Modelli

deve essere lineare (NO variabili logiche moltiplicate o funzioni di max() e min())

- Trovare una funzione obbiettivo (max, min)
- Aggiungere variabili quantitative o logiche
- Trovare vincoli in base al problema
  - vincoli di copertura: quando $\sum x_i \geq 1$ ovvero ci assicuriamo che ogni "elemento" sia associato ad almeno una "destinazione". Se al posto di "$\geq$" metto "$=$" diventa un problema di partizione
  - vincoli di interezza: affermano che le variabili in tali vincoli sono intere ($\in \mathbb{Z}$)
  - vincoli di soglia: permettono di lavorare anche con i valori assoluti e variabili ausiliarie (approssimazioni superiori o inferiori del valore cercato).
  - vincoli di semiassegnamento: ciascun "oggetto" viene assegnato ad uno e uno solo "elemento" (NON è suriettiva).

Grafi (Modelli)

- Trovare la sorgente (s) e il pozzo (t) (possono essere molteplici)
- Trasferire il flusso da s a t considerando c_{ij} (costo attivazione arco) b_i (bilanci dei nodi).
- Trovare una funzione obbiettivo (max, min)
- Trovare i vincoli relativi al problema
  - Vincoli di conservazione del flusso: $\sum_{j \in BN(i)} x_{ij} - \sum_{j \in FN(i)} x_{c_j} = b_i \quad (\sum_{i \in N} b_i = 0)$
  - Vincoli di capacità: $0 \leq x_{ij} \leq u_{ij} \quad (i,j) \in A$
- Se immaginiamo il flusso da mandare come dei pacchetti allora s dovrà inviare esattamente P pacchetti quindi come bilancio avrà $-P$ mentre t dovrà ricevere P pacchetti quindi come bilancio avrà P.

$$\hookrightarrow b_i = \begin{cases} -P & i = s \\ 0 & i \neq s,t \\ P & i = t \end{cases}$$

Grafi

- Costruzione di un **albero dei cammini minimi** (contiene il cammino di costo minimo da s a t). Per verificare che un albero sia dei cammini minimi bisogna controllare le **condizioni di Bellmann** ($d_i + c_{ij} \geq d_j$) per ogni $(i,j) \in A$. Ovviamente $d_r = 0$ dove r è la radice e d_i è il costo del cammino da r ad i.
  Con l'aggiunta di un **arco fittizio** di costo $M = (n-1)C_{max} + 1$ possiamo trovare un albero dei cammini minimi.
  Il **costo** di un albero dei cammini minimi è la somma dei bilanci dei suoi nodi.
- Per "visitare" un grafo possiamo usare **Dijkstra** se tutti gli archi hanno costo ≥ 0 e **Bellmann-Ford** se alcuni sono < 0.
- Costruzione di un **albero di copertura di costo minimo** (sottografo connesso privo di cicli). Esistono due metodi per verificare che un albero sia di copertura di costo minimo:
  - **Ottimalità per tagli**: (N, A_T) è un albero di copertura di costo minimo $\iff$ il costo di ciascun $(i,j) \in A_T$ è $\leq$ del costo di ciascun arco del taglio che si viene a formare rimuovendo (i,j).
  - **Ottimalità per cicli**: (N, A_T) è un albero di copertura di costo minimo $\iff$ il costo di ciascun $(i,j) \notin A_T$ è $\geq$ al costo di ciascun arco del ciclo che si viene a formare aggiungendo (i,j).
- **Taglio**: Partizione di un grafo in 2 insiemi (N', N'') t.c. $N', N'' \subseteq N$ $N' \cap N'' = \emptyset$, $N' \cup N'' = N$.
- Algoritmi di inserzione e cancellazione di archi:
  - **Algoritmo di Kruskal**: Si ordinano gli archi per costo non decrescente e vengono esaminati con tale ordine. Se l'arco analizzato non forma un ciclo con gli archi già inseriti lo si aggiunge, altrimenti lo si scarta.
  - **Algoritmo di Prim**: Fa esclusivamente inserzioni. Preso un taglio che esclude un nodo i dagli altri, scegliamo l'arco di costo minimo tra gli

archi del taglio inserendolo, continuo l'algoritmo creando un nuovo taglio che separa il nodo i ed il nodo j (che abbiamo collegato precedentemente) dal resto del grafo e proseguiamo scegliendo l'arco di costo minimo del taglio che si crea, e così via.

Grafi (Problemi di flusso massimo)

- Ogni nodo ha un bilancio b_i, mentre ogni arco ha un flusso già presente x_{ij} ed una capacità u_{ij}
- Se $x_{ij} = u_{ij}$ l'arco si dice **saturo**, mentre $u_{ij} - x_{ij}$ è la **capacità residua**
- In un grafo è possibile trovare un **cammino aumentante** in grado di aumentare il valore del flusso di una quantità $\theta = \min\{u_{ij} - x_{ij} : (i,j) \in P\}$ dove "P" è il cammino. (>0)
- Gli archi che "seguono" il verso del cammino vengono detti **concordi** mentre quelli che vanno nella direzione opposta sono detti **discordi**.
- La costruzione del **grafo residuo** avviene partendo da un grafo e inserendo nel grafo residuo gli archi non saturi nello stesso verso del grafo originale e gli archi non vuoti (che contengono flusso) in verso opposto e non inserendo gli archi saturi dato che da essi non può circolare ulteriore flusso.
- (N_s, N_T) è un **taglio che separa s da t** se $s \in N_s$ e $t \in N_t$ ndichiamo con $u(N_s, N_t) = \sum_{(i,j) \in A_T} u_{ij}$ la **capacità del taglio** e con $x(N_s, N_t) = \sum_{(i,j) \in A^+} x_{ij} - \sum_{(i,j) \in A^-} x_{ij}$ il **flusso nel taglio**.
- Il massimo valore del flusso ammissibile è uguale alla minima capacità dei tagli che separano s da t.

Grafi (Problemi di flusso di costo minimo)

- Ogni arco avrá un costo c_{ij} e una capacitá u_{ij}, con l'utilizzo di uno **pseudoflusso** (grafo dove ogni arco ha un flusso x_{ij}) possiamo costruire un **grafo residuo** con lo stesso procedimento dei pb. di flusso di costo massimo ma se $x_{ij} < u_{ij}$ nell'arco metteremo $u_{ij} - x_{ij}$ e costo c_{ij}, se $x_{ij} > u_{ij}$ metteremo x_{ij} e costo $-c_{ij}$.
- A partire da uno pseudoflusso x é possibile calcolare gli sbilanciamenti dei nodi, un nodo ha **surplus** se trattiene del flusso che deve inviare e ha **deficit** se non riceve del flusso che dovrebbe ricevere. Indichiamo con $e_i(x) = \sum x_{ji} - \sum x_{ij} - b_i$ lo **sbilanciamento** del nodo i a partire dallo pseudoflusso x e se $e_i(x) = 0$ il nodo si dice bilanciato, se $e_i(x) \geq 0$ il nodo ha surplus, $e_i(x) < 0$ vuol dire che il nodo ha deficit.
- Uno **pseudoflusso** é **minimale** se non ammette cicli aumentanti di costo negativo ed é lo pseudoflusso di costo minimo tra gli pseudoflussi con gli stessi sbilanciamenti. É inoltre uguale al flusso ammissibile di costo minimo.
- Pseudoflusso minimale ammissibile ≡ flusso ammissibile di costo minimo.

Sbilanciamento complessivo $g(x) = 0$

$$x_{ij} = \begin{cases} 0 & \text{se } c_{ij} \geq 0 \\ u_{ij} & \text{se } c_{ij} < 0 \end{cases}$$

Programmazione lineare

- Trasformare un problema (P) in (D) e viceversa.
  - **variabili**: $x \in \mathbb{R}^n$ (Vettori)
  - **funzione obbiettivo**: $\max c \cdot x$, $\min y \cdot b$
  - **vincoli**: $A_i x \leq b_i$, dove A_i é la i-esima riga della matrice A, $x \in \mathbb{R}^n$ e $b_i \in \mathbb{R}$ é l'i-esimo termine del vettore $b \in \mathbb{R}^m$.
- Indichiamo con $I(\bar{x}) = \{i : A_i \bar{x} = b_i\}$ l'insieme dei **vincoli attivi** per $\bar{x}$.

- $B \subseteq \{1, \dots, m\}$ si dice **Base** se $|B| = n$ e A_B è invertibile.
- $\bar{x} = A_B^{-1} \cdot b_B$ si dice **soluzione di base**.
- $(P)\ \max\{c \cdot x : Ax \leq b, x \geq 0\}$ e $\min\{y \cdot b : yA \geq c, y \geq 0\}$ si dice **coppia simmetrica** della P.L.
- $(P)\ \max\{c \cdot x : Ax \leq b\}$ e $\min\{y \cdot b : yA = c, y \geq 0\}$ si dice **coppia asimmetrica** della P.L.
- Per il Teorema della **dualità debole**, prese $\bar{x} \in \mathbb{R}^n$ soluzione ammissibile per (P) e $\bar{y} \in \mathbb{R}^m$ soluzione ammissibile per (D) allora $c \cdot x \leq y \cdot b$.
- Proprietà della dualità debole:
  - Se (P) è **superiormente illimitato**, (D) è vuoto.
  - Se (D) è **inferiormente illimitato**, (P) è vuoto.
  - $\max\{c \cdot x : Ax \leq b\} \leq \min\{y \cdot b : yA = c, y \geq 0\}$.
  - Se $c \cdot x = y \cdot b \Rightarrow \bar{x}$ ottima per (P) e $\bar{y}$ ottima per (D).
- $\bar{x}$ è ottima per (P) $\Leftrightarrow \exists\, \bar{y}$ ammissibile per (D) e viceversa.
- Indichiamo con $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ il **poliedro** (convesso).
- $P_I = \{x \in P : Ax = b_i, i \in I\}$ è la **faccia di P**.
- Un **vertice** è una faccia costituita da un solo punto.
- $\text{conv}(\{x_1, \dots, x_s\}) = \left\{\sum_{i=1}^{s} \lambda_i x_i : \lambda_i \geq 0, \sum_{i=1}^{s} \lambda_i = 1\right\}$ è l'**involucro convesso**, ovvero l'unione di tutti i vertici del poliedro. $\{x_1, \dots, x_s\}$ sono **vertici**.
- $\text{cono}(\{v_1, \dots, v_t\}) = \left\{\sum_{i=1}^{t} \gamma_i v_i : \gamma_i \geq 0\right\}$ è l'**involucro conico**, ovvero il cono formato dalle due rette delle direzioni ammissibili, o ancora meglio tutta la **regione ammissibile** tolto l'involucro convesso. $\{v_1, \dots, v_t\}$ sono **raggi**.
- Il nostro poliedro sarà l'unione di questi involucri, quindi $P = \text{conv}(\{x_1, \dots, x_s\}) + \text{cono}(\{v_1, \dots, v_t\})$.
- Preso un poliedro $P \subseteq \mathbb{R}^n$, $\max\{c \cdot x : x \in P\}$ ha **ottimo finito** $\Leftrightarrow$ $c \cdot v_i \leq 0$ con $i = 1, \dots, t$, in tal caso $\exists\, k \in \{1, \dots, s\}$ t.c. x^k è soluzione ottima.
- $\bar{\xi}$ è **direzione di crescita** se $c \cdot \bar{\xi} > 0$ (angolo $< 90°$) e "aumenta" il valore di c.

- Supponendo che (P) e (D) abbiano regioni ammissibili non vuote, allora $\max\{c\cdot x : Ax \le b\} = \min\{y\cdot b : yA = c, y \ge 0\}$.
  Proprietà della **dualità forte**:
  - Se (P) è **superiormente illimitato**, (D) è vuoto.
  - Se (D) è **inferiormente illimitato**, (P) è vuoto.
  - (P) ha **ottimo finito** $\iff$ (D) ha ottimo finito.
- $\xi \in \mathbb{R}^n$ si dice **direzione ammissibile** per $\bar{x} \in \mathbb{R}^n$ ammissibile per (P) se $\exists \bar{\lambda} > 0$ t.c. $\bar{x} + \lambda\xi$ è ammissibile per (P) $\forall \lambda \in [0, \bar{\lambda}]$.
  ξ è una direzione ammissibile per $\bar{x} \iff A_i \cdot \xi \le 0 \quad \forall i \in I(\bar{x})$.
  ξ è una direzione di crescita per $c\cdot x \iff c\cdot\xi > 0$
- $\bar{x} \in \mathbb{R}^n$ è una soluzione ottima $\iff$ non ammette direzioni ammissibili di crescita.
- $\begin{cases} A_{I(\bar{x})}\cdot\xi \le 0 \\ c\cdot\xi > 0 \end{cases}$ non ha soluzione $\xi \in \mathbb{R}^n$ $\iff$ $\begin{cases} y_{I(\bar{x})} A_{I(\bar{x})} = c \\ y_{I(\bar{x})} \ge 0 \end{cases}$ ammette una soluzione $y_{I(\bar{x})}$

Lemma di Farkas

- Prese $\bar{x}$ ammissibile per (P) e $\bar{y}$ ammissibile per (D), per il Teorema degli **scarti complementari** $\bar{x}$ e $\bar{y}$ sono ottime $\iff \bar{y}(b_i - A_i\bar{x}) = 0$.
- Se $\bar{y}_i > 0 \Rightarrow A_i\bar{x} = b_i$ oppure $A_i\bar{x} < b_i \Rightarrow \bar{y}_i > 0$.
- Indichiamo con $\bar{x} = A_B^{-1}\cdot b_B$ la **soluzione primale di base** e con $\bar{y} = (\bar{y}_B, \bar{y}_N)$ (dove $\bar{y}_B = cA_B^{-1}$, $\bar{y}_N = 0$) la **soluzione duale di base**.
- Una base B è **primale ammissibile** $\iff A_N\bar{x} \le b_N$ dove N è l'insieme degli indici non in base.
- Una base B per (P) è **degenere** $\iff \exists\, i \notin B : i \in I(\bar{x})$
- Una base B è **duale ammissibile** $\iff \bar{y}_B \ge 0$
- Una base B per (D) è **degenere** $\iff \exists\, i \in B : y_i = 0$
- $\bar{\lambda}_i$ è il **massimo spostamento** da $\bar{x}$ lungo $\bar{\xi}$ per cui il vincolo è soddisfatto.
  $\bar{\lambda}_i = +\infty$ se $A_i\cdot\bar{\xi} \le 0$, $i \in N$ oppure $\bar{\lambda}_i = \dfrac{(b_i - A_i\bar{x})}{A_i\cdot\bar{\xi}}$ se $A_i\bar{\xi} > 0$, $i \in N$.
  $\bar{\lambda} = \min\{\bar{\lambda}_i : i \in N\} \rightarrow$ **massimo spostamento ammissibile**.
  Un $k \in N$ t.c. $\bar{\lambda} = \bar{\lambda}_k$ si dice **indice entrante**
  Un $h \in B$ t.c. $\bar{y}_h < 0$ si dice **indice uscente**

$B(h)$ è la posizione di h nella base.

- $B = B \setminus \{h\} \cup \{K\}$ è la nuova base ottenuta alla fine di una iterazione dell'algoritmo del simplesso primale
- $\bar{\xi} = -A_B^{-1} \cdot u_{B(h)}$, ovvero si prende la $B(h)$-esima colonna della matrice e la si cambia di segno.
- Se durante un'iterazione dell'algoritmo del simplesso primale trovassimo più di uno h o K potremmo usare la regola di anticiclo di Bland prendendo il minore (vale per entrambi):
  - $h = \min\{i \in B : \bar{y}_i < 0\}$
  - $K = \min\{i \in N : \bar{\lambda} = \bar{\lambda}_i\}$
- $d = (-\gamma_B, 1, 0)$ con $\gamma_B = A_K A_B^{-1}$ si dice direzione di decrescita per l'algoritmo del simplesso duale.
- Qui $K = \min\{i \in N : A_i \bar{x} > b_i\}$ indice entrante.
- Il massimo passo di spostamento è $\bar{y} + \bar{\theta} d$ ed è maggiore o uguale a $0 \iff \theta \leq \bar{\theta} = \min\{\theta_i : i \in B\}$.

$$\bar{\theta}_i = \begin{cases} \bar{y}_i / \gamma_i & \text{se } \gamma_i > 0 \\ +\infty & \text{se } \gamma_i \leq 0 \end{cases}$$

- $h = \min\{i \in B : \bar{\theta} = \bar{\theta}_i\}$ indice uscente.
- $B = B \setminus \{h\} \cup \{K\}$ è la nuova base.

Programmazione lineare intera

- Fare i rilassamenti continui, se ho che $x_1, \dots, x_n \in \mathbb{Z}_+$ allora posso trasformarli in $-x_1, \dots, -x_n \leq 0$ se ho un problema di max altrimenti $y \geq 0$ se ho un problema di min.
- Se tutti i vertici del poliedro sono a componenti intere, allora (P) ammette una soluzione ottima a componenti intere. Tale proprietà prende il nome di proprietà dell'interezza.
- $S = P \cap \mathbb{Z}^n$ e $T = Q \cap \mathbb{Z}^m$ sono regioni ammissibili.

- Se S é finito allora $\text{conv}\, S$ é un poliedro con la proprietá dell'interezza e quindi $\max c \cdot x : x \in S \equiv \max c \cdot x : x \in \text{conv}\, S$.
- É possibile escludere una soluzione ottima $\bar{y}$ se in esso vi sono componenti frazionarie. Per farlo usiamo il piano di Taglio di Gomory a partire da un $r \in B$ t.c. $\bar{y}_r \notin \mathbb{Z}$, se ce sono molteplici le prendiamo in ordine crescente di r.
  Per prima cosa bisogna calcolare la matrice $\tilde{A} = A_N A_B^{-1}$ numerando le colonne con gli indici della base (B) e le righe con gli indici non in base (N) ed infine usare la seguente formula $\sum_{j \in N} \{\tilde{A}_{jr}\} \bar{y}_j \geq \{\bar{y}_r\}$ dove $\{\cdot\}$ denota la parte frazionaria.
- Per Trasformare un problema (P) in un problema (D) bisogna scrivere $-\min -c \cdot x$ come funzione obbiettivo e aggiungere una variabile di scarto per ogni vincolo che abbiamo ed infine sostituire il "$\leq$" con "$=$" e porre tutte le variabili ≥ 0 per i rilassamenti continui.

Programmazione lineare intera (Metodi enumerativi)

- Sono problemi di P.L.I. con un numero finito di soluzioni, quindi é possibile fare un'analisi diretta o indiretta di tutte le soluzioni ed enumerarle.
- Costruire in maniera "dinamica" l'albero di enumerazione.
- Utilizzare il metodo Branch and Bound:
  - Partire da una soluzione ammissibile.
  - Trovare un rilassamento del problema.
  - Seguire le regole di ramificazione (costruire dinamicamente)
  - Seguire le regole di potatura:
    1) Posso potare se il valore ottimo del rilassamento del sottoproblema é peggiore / non-migliore del valore della soluzione ammissibile

attuale.

2) Posso potare se la soluzione ottimo trovata per il rilassamento del sottoproblema è ammissibile per P (Problema)

3) Posso potare se il sottoalbero non contiene soluzioni ammissibili per P.

- Se mi trovo in un problema di massimizzazione:
  - Soluzione ammissibile → valutazione inferiore del valore ottimo
  - Valore ottimo del rilassamento → valutazione superiore del valore ottimo del sottoproblema.
- Se mi trovo in un problema di minimizzazione:
  - Soluzione ammissibile → valutazione superiore del valore ottimo
  - Valore ottimo del rilassamento → valutazione inferiore del valore ottimo del sottoproblema.

Programmazione lineare intera (Problema dello zaino)

- Abbiamo n oggetti, $c_i > 0$ è il beneficio, $a_i > 0$ è il peso per ogni oggetto e abbiamo uno zaino con capacità $b > 0$ ($\sum_{i=1}^{n} a_i > b$)
- Preso un problema (P) della forma $\max c \cdot x : Ax \leq b, x \geq 0$ è facile ottenere (D), della forma $\min yb : yA \geq c$ e $y \geq 0$, dato che $y \geq \max\left\{\frac{c_i}{a_i} : i = 1, \dots, n\right\}$ ($\frac{c_i}{a_i} \geq 0 \ \forall i$ quindi $y \geq 0$ superfluo) possiamo riscrivere (D) in questo modo $\min yb : y \geq \frac{c_i}{a_i}$.
  - soluzione ottima per (P): $\bar{x}_i = \begin{cases} b/a_i & \text{se } i = k \\ 0 & \text{se } i \neq k \end{cases}$
  - soluzione ottima per (D): $\bar{y} = \max\left\{\frac{c_i}{a_i} : i = 1, \dots, n\right\} = \frac{c_k}{a_k}$
- Il "beneficio unitario" o rendimento è quindi $\frac{c_i}{a_i}$.
- Una volta trovati i rendimenti bisogna ordinare le variabili per rendimento in maniera decrescente ($\frac{c_1}{a_1} \geq \frac{c_2}{a_2}, \dots, \geq \frac{c_n}{a_n}$)

- Un altro metodo per trovare le soluzioni ottime è quello di ordinare per rendimento le variabili (come prima) ma quando andremo a sceglierle, se l'aggiunta di quell'oggetto supera la capacità dello zaino, invece di non inserirlo, andremo ad aggiungerne una sua frazione (il massimo possibile da poter aggiungere). Quindi prendo $h = \{1, \dots, n-1\}$ in modo tale che inserisco i primi $n-1$ oggetti e quando supero la capacità inserisco la frazione:

- soluzione ottima per (P): $\bar{x}_j = \begin{cases} 1 & \text{se } j \le h \\ (b - \sum_{j=1}^{h} a_i)/a_{h+1} & \text{se } j = h+1 \\ 0 & \text{se } j > h+1 \end{cases}$

- Una volta trovati i rendimenti e ordinato le variabili nel modo corretto trova la soluzione ammissibile guardando i coefficienti delle variabili nel vincolo ed inserendole confrontando con la capacità b. Questo ci dà una valutazione inferiore se siamo in un problema di max e superiore se siamo in un problema di min, il valore va calcolato moltiplicando gli 1 e gli 0 della soluzione per i coefficienti della funzione obiettivo e sommandoli tra loro. Poi bisogna calcolare il rilassamento con il secondo metodo e otteniamo una valutazione superiore se siamo in un problema di max e una valutazione inferiore se siamo in un problema di min.
- A questo punto bisogna creare l'albero di enumerazione seguendo le rispettive regole.

Programmazione lineare intera (Problema del commesso viaggiatore)

- A partire da un grafo completo non orientato indichiamo con c_{ij} la "lunghezza" dell'arco $(i,j) \in A$.

- L'obbiettivo è trovare un **ciclo hamiltoniano** ovvero un ciclo che passa per ciascun nodo $i \in N$ una ed una solo volta.
- Se prendiamo una variabile logica $x_{ij} = 1$ se $(i,j) \in A$ appartiene al ciclo e $x_{ij} = 0$ altrimenti, allora $\min \sum_{(i,j) \in A} c_{ij} x_{ij}$ è il nostro obbiettivo.
- Il **grado** di ciascun nodo $i \in N$ è 2 $\rightarrow \sum_{(i,j) \in A} x_{ij} = 2$
- Un **taglio** deve prendere almeno due archi $\rightarrow \sum_{(i,j) \in A(N', N'')} x_{ij} \geq 2$
- Un **K-albero** è un albero di copertura per il grafo privato del nodo K e degli archi incidenti in K più due archi incidenti in K.

Un ciclo hamiltoniano è un K-albero in cui ogni nodo ha grado 2.

- Un **K-albero di costo minimo** è un albero di copertura per il grafo privato del nodo K e degli archi incidenti in K più due archi incidenti in K di costo minimo.
- Preso un insieme $V = \{i_1, \dots, i_K\}$ con i_j distinti:

$$i_{K+1} \text{ soddisfa } \begin{cases} i_{K+1} \notin V \\ c_{i_K i_{K+1}} = \min \{ c_{i_K j} : j \notin V \} \end{cases}$$

- Per trovare la soluzione ottima e quindi un ciclo hamiltoniano abbiamo bisogno di alcune regole da seguire.
  Per prima cosa seguire tutte le regole del branch bound per la costruzione dell'albero di enumerazione in base alla tecnica di rilassamento, soluzione iniziale e ramificazione.