** e della **matematica**.

# Modelli

- Un **modello** è una descrizione *astratta*, e scritta nel *linguaggio della matematica*, della parte di realtà utile al processo decisionale.
- Esistono tre tipi di modelli:
  - **Modelli basati sui Giochi**
    - La ricerca di una soluzione viene vista come risultante dall'**interazione** tra due o più agenti, ciascuno dei quali corrisponde ad una delle parti in gioco.
  - **Modelli di Simulazione**
    - Il problema viene studiato **riproducendo** il comportamento del sistema cui il problema si riferisce.
    - La riproduzione si basa sulla generazione di istanze casuali del problema.
  - **Modelli Analitici**
    - Il problema viene descritto attraverso un **modello matematico** il più possibile **fedele** alla situazione reale che si vuole rappresentare...
    - ... ma sufficientemente **astratto** da permettere la determinazione di una soluzione in modo analitico.

# Problemi

- Un **problema** non è nient'altro che una **domanda**, espressa in termini generali, ma la cui risposta dipende da un certo numero di **parametri** e **variabili**.
- Un problema $\mathcal{P}$ viene di solito **descritto** tramite:
  - La descrizione dei suoi parametri e variabili.
  - La descrizione delle caratteristiche che le soluzioni desiderate devono avere.
- Un'**istanza** del problema $\mathcal{P}$ si ottiene specificando dei **valori concreti** per tutti i parametri del problema (ma non per le variabili!).

# Problemi

- Un **problema** non è nient'altro che una **domanda**, espressa in termini generali, ma la cui risposta dipende da un certo numero di **parametri** e **variabili**.
- Un problema $\mathcal{P}$ viene di solito **descritto** tramite:
  - La descrizione dei suoi parametri e variabili.
  - La descrizione delle caratteristiche che le soluzioni desiderate devono avere.
- Un'**istanza** del problema $\mathcal{P}$ si ottiene specificando dei **valori concreti** per tutti i parametri del problema (ma non per le variabili!).
- **Esempi**

$$ax^2 + bx + c = 0 \qquad\qquad 5x^2 - 6x + 1 = 0$$

$$Ax \leq 0 \qquad\qquad \begin{cases} x + y = c \\ x - y = d \end{cases}$$

# Descrivere un Problema

- Un modo molto comune di (cominciare a) descrivere un problema $\mathcal{P}$ è quello di dare l'insieme $\mathbb{F}_{\mathcal{P}}$ delle sue **soluzioni ammissibili**.
  - Di solito $\mathbb{F}_{\mathcal{P}}$ viene specificato dando $\mathbb{G} \supseteq \mathbb{F}_{\mathcal{P}}$ e descrivendo poi dei vincoli che un generico $g \in \mathbb{G}$ deve soddisfare per far parte di $\mathbb{F}_{\mathcal{P}}$.
  - Gli elementi di $\mathbb{G} - \mathbb{F}_{\mathcal{P}}$ sono detti **soluzioni non ammissibili**.
- Talvolta, il problema consiste nel trovare **una** soluzione ammissibile, talvolta occorre andare oltre...

# Descrivere un Problema

- Un modo molto comune di (cominciare a) descrivere un problema $\mathcal{P}$ è quello di dare l'insieme $\mathbb{F}_{\mathcal{P}}$ delle sue **soluzioni ammissibili**.
  - Di solito $\mathbb{F}_{\mathcal{P}}$ viene specificato dando $\mathbb{G} \supseteq \mathbb{F}_{\mathcal{P}}$ e descrivendo poi dei vincoli che un generico $g \in \mathbb{G}$ deve soddisfare per far parte di $\mathbb{F}_{\mathcal{P}}$.
  - Gli elementi di $\mathbb{G} - \mathbb{F}_{\mathcal{P}}$ sono detti **soluzioni non ammissibili**.
- Talvolta, il problema consiste nel trovare **una** soluzione ammissibile, talvolta occorre andare oltre...
- **Esempio**:

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

# Descrivere un Problema

- Un modo molto comune di (cominciare a) descrivere un problema $\mathcal{P}$ è quello di dare l'insieme $\mathbb{F}_{\mathcal{P}}$ delle sue **soluzioni ammissibili**.
  - Di solito $\mathbb{F}_{\mathcal{P}}$ viene specificato dando $\mathbb{G} \supseteq \mathbb{F}_{\mathcal{P}}$ e descrivendo poi dei vincoli che un generico $g \in \mathbb{G}$ deve soddisfare per far parte di $\mathbb{F}_{\mathcal{P}}$.
  - Gli elementi di $\mathbb{G} - \mathbb{F}_{\mathcal{P}}$ sono detti **soluzioni non ammissibili**.
- Talvolta, il problema consiste nel trovare **una** soluzione ammissibile, talvolta occorre andare oltre...
- **Esempio**:

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

# Problemi di Ottimizzazione - I

- I **problemi di ottimizzazione** sono i problemi che studieremo in questo corso.
- Un problema di ottimizzazione $\mathcal{P}$ viene descritto:
  - Dando l'insieme $\mathbb{F}_{\mathcal{P}}$ delle sue soluzioni ammissibili.
  - Specificando una **funzione obiettivo**

    $$c_{\mathcal{P}} : \mathbb{F}_{\mathcal{P}} \to \mathbb{R}$$

    che misuri il **costo** o il **beneficio** di ogni soluzione ammissibile.
- Un **problema** (di ottimizzazione) **di massimo** $\mathcal{P}$ consiste nel deteminare il valore

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

- Un **problema** (di ottimizzazione) **di minimo** $\mathcal{P}$ consiste invece nel deteminare il valore

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

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.
- **Esempio:**
  - $\mathbb{G} = \mathbb{R}$;

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.
- **Esempio:**
  - $\mathbb{G} = \mathbb{R}$;
  - $\mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\}$;

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.
- **Esempio:**
  - $\mathbb{G} = \mathbb{R}$;
  - $\mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\}$;
  - $c_{\mathcal{P}} : \mathbb{R} \to \mathbb{R}, \quad c_{\mathcal{P}}(g) = g^2$;

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.
- **Esempio:**
  - $\mathbb{G} = \mathbb{R}$;
  - $\mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\}$;
  - $c_{\mathcal{P}} : \mathbb{R} \to \mathbb{R}, \quad c_{\mathcal{P}}(g) = g^2$;
  - $Z_{\mathcal{P}} = \max\{x^2 \mid 5x^2 - 6x + 1 = 0\}$;

# Problemi di Ottimizzazione - II

- Ad ogni problema di massimo $\mathcal{P}$ **corrisponde** un problema di minimo $\mathcal{P}'$ tale che $c_{\mathcal{P}'}(g) = -c_{\mathcal{P}}(g)$. Infatti:

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

- Dato $\mathcal{P}$, $Z_{\mathcal{P}}$ è detto **valore ottimo** per $\mathcal{P}$.
- Dato $\mathcal{P}$, un $g^* \in \mathbb{F}_{\mathcal{P}}$ tale che $Z_{\mathcal{P}} = c_{\mathcal{P}}(g^*)$ è detto **soluzione ottima**.
- **Esempio:**
  - $\mathbb{G} = \mathbb{R}$;
  - $\mathbb{F}_{\mathcal{P}} = \{x \in \mathbb{R} \mid 5x^2 - 6x + 1 = 0\}$;
  - $c_{\mathcal{P}} : \mathbb{R} \to \mathbb{R}, \quad c_{\mathcal{P}}(g) = g^2$;
  - $Z_{\mathcal{P}} = \max\{x^2 \mid 5x^2 - 6x + 1 = 0\}$;
  - Valore ottimo? Soluzione ottima?

# Quattro Casi

- **Problema Vuoto**
  - $\mathbb{F}_{\mathcal{P}} = \emptyset$, e per convenzione si assume che $Z_{\mathcal{P}} = \infty$.
  - Non è detto sia triviale rilevarlo.
- **Problema Illimitato**
  - Nel caso di problema di massimo, per ogni $x \in \mathbb{R}$ esiste $g \in \mathbb{F}_{\mathcal{P}}$ con $c_{\mathcal{P}}(g) \geq x$. In tal caso $Z_{\mathcal{P}} = +\infty$.
  - Dualmente nel caso di problema di minimo.
- **Valore Ottimo Finito, ma non Soluzione Ottima Finita**.
  - $Z_{\mathcal{P}}$ esiste finito, ma $c_{\mathcal{P}}(g) \neq Z_{\mathcal{P}}$ per ogni $g$.
  - Esempio: $\inf\{x \mid x > 0\}$.
  - Evitermo accuratamente questi casi.
- **Valore Ottimo Finito, e Soluzione Ottima Finita**.

# Ottimizzazione e Decisione

- Un **problema di decisione** $\mathcal{P}$ consiste semplicemente nel determinare una qualunque $g \in \mathbb{F}_{\mathcal{P}}$ oppure nel concludere che il problema è vuoto, qualora $\mathbb{F}_{\mathcal{P}} = \emptyset$.
- Dato un problema di decisione $\mathcal{P}$, il relativo **problema di certificato** per $\mathbb{G} \supseteq \mathbb{F}_{\mathcal{P}}$ consiste nel nel dire se $g \in \mathbb{F}_{\mathcal{P}}$, data $g \in \mathbb{G}$.
- Dato un problema di decisione, è sempre possibile vedere quest'ultimo come problema di ottimizzazione.
- Il **contrario**? Dato $\mathcal{P}$ problema di ottimizzazione:
  - Si può considerare $\mathcal{R}$ decisionale tale che
  $$\mathbb{F}_{\mathcal{R}} = \{g \in \mathbb{F}_{\mathcal{P}} \mid c_{\mathcal{P}}(g) = Z_{\mathcal{P}}\}.$$
  - Dato $x \in \mathbb{R}$, si può anche considerare $\mathcal{R}_k$ decisionale con
  $$\mathbb{F}_{\mathcal{R}_k} = \{g \in \mathbb{F}_{\mathcal{P}} \mid c_{\mathcal{P}}(g) \leq k\}$$
  (se $\mathcal{P}$ è di minimo, altrimenti il duale).

# Ottimizzazione e Algoritmi

- Un **algoritmo esatto** per $\mathcal{P}$ è un algoritmo che, presa in input un'*istanza* di $\mathcal{P}$, fornisce in output *una* soluzione ottima $g^*$ di $\mathcal{P}$ (se esiste).
  - Gli algoritmi esatti sono esattamente ciò che cerchiamo...
  - ...ma per molti problemi hanno complessità troppo alta.
- Gli **algoritmi euristici** determinano invece una *qualsiasi* soluzione ammissibile e quindi calcolano implicitamente
  - un'approssimazione *superiore* (se il problema è di minimo);
  - un'approssimazione *inferiore* (se il problema è di massimo);

  del valore ottimo.

# Qualità degli Algoritmi Euristici

- In linea di principio, gli algoritmi euristici potrebbero concludere che non esiste soluzione ammissibile *anche se* $\mathbb{F}_{\mathcal{P}} \neq \emptyset$.
  - In altre parole, l'approssimazione può essere arbitrariamente cattiva.
- Dato $\mathcal{P}$ e $g \in \mathbb{F}_{\mathcal{P}}$ definiamo:
  - **Errore Assoluto** di $g$ la quantità:

$$\mathcal{E}_{\mathcal{P}}(g) = c_{\mathcal{P}}(g) - Z_{\mathcal{P}}.$$

  - **Errore Relativo** di $g$ la quantità:

$$\mathcal{R}_{\mathcal{P}}(g) = \frac{\mathcal{E}_{\mathcal{P}}(g)}{|Z_{\mathcal{P}}|} = \frac{c_{\mathcal{P}}(g) - Z_{\mathcal{P}}}{|Z_{\mathcal{P}}|}$$

- Una soluzione $g$ si dice $\varepsilon$**-ottima** se $\mathcal{R}_{\mathcal{P}}(g) \leq \varepsilon$.
- Un algoritmo euristico si dice $\varepsilon$**-approssimato** se produce soluzioni $\varepsilon$-ottime.

# Rilassamenti

- Talvolta anche calcolare l'errore diventa problematico, e quindi si procede risolvendo un problema che è un'**approssimazione** del problema di partenza.
- Dato $\mathcal{P}$ (ad esempio di minimo), un **rilassamento** di $\mathcal{P}$, è un qualunque problema $\overline{\mathcal{P}}$ definito come segue

$$\min\{c_{\overline{\mathcal{P}}}(g) \mid g \in \mathbb{F}_{\overline{\mathcal{P}}}\},$$

  dove $\mathbb{F}_{\overline{\mathcal{P}}} \supseteq \mathbb{F}_{\mathcal{P}}$ e $\forall g \in \mathbb{F}_{\mathcal{P}}.c_{\overline{\mathcal{P}}}(g) \leq c_{\mathcal{P}}(g)$ (e dualmente per i problemi di massimo). Il valore $Z_{\overline{\mathcal{P}}}$ è inferiore a $Z_{\mathcal{P}}$.
- Osserviamo che:
  - I rilassamenti, spesso, ammettono soluzioni algoritmiche di complessità inferiore.
  - Se la soluzione ottima $g^*$ di $\overline{\mathcal{P}}$ soddisfa $g^* \in \mathbb{F}_{\mathcal{P}}$ e $c_{\overline{\mathcal{P}}}(g^*) = c_{\mathcal{P}}(g^*)$, allora

$$c_{\overline{\mathcal{P}}}(g^*) = Z_{\overline{\mathcal{P}}} \leq Z_{\mathcal{P}} \leq c_{\mathcal{P}}(g^*) = c_{\overline{\mathcal{P}}}(g^*).$$

Sezione 3

# Modelli

# Dai Problemi ai Modelli

- Una volta individuato il problema, occorre *classificarlo*, in modo da poter riconoscerlo come problema di un certo tipo…
  - …e quindi magari utilizzare algoritmi efficienti per la risoluzione del problema.

# Dai Problemi ai Modelli

- Una volta individuato il problema, occorre *classificarlo*, in modo da poter riconoscerlo come problema di un certo tipo...
  - ...e quindi magari utilizzare algoritmi efficienti per la risoluzione del problema.
- Una categoria di problemi dello stesso tipo si dice anche **modello**.
- In questa parte del corso studieremo un particolare modello, ovvero quello della **programmazione lineare**.

# Programmazione Lineare — I

- Un **problema di programmazione lineare** (PL) è un problema di ottimizzazione definito dando:
  - Un numero finito $n \in \mathbb{N}$ di *variabili reali*

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

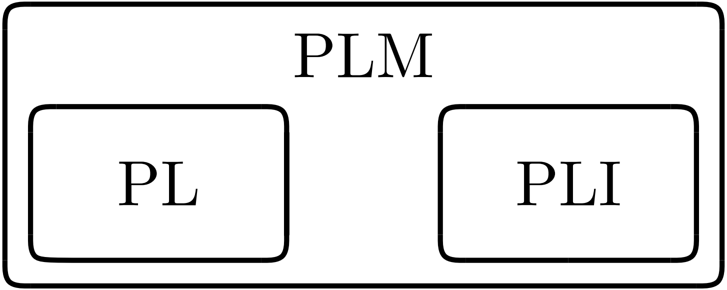
  - Una *funzione obiettivo* $f : \mathbb{R}^n \to \mathbb{R}$ nella forma

$$f(x) = cx.$$

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

$$ax = b \qquad ax \leq b \qquad ax \geq b$$

    dove $a \in \mathbb{R}^n$ e $b \in \mathbb{R}$.
- Talvolta risulta molto utile assumere che $x \in \mathbb{N}^n$, ovvero che le soluzioni ammissibili siano (vettori di) *numeri naturali*. Si parla in questo caso di **programmazione lineare intera** (PLI).


PLM
PL
PLI

# Programmazione Lineare — III

- Un problema di PL può **sempre** essere espresso nella forma seguente:
$$\max\{cx \mid Ax \leq b\}$$
dove $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$. Infatti:
  - Se il problema $\mathcal{P}$ è un problema di minimo, basta considerare $f(x) = (-c)x$.
  - Ogni vincolo $ax = b$ diventa la coppia di vincoli $ax \leq b$ e $ax \geq b$.
  - Ogni vincolo $ax \geq b$ è equivalente a $(-a)x \leq (-b)$.

# Programmazione Lineare — Esempi

- **Esempio:** Pianificazione della Produzione;
- **Esempio:** Il Problema della Fonderia.

# Programmazione Lineare Intera

- Nella programmazione lineare, le variabili rappresentano *quantità*.
- Nella PLI, invece, le variabili possono essere:
  - **Quantitative**, ovvero rappresentare quantità.
  - **Logiche**, ovvero rappresentare valori binari, booleani.
- Una variabile $x$ è *logica* se vale che

$$x \in \mathbb{N} \qquad 0 \leq x \qquad x \leq 1$$

- Le variabili logiche possono essere utilizzate per modellare:
  - L'**assegnamento** di una risorsa ad un task;
  - Il fatto che una certa attività **si debba** eseguire **oppure no**.

# Programmazione Lineare Intera — Esempi

- **Esempio:** Lo Zaino;
- **Esempio:** Albero di Copertura Minimo;
- **Esempio:** Il Commesso Viaggiatore.

# Relazioni Logiche

- Spesso le **relazioni** intercorrenti tra le variabili logiche hanno esse stesse natura logica.
  - Ad esempio, $x$ vale se e sole se $y$ e $z$ valgono.
- Possiamo modellare **tutte** le relazioni logiche tramite semplici vincoli lineari:

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

$$x = 1 - y.$$

**Implicazione** ($z = (x \rightarrow y)$)

$$\begin{aligned} x + z &\geq 1; \\ z &\geq y; \\ x + z &\leq 1 + y. \end{aligned}$$

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

$$\begin{aligned} z &\leq x; \\ z &\leq y; \\ z &\geq x + y - 1. \end{aligned}$$

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

$$\begin{aligned} z &\geq x; \\ z &\geq y; \\ z &\leq x + y. \end{aligned}$$

- Conseguenza: il problema è **NP**-difficile.

# Vincoli di Assegnamento — I

- Un tipo di vincoli che si presentano spesso in concreto sono i **vincoli di assegnamento**.
  - Possono essere trattati in modo molto agevole con la PLI
- Si parte da:
  - Un insieme $N = \{1, \ldots, n\}$ di **oggetti**;
  - Un insieme $V = \{1, \ldots, m\}$ di **luoghi**.
- L'idea è quella di rappresentare le varie **condizioni** in cui assegnare oggetti a luoghi.
- La variabile $x_{ij}$ (dove $1 \leq i \leq n$ e $1 \leq j \leq m$) prende valori in $\{0, 1\}$ e modella il fatto che l'$i$-esimo oggetto è stato assegnato al $j$-esimo luogo.

# Vincoli di Assegnamento — II

- **Vincoli di Semi-Assegnamento**: ogni oggetto è assegnato ad un luogo.

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad (1 \leq i \leq n).$$

- **Insiemi Ammissibili**
  - Talvolta, ogni oggetto $i \in \{1, \dots, n\}$ può essere assegnato ad uno specifico insieme $B(i) \subseteq V$ di luoghi.
  - In tal caso, $x_{ij}$ esiste solo se $i \in B(i)$.
  - Il vincolo di semi-assegnamento diventa

$$\sum_{j \in B(i)} x_{ij} = 1 \qquad (1 \leq i \leq n).$$

# Vincoli di Assegnamento — III

- **Vincoli di Assegnamento**: ogni oggetto è assegnato ad un luogo e ad ogni luogo è assegnato un oggetto.

$$\sum_{j=1}^{m} x_{ij} = 1 \quad (1 \leq i \leq n) \qquad\qquad \sum_{i=1}^{n} x_{ij} = 1 \quad (1 \leq j \leq m).$$

- **Ordinamento**.
  - I vincoli di *assegnamento* (certo non quelli di semi-assegnamento) possono essere un modo per imporre che gli $n$ lavori siano eseguiti in un certo ordine.
  - La variabile $x_{ij}$ indicherà quindi se l'$i$-esimo lavoro è effettuato come $j$-esimo (se vale 1) o meno (se vale 0).

# Vincoli di Assegnamento — Esempi

- **Esempio:** Assegnamento di Costo Minimo;
- **Esempio:** Ordinamento di Lavori su Macchine;

# Selezione di Sottoinsiemi — I

- Sia $N = \{1, \ldots, n\}$ un insieme finito di elementi e sia poi $F = \{F_1, \ldots, F_m\}$ una famiglia di suoi sottoinsiemi, dove $F_i \subseteq N$.
- Ad ogni $F_j$ (con $1 \leq j \leq m$) associamo un costo $c_j$.
- Vogliamo determinare $D \subseteq F$ di **costo minimo**, tra tutti i sottoinsiemi di $F$ che soddisfano certi vincoli.

## Selezione di Sottoinsiemi — I

- Sia $N = \{1, \ldots, n\}$ un insieme finito di elementi e sia poi $F = \{F_1, \ldots, F_m\}$ una famiglia di suoi sottoinsiemi, dove $F_i \subseteq N$.
- Ad ogni $F_j$ (con $1 \leq j \leq m$) associamo un costo $c_j$.
- Vogliamo determinare $D \subseteq F$ di **costo minimo**, tra tutti i sottoinsiemi di $F$ che soddisfano certi vincoli.
- Tale situazione può essere **rappresentata** con una matrice $A = (a_{ij}) \in \{0, 1\}^{n \times m}$ dove

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

- Il vettore delle **variabili** avrà la forma $x = (x_1, \ldots, x_m)$ dove

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

# Selezione di Sottoinsiemi — II

- La **funzione obiettivo**, da minimizzare, sarà sempre $\sum_{i=1}^{m} c_j x_j$.
- I **vincoli** dipendono invece dal problema. Esempi:
  - **Problema di Copertura**: ognuno degli elementi di $N$ sta in *almeno* in uno degli elementi di $D$. Quindi:

$$\sum_{j=1}^{m} a_{ij} x_j \geq 1 \quad (1 \leq i \leq n).$$

  - **Problema di Partizione**: ognuno degli elementi di $N$ sta in *esattamente* uno degli elementi di $D$. Quindi:

$$\sum_{j=1}^{m} a_{ij} x_j = 1 \quad (1 \leq i \leq n).$$

  - **Problema di Riempimento**: ognuno degli elementi di $N$ sta in *al più* uno degli elementi di $D$. Quindi:

$$\sum_{j=1}^{m} a_{ij} x_j \leq 1 \quad (1 \leq i \leq n).$$

# Variabili a Valori Discreti

- Spesso le variabili in gioco sono vincolate a prendere il loro valore da un insieme che:
  - **Non** è semplicemente $\{0,1\}$.
  - **Non** è $\mathbb{N}$.
  - **Non** è un intervallo.
- Per esempio, potremmo essere interessati a vincolare $x$ a stare nell'insieme $\{v_1,\ldots,v_n\}$ dove i $v_i$ sono valori reali distinti.
- In tal caso, porcederemo introducendo $n$ variabili $y_1,\ldots,y_n$ vincolate come segue:

$$y_i \in \{0,1\} \qquad \sum_{i=1}^{n} y_i = 1 \qquad x = \sum_{i=1}^{n} v_i y_i$$

# Variabili a Valori Discreti — Esempio

- **Esempio**: Progetto di Reti

# Minima Quantità Positiva Prefissata

- Quando una variabile $x$ rappresenta un certo livello di produzione, capita spesso che il valore di tale variabile debba viaggiare in un insieme

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

  dove 0 rappresenta l'**assenza** di produzione, mentre l'intervallo $[l, u]$ rappresenta i possibili **livelli** di produzione quando il meccanismo è attivo.
- Per modellare tutto questo:
  - Introduciamo una variabile logica $y \in \{0, 1\}$ che indica la presenza o meno di produzione.
  - I vincoli saranno poi

$$ly \leq x \qquad x \leq uy.$$

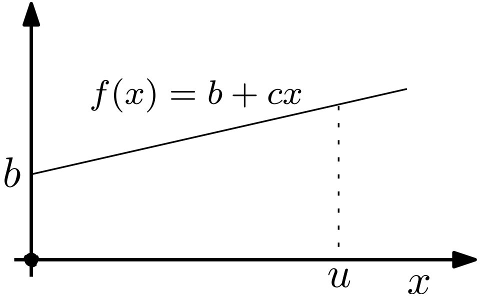
  - **Correttamente**, se $y = 0$, allora $x = 0$. Altrimenti, $l \leq x \leq u$.

# Funzione con Carico Fisso — I

- Si supponga di lavorare con la seguente funzione **con carico fisso** (dove $b, c > 0$), da minimizzare:

$$f(x) = \begin{cases} 0 & \text{se } x = 0; \\ b + cx & \text{se } x \in (0, u]. \end{cases}$$

- La situazione è dunque la seguente:



# Funzione con Carico Fisso — II

- Introdiamo, al solito, una variabile logica $y$, che rappresenta, intuitivamente, la **presenza di produzione**.
  - Occorreranno i seguenti due vincoli:

$$0 \leq x \qquad x \leq yu$$

- La funzione sarà rappresentata tramite una nuova funzione

$$g(x,y) = by + cx.$$

- Abbiamo infatti che

$$g(0,0) = 0; \qquad g(x,1) = b + cx.$$

- Si noti che se $y = 0$, allora $x = 0$, ma che **non vale** il viceversa. In altre parole i due valori $g(0,1)$ e $g(0,0)$ sono diversi, ma corrispondono entrambi a soluzioni ammissibili (ovvero $(0,1)$ e $(0,0)$). La funzione va però **minimizzata** e quindi si sceglie correttamente $g(0,0)$.

# Vincoli di Soglia — Esempio

- **Esempio**: Ordinamento di Valori su Macchine

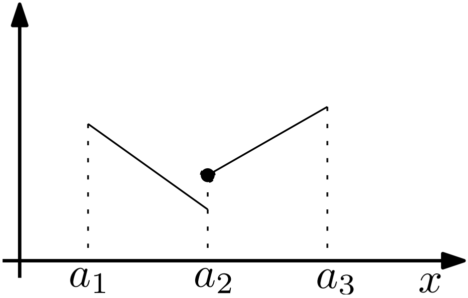
# Come Rappresentare il Valore Assoluto

- Possiamo avere a che fare con il valore assoluto:
  - **Nei Vincoli**.
    - Il vincolo $|g(x)| \leq b$ può essere espresso come la congiunzione di due vincoli:
      $$g(x) \leq b; \qquad -g(x) \leq b.$$
      (se $b$ è un reale positivo).
    - In casi più complessi non è sempre possibile ridurre il vincolo alla congiunzione di vincoli lineari.
  - **Nella Funzione Obiettivo**.
    - Ad esempio, il problema di massimizzare $|f(x)|$, con $x \in X$, può essere risolto risolvendo i due seguenti problemi
      $$\max\{f(x) \mid x \in X\} \qquad \max\{-f(x) \mid x \in X\}$$
      e confrontando i rispettivi valori ottimi.
    - Analogamente per i problemi di minimizzazione

# Funzioni Lineari a Tratti — I

- Un problema molto interessante è proprio quello di rappresentare funzioni lineari **a tratti**, eventualmente con l'ausilio di variabili logiche.
- Supponiamo di essere nella situazione seguente:

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



# Funzioni Lineari a Tratti — II

- In analogia con quanto fatto per il carico fisso, introduciamo due **variabili logiche ausiliarie** $y_1, y_2$ con il significato seguente

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

- Inoltre, introduciamo due altre **variabili quantitative ausiliarie** $z_1$ e $z_2$ tali che:

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

- Tutto questo è catturabile tramite i **vincoli** seguenti:

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

# Funzioni Lineari a Tratti — III

- A questo punto possiamo rappresentare la funzione $f$ attraverso $g : \mathbb{R}^4 \to \mathbb{R}$, definita come segue:

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

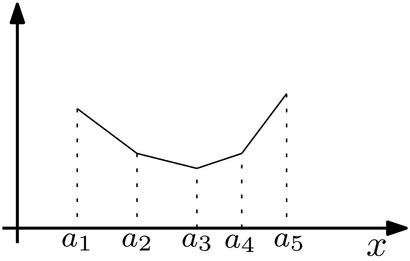
- Il valore di $f$ in ogni punto $x \in [a_1, a_3]$ è rappresentato **univocamente** da una quadrupla di valori $(z_1, z_2, y_1, y_2)$.
  - L'unica eccezione è $x = a_2$, che corrisponde alle due quadruple seguenti:

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

    dei quali solo il **primo** è accettabile
  - Se supponiamo il prolema sia un problema di *minimo*, allora possiamo considerare il problema come benigno, visto che nel punto di discontinuità $f$ cresce.
- Tutto questo può essere generalizzato al caso di funzioni lineari a $n > 2$ tratti.

# Funzioni Convesse — I

- Nelle funzioni lineari a tratti, la necessità di usare *variabili logiche* (e quindi di procedere per casi) deriva dalla **non-convessità** della funzione considerata.
- Una funzione $f$ lineare a $n$ tratti si dice **convessa** se valgono le seguenti due condizioni:
  - $f$ deve essere *continua*, ossia $b_{i+1} + c_{i+1}a_{i+1} = b_i + c_i a_{i+1}$ per ogni $1 \leq i < 1$.
  - La derivata di $f$ deve essere *non-decrescente*, ossia $c_{i+1} \geq c_i$ per ogni $1 \leq i < 1$.
- Esempio:



# Funzioni Convesse — II

- Se $f$ è convessa, la **minimizzazione** di $f$ diventa la minimizzazione di

$$g(z_1, \ldots, z_n) = b_1 + c_1 a_1 + \sum_{i=1}^{n} c_i z_i$$

con i seguenti vincoli:

$$0 \leq z_i \leq a_{i+1} - a_i \qquad x = a_1 + \sum_{i=1}^{n} z_i$$

- Se la funzione $f$ ha ottimo $x^*$, tale valore ottimo può sempre essere ricostruito usando i segmenti di indice inferiore, proprio grazie alla convessità.

# Programmi Lineari in `GNU MathProg`

- `GNU MathProg` è un linguaggio in cui è possibile scrivere dei modelli di programmazione lineare.
- É il linguaggio di input id un certo numero di solver, tra cui `GLPK`.
- Si può sperimentare anche via web:

  `http://www3.nd.edu/~jeff/mathprog/`

```
# Esercizio 1.26
var x1>=0;
var x2>=0;
var y1>=0;
var y2>=0;
var y3>=0;
s.t. c1: x1 + x2 <= 200;
s.t. c2: y1 + y2 + y3 <= 250;
s.t. c3: 5.8 * x1 + 3.1 * x2 - 1.0 * y1 + 1.2 * y2 + 2.0 * y3 >= 0;
s.t. c4: 2.8 * x1 + 0.1 * x2 - 4.0 * y1 - 1.8 * y2 - 1.0 * y3 <= 0;
maximize obj: 50 * x1 + 30 * x2 + 20 * y1 + 40 * y2 + 35 * y3;
solve;
display x1, x2, y1, y2, y3;
end;
```

```
Display statement at line 13
x1.val = 159.259259259259927
x2.val = 40.7407407407407407
y1.val = 0
y2.val = 250
y3.val = 0
```

```
# Esercizio 1.29
var x12>=0 integer;
var x23>=0 integer;
var x34>=0 integer;
var x45>=0 integer;
var x6>=0 integer;
s.t. c1: x12 >= 70;
s.t. c2: x12 + x23 >= 80;
s.t. c3: x23 + x34 >= 50;
s.t. c4: x34 + x45 >= 60;
s.t. c5: x45 >= 40;
s.t. c6: x6 >= 30;
minimize obj: x12 + x23 + x34 +x45 +x6;
solve;
display x12,x23,x34,x45,x6;
end;
```

```
Display statement at line 15
x12.val = 70
x23.val = 30
x34.val = 20
x45.val = 40
x6.val = 30
```