Università Roma Tre – Dipartimento di Matematica e Fisica
IN440 – Ottimizzazione Combinatoria
Prof. Marco Liverani

Ottimizzazione combinatoria e programmazione matematica

Famiglie di problemi

Possiamo distinguere diverse famiglie di problemi sulla base del *tipo di soluzione* richiesta:

- **Problemi di decisione** o di esistenza: si richiede di *decidere* se una soluzione *esiste*; la risposta è binaria, di tipo «sì/no», «vero/falso», «0/1»
  - Esempi: Esiste uno zero della funzione $f(x)$ nell'intervallo (a, b)? Esiste un cammino tra i vertici u e v del grafo G?
  - Nota: non si richiede di esibire una soluzione, ma soltanto di decidere se almeno una soluzione esiste

- **Problemi di ricerca**: si richiede di esibire *una* soluzione del problema
  - Esempi: Calcolare uno zero della funzione $f(x)$ nell'intervallo (a, b). Trovare un cammino tra i vertici u e v del grafo G
  - Nota: risolvere un problema di ricerca equivale a risolvere anche il problema di decisione corrispondente

- **Problemi di enumerazione**: si richiede di calcolare *tutte* le soluzioni del problema
  - Esempi: Calcolare tutti gli zeri della funzione $f(x)$ nell'intervallo (a, b). Trovare tutti i cammini tra i vertici u e v del grafo G
  - Nota: risolvere un problema di enumerazione equivale a risolvere anche il problema di ricerca corrispondente

Problemi di ottimizzazione

- I problemi di ottimizzazione richiedono di calcolare, se esiste, una soluzione tra quelle che rendono *minimo* o *massimo* (a seconda del problema) un determinato criterio di scelta della soluzione
- Il criterio di scelta è espresso mediante una funzione $f(x)$, detta ***funzione obiettivo*** del problema: ***ottimizzare*** la funzione obiettivo significa trovare una soluzione x^* del problema, la ***soluzione ottima***, che renda minima o massima la funzione obiettivo

- Formalmente un'istanza di un problema di ottimizzazione è definita mediante una coppia (A, f), dove A è l'insieme delle ***soluzioni ammissibili*** per il problema in esame e f: $A \longrightarrow \mathbb{R}$ è la funzione obiettivo che si deve ottimizzare
- Si chiede di trovare i valori $x^* \in A$ tali che $f(x^*) \leq f(x)$ per ogni $x \in A$ (problema di minimizzazione)
- Tali valori di x^* sono le soluzioni ottime globali; se invece $f(x^*) \leq f(x)$ solo per i valori di x in un intorno di x^*, allora x^* è una soluzione ottima locale
- L'*insieme delle soluzioni ammissibili* è un sottoinsieme dello ***spazio delle soluzioni*** e in genere viene definito attraverso un insieme di ***vincoli*** che delimitano e circoscrivono l'insieme entro cui possono assumere i valori le variabili del problema

Problemi di programmazione matematica

- Nei problemi di **programmazione matematica** i vincoli che circoscrivono l'insieme A delle soluzioni ammissibili, sono espressi attraverso un insieme di equazioni e disequazioni (anche molto numerose)
- Problema di programmazione matematica:

  minimizzare $f(x)$ per x tale che

  $g_i(x) \geq 0$ per $i = 1, 2, ..., m$

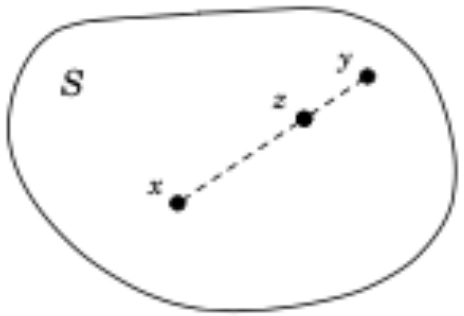
  $h_j(x) = 0$ per $j = 1, 2, ..., p$

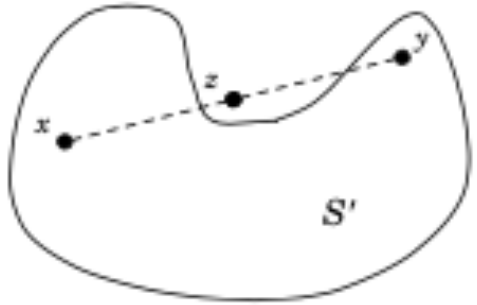
  dove le funzioni $g_i(x)$ e $h_j(x)$ costituiscono i vincoli del problema

Combinazione convessa

- Dati due punti $x, y \in \mathbb{R}^n$ una **combinazione convessa** di x e y è un punto $z = \lambda x + (1 - \lambda)y$, con $\lambda \in \mathbb{R}$ e $0 \leq \lambda \leq 1$
- Un insieme $S \subseteq \mathbb{R}^n$ è **convesso** se contiene qualsiasi combinazione convessa di elementi di S: $x, y \in S \Rightarrow \lambda x + (1 - \lambda)y \in S$

S convesso





S' **non** convesso

- **Prop.:** Se $S_1, \ldots, S_k$ sono insiemi convessi, allora anche $\cap_{i=1}^{k} S_i$ è convesso

Programmazione convessa

- Una funzione $f: S \longrightarrow \mathbb{R}$ è **convessa** in S convesso se, per ogni $x, y \in S$, risulta $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$

- **Prop.:** Sia $f(x)$ una funzione convessa nell'insieme convesso S; allora per ogni $t \in \mathbb{R}$ l'insieme $S_t = \{x \in S: f(x) \leq t\}$ è convesso

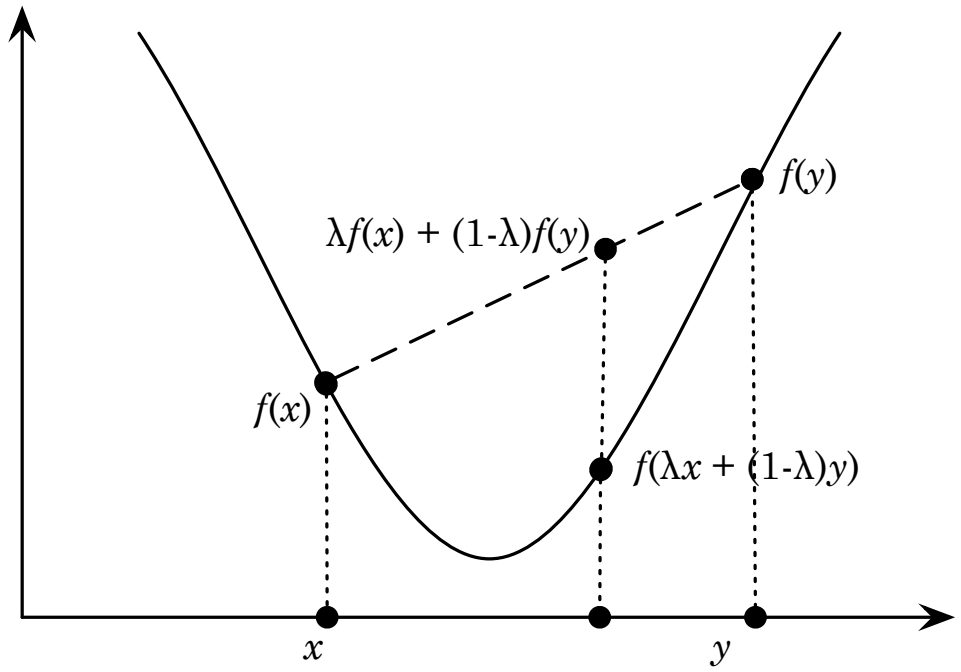
- Un problema di ottimizzazione formulato come un problema di programmazione matematica

  minimizzare $f(x)$ per x tale che

  $g_i(x) \geq 0$ per $i = 1, 2, ..., m$

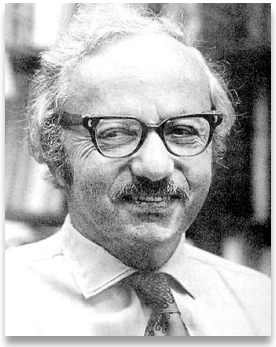
  $h_j(x) = 0$ per $j = 1, 2, ..., p$

  è un problema di **programmazione convessa** se
  - la funzione obiettivo $f(x)$ è convessa
  - le funzioni $g_i(x)$ sono concave ($-g_i(x)$ sono convesse)
  - le funzioni $h_j(x)$ sono lineari



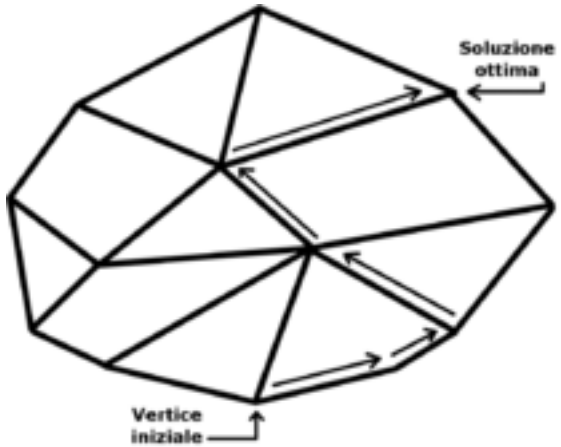
- **Prop.:** In un problema di programmazione convessa ogni soluzione ottima locale è anche una soluzione ottima globale

Programmazione lineare



George Dantzig
(1914 – 2005)

- Un problema di programmazione matematica in cui la funzione obiettivo e i vincoli del problema sono espressi con funzioni lineari, è un problema di **programmazione lineare** (LP)
- Se la funzione obiettivo è lineare, allora i suoi punti di minimo e di massimo si trovano sicuramente sulla frontiera dell'insieme A delle soluzioni ammissibili
- **George Danzig** nel 1947 propose un metodo generale per la risoluzione di problemi di programmazione lineare, noto come **metodo del simplesso**, basandosi su tale osservazione
  - L'algoritmo di Dantzig opera su un ***simplesso***, ossia un **politopo di n dimensioni** con $n + 1$ vertici (nel piano, lo spazio di dimensione $n = 2$, è un triangolo, nello spazio di dimensione $n = 3$ un simplesso è un tetraedro, ecc.)
  - In un problema di programmazione lineare l'insieme A delle soluzioni ammissibili è un simplesso
  - Il metodo del simplesso individua la soluzione ottima del problema di programmazione lineare, percorrendo gli spigoli del politopo di n dimensioni (il simplesso): la soluzione ottima si troverà su uno dei vertici



Programmazione Lineare Intera e Ottimizzazione Combinatoria

- Nei problemi di **programmazione lineare intera** (PLI) *alcune* delle variabili del problema sono vincolate ad assumere valori interi
  - in questo modo si riducono drasticamente i punti dello spazio delle soluzioni ammissibili
  - se nel problema PLI si elimina il «vincolo di integrità» (le variabili possono essere tutte a valori reali, non solo interi) si ottiene un problema di PL detto **rilassamento continuo** del problema PLI corrispondete
  - se il rilassamento continuo ha le stesse soluzioni del problema PLI corrispondente, allora si dice che gode della **proprietà di integralità**

- Nei problemi di **programmazione lineare intera 0-1** alcune variabili possono assumere solo i valori nell'insieme $\{0, 1\}$

- Se *tutte* le variabili del problema di programmazione lineare sono vincolate ad assumere solo valori interi, il problema è di **ottimizzazione combinatoria**

Programmazione Lineare: alcuni esempi

- **Problema della bisaccia** (*knapsack problem*): abbiamo un sacco (una «bisaccia») che può reggere fino ad un peso P; dobbiamo usare il sacco per trasportare un certo numero di oggetti di peso diverso selezionandoli da una collezione di n elementi; ogni oggetto ha un costo/valore c_i e un peso p_i
- *Problema di ottimizzazione*: voglio riempire il sacco senza superare la sua capacità P e massimizzando il valore degli oggetti selezionati
- Alle variabili $x_i \in \{0, 1\}$ $(i = 1, ..., n)$ assegniamo il valore $x_i = 1$ se l'oggetto i viene scelto e $x_i = 0$ altrimenti
- Una soluzione del problema è il vettore $\boldsymbol{x} = (x_1, ..., x_n) \in \{0, 1\}^n$ con cui può essere identificato il contenuto della bisaccia
- Possiamo formalizzare il problema di ottimizzazione come un *problema di programmazione lineare*:

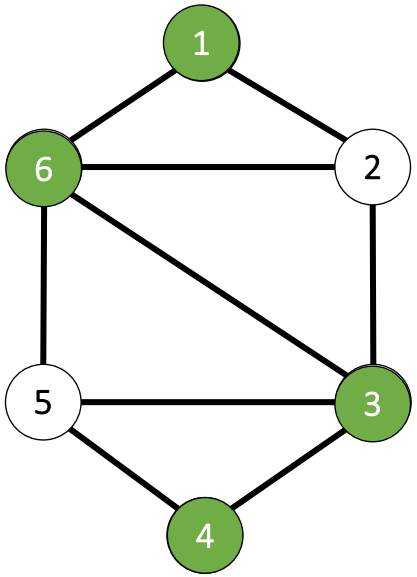
$$\text{massimizzare } f(x) = \sum_{i=1}^{n} c_i x_i$$

$$\sum_{i=1}^{n} p_i x_i \leq P$$

$$x_i \in \{0,1\} \quad i = 1,2,\ldots,n$$

Programmazione Lineare: alcuni esempi

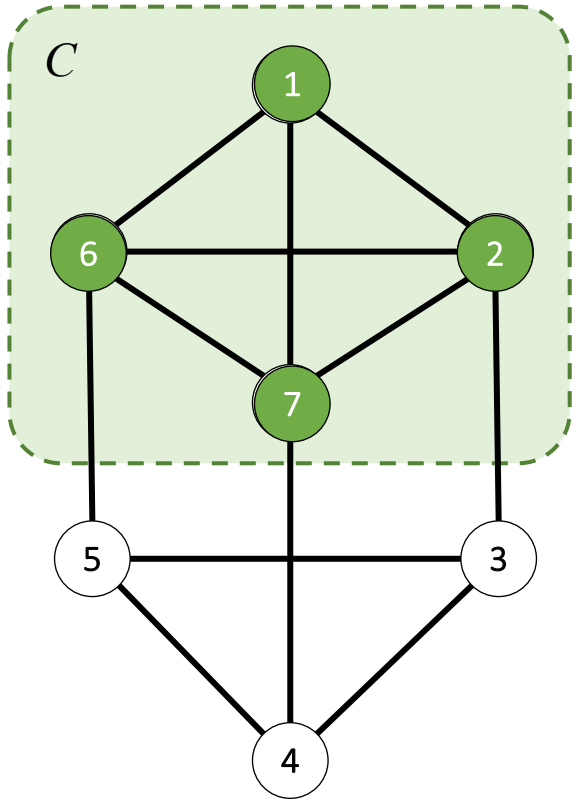
- Dato un grafo $G = (V, E)$ un **ricoprimento di vertici** (*vertex cover*) è un sottoinsieme $C \subseteq V$ tale che ogni spigolo di G abbia almeno un estremo in C
- *Problema di ottimizzazione combinatoria*: dato G calcolare il ricoprimento di vertici C di cardinalità minima



- Impostiamolo come problema di programmazione lineare intera:
  - Le variabili del problema sono $x_1, \ldots, x_n$ e sono definite in modo tale che per ogni $v_i \in V(G)$ sia $x_i = 1$ se $v_i \in C$ e $x_i = 0$ altrimenti (se $v_i \notin C$)
  - La funzione obiettivo da *minimizzare* è $f(x_1, \ldots, x_n) = \sum_{v_i \in V} x_i$
  - C è un ricoprimento (una soluzione ammissibile) se per ogni spigolo $(v_i, v_j) \in E(G)$ almeno uno dei due vertici v_i e v_j è in C, ossia se $x_i + x_j \geq 1$
- *Problema di programmazione lineare intera*:

$$\begin{cases} \min \sum_{v_i \in V} x_i \\ \forall\, (v_i, v_j) \in E,\ x_i + x_j \geq 1 \\ \forall\, v_i \in V\ x_i \in \{0, 1\} \end{cases}$$

Programmazione Lineare: alcuni esempi



- Dato un grafo $G = (V, E)$ una **clique** è un sottoinsieme $C \subseteq V$ tale che ogni coppia di vertici in C sia collegata da uno spigolo in G (C induce un sottografo completo su G)
- *Problema di ottimizzazione combinatoria*: dato G individuare una **clique di dimensione massima** su G

- Impostiamolo come problema di programmazione lineare intera:
  - Le variabili del problema sono $x_1, \ldots, x_n$ e sono definite in modo tale che per ogni $v_i \in V(G)$ sia $x_i = 1$ se v_i è nella clique C e $x_i = 0$ altrimenti (se $v_i \notin C$)
  - La funzione obiettivo da massimizzare è $f(x_1, \ldots, x_n) = \sum_{v_i \in V} x_i$
  - C è una clique se per ogni coppia $(v_i, v_j) \notin E(C)$ risulta $v_i \notin C$ o $v_j \notin C$; formalmente per ogni $(v_i, v_j) \notin E(C)$ risulta $x_i + x_j \leq 1$
  - *Problema di programmazione lineare intera:*

$$\begin{cases} \max \sum_{v_i \in V} x_i \\ \forall\, (v_i, v_j) \notin E,\ x_i + x_j \leq 1 \\ \forall\, v_i \in V\ x_i \in \{0, 1\} \end{cases}$$

Programmazione Lineare: alcuni esempi

- Sia $G = (V, E)$ un grafo con dei «costi» non negativi assegnati agli spigoli: per ogni $(v_i, v_j) \in E(G)$ sia $w_{ij} \geq 0$ il costo corrispondente
- *Problema di ottimizzazione combinatoria*: dati due vertici $s, t \in V(G)$ si vuole trovare il **cammino p di costo minimo** da s a t
- *Problema di programmazione lineare intera*: per ogni spigolo $(v_i, v_j) \in E(G)$ sia $x_{ij} = 1$ se (v_i, v_j) è sul cammino p: $s \rightsquigarrow t$, altrimenti $x_{ij} = 0$
- La *funzione obiettivo* da minimizzare è la seguente: $f(\mathbf{x}) = \sum_{(v_i,v_j)\in E} w_{ij}x_{ij}$
- Affinché p sia un cammino da s a t deve risultare: $\sum_{(v_i,v_h)\in E} x_{hj} - \sum_{(v_j,v_i)\in E} x_{ij} = \begin{cases} -1, & v_h = s \\ 1, & v_h = t \\ 0, & \forall v_h \in V \setminus \{s,t\} \end{cases}$
- Formalizzazione come problema di *programmazione lineare intera*:

$$\begin{cases} \min \sum_{(v_i,v_j)\in E} w_{ij}x_{ij} \\ \sum_{(v_i,v_h)\in E} x_{hj} - \sum_{(v_j,v_i)\in E} x_{ij} = -1, \text{ se } v_h = s \\ \sum_{(v_i,v_h)\in E} x_{hj} - \sum_{(v_j,v_i)\in E} x_{ij} = 1, \text{ se } v_h = t \\ \sum_{(v_i,v_h)\in E} x_{hj} - \sum_{(v_j,v_i)\in E} x_{ij} = 0, \text{ se } v_h \in V \setminus \{s,t\} \\ \forall\, (v_i, v_j) \in E,\ x_{ij} \in \{0,1\} \end{cases}$$

Programmazione Lineare: alcuni esempi

- Sia $G = (V, E)$ un grafo con dei «costi» non negativi assegnati agli spigoli: per ogni $(v_i, v_j) \in E(G)$ sia $w_{ij} \geq 0$ il costo corrispondente
- Un albero ricoprente (*spanning tree*) di G è un sottografo $T = (V, E')$ tale che $E' \subseteq E(G)$ in modo che T sia connesso e aciclico (un albero)
- *Problema di ottimizzazione*: dato G trovare l'**albero ricoprente di costo minimo**; in altri termini si vuole trovare l'albero ricoprente T^* del grafo G tale che

$$\sum_{(v_i,v_j)\in E(T^*)} w_{ij} = \min_{T \text{ che ricopre } G} \sum_{(u,v)\in E(T)} w_{uv}$$

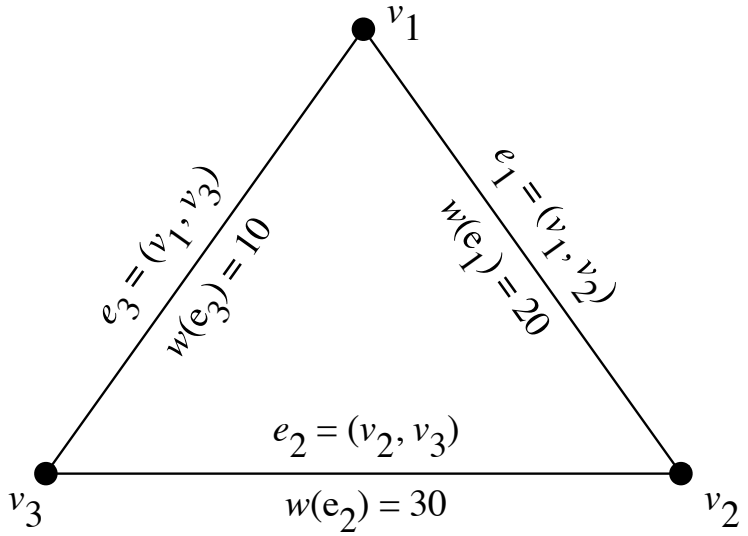
- Variabili per la formalizzazione del problema in termini di *programmazione matematica*: per ogni spigolo $e_i \in E(G)$, $i = 1, \ldots, m$, il cui costo è w_i, definiamo la variabile $x_i = 1$ se $e_i \in E(T)$, altrimenti $x_i = 0$
- Possiamo così definire la *funzione obiettivo da minimizzare*: $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$

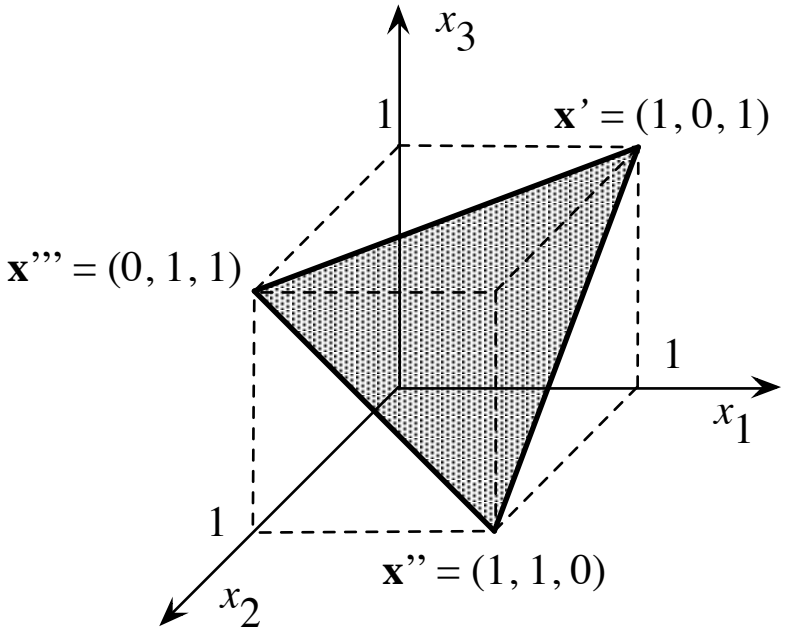
- Problema di *programmazione lineare intera*:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \textstyle\sum_{i=1}^{m} w(e_i) x_i \\ \text{per } \mathbf{x} \text{ tale che} \quad & x_1 + x_2 + \cdots + x_m = n - 1 \\ & x_i \geq 0 \ \text{ per } i = 1, 2, \ldots, m \\ & x_i \leq 1 \ \text{ per } i = 1, 2, \ldots, m \end{aligned}$$

Programmazione Lineare: alcuni esempi

- Consideriamo un'istanza estremamente semplice del problema dell'albero ricoprente di costo minimo (*minimum spanning tree*): sia $G = (V, E)$ con
  - $V = \{v_1, v_2, v_3\}$
  - $E = \{e_1 = (v_1, v_2), e_2 = (v_2, v_3), e_3 = (v_3, v_1)\}$
  - con $w_1 = w(e_1) = 20$, $w_2 = w(e_2) = 30$, $w_3 = w(e_3) = 10$
- Lo spazio delle soluzioni è costituito da $A = \{ \boldsymbol{x}' = (1, 0, 1), \boldsymbol{x}'' = (1, 1, 0), \boldsymbol{x}''' = (0, 1, 1)\}$
- Effettuando il *rilassamento continuo* del problema di programmazione intera possiamo estendere A al politopo di dimensione 3
- È così evidente che le soluzioni del problema si trovano nei vertici del simplesso
- In particolare la soluzione ottima si ha per $\boldsymbol{x}'$: $f(\boldsymbol{x}') = 20 + 10 = 30$





Riferimenti bibliografici

- Cormen, Leiserson, Rivest, Stein, «*Introduzione agli algoritmi e strutture dati*», terza edizione, McGraw-Hill (Cap. 29)
- Christos Papadimitriou, Kenneth Steiglitz, «*Combinatorial Optimization – Algorithms and Complexity*», Dover Publications Inc., 1998
- Antonio Sassano, «*Modelli e algoritmi della Ricerca Operativa*», Franco Angeli, 1999
- Paolo Serafini, «*Ottimizzazione*», Zanichelli, 2000
- Marco Liverani, *Dispense del Corso di Ottimizzazione Combinatoria: Problemi di Ottimizzazione e Programmazione Matematica* (http://www.mat.uniroma3.it/users/liverani/doc/disp_oc_05.pdf)