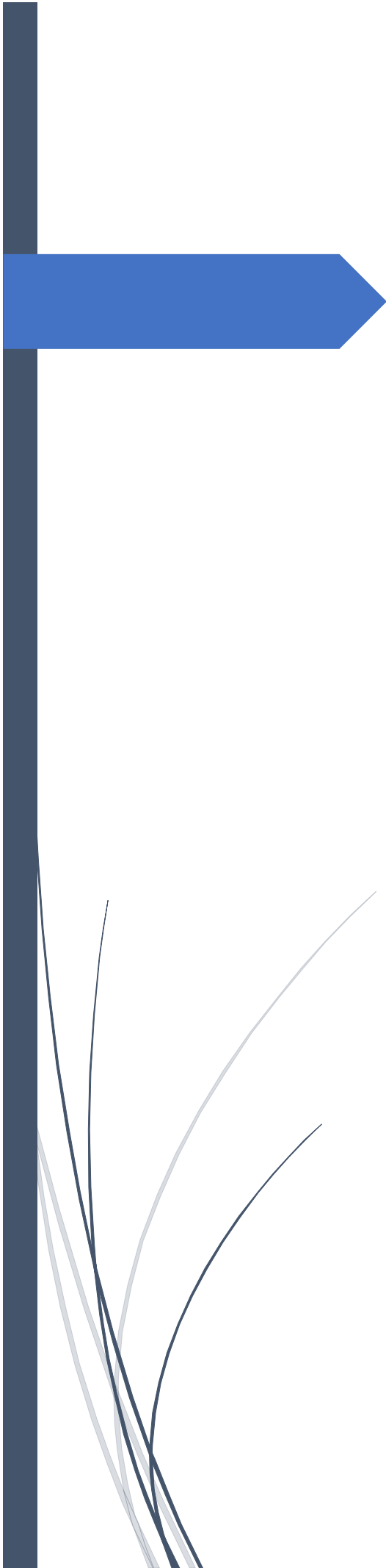




Ricerca operativa semplice (per davvero)

Rovesti Gabriel

Attenzione



Il file non ha alcuna pretesa di correttezza; di fatto, è una riscrittura attenta di appunti, slide, materiale sparso in rete, approfondimenti personali dettagliati al meglio delle mie capacità. Credo comunque che, per scopo didattico e di piacere di imparare (sì, io studio per quello e non solo per l'esame) questo file possa essere utile. Semplice si pone, per davvero ci prova.

Thank me sometimes, it won't kill you that much.

Gabriel

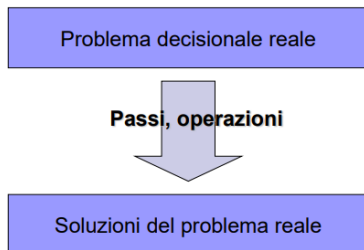
Sommario

Introduzione al corso.....	3
Modelli di programmazione lineare	6
Esercizi di Programmazione Lineare: Modello dieta e modello di costo minimo e vari esempi.....	10
Esercizi di Programmazione Lineare: Modello di trasporto e vari esempi.....	14
Conclusione Programmazione Lineare –	19
Esercizi, F.o. in forma min-max, max-min e min-abs, Modelli con vincoli di tipo logico, Vincoli di Big-M	19
Laboratorio 1: Utilizzo di solver per programmazione matematica.....	25
Programmazione Lineare e Metodo del Simplex: Vertici (e vertice ottimo), poliedri, forma standard	29
Programmazione Lineare e Metodo del Simplex	34
Laboratorio 2 – AMPL per programmazione matematica (da slide “Introduzione ai risolutori Excel e AMPL”)	40
Metodo del simplex: cambio base, costi ridotti, variabili entranti ed uscenti, condizione di illimitatezza, passi metodo del simplex.....	44
Metodo del simplex: casi notevoli simplex e regola anticiclo di Bland	54
Laboratorio 3: Risoluzione di vari modelli con AMPL.....	65
Laboratorio 4: Ulteriori modelli di PL ed esempi di esercizi AMPL da esame	72
Esercizi vari Metodo del Simplex	76
Dualità e Programmazione Lineare: Condizioni di ottimalità/complementarietà ed esercizi	89
Laboratorio 5 – Scenari modello distribuzione PC, esempi scripting, esempio modello primale/duale	96
Problemi di ottimizzazione su reti di flusso e algoritmi per il problema dei cammini minimi	101
Il problema del cammino minimo, Algoritmo Label Correcting per problema cammino minimo e Generico: Correttezza e Convergenza, Albero e grafo dei cammini minimi.....	104
Algoritmo di Bellman-Ford: Convergenza, Correttezza, Complessità. Esempi applicativi e Cammini Minimi con massimo numero di archi/hop	111
Tutorato (1) del prof: Esercizi di programmazione lineare	117
Tutorato (2) del prof: Esercizi di programmazione lineare, Simplex, CCPD.....	119
Tutorato (3): Svolgimento esame 23-01-2021	124
Laboratorio 6: Flussi con Reti	135
Branch-and-Bound per problemi di Programmazione Lineare Intera.....	141
Branch-and-Bound per problemi di Programmazione Lineare Intera.....	146
Laboratorio 7 – Branch and Bound con AMPL	154
Esercizi sul Branch and Bound	164
Tutorato del prof: Svolgimento alcuni esercizi Appello 10/02/2020 Tema A	166
Tutorato del prof: Soluzioni ad alcuni esercizi di “Varie”	168

Introduzione al corso

Ricerca Operativa (Operations Research) → Supporto ai processi decisionali in sistemi complessi, operando la decisione migliore a seconda del tipo di problema decisionale.

Cerco i passi che mi permettono di passare dalla decisione del problema alla soluzione stessa, applicando dei passi (quindi, un algoritmo), cercando un'ottimizzazione (soluzione migliore possibile)



Ricerca delle operazioni con metodo scientifico

Da Wikipedia, definizione espansiva e quella presente alla data di scrittura di questo file:

“La ricerca operativa (nota anche come teoria delle decisioni, scienza della gestione o, in inglese, operations research ("Operational Research" in Europa) e indicata con le sigle RO o OR) è la branca della matematica applicata in cui problemi decisionali complessi vengono analizzati e risolti mediante modelli matematici e metodi quantitativi avanzati (ottimizzazione, simulazione, ecc.) come supporto alle decisioni stesse. La ricerca operativa riveste un ruolo importante nelle attività decisionali perché permette di operare le scelte migliori per raggiungere un determinato obiettivo rispettando vincoli che sono imposti dall'esterno e non sono sotto il controllo di chi deve compiere le decisioni.

L'obiettivo è dunque quello di fornire un supporto alla presa di decisioni. Per giungere a questo scopo, la ricerca operativa fornisce strumenti matematici e algoritmici di supporto alle attività decisionali in cui occorre gestire e coordinare attività e risorse limitate al fine di massimizzare o minimizzare una funzione obiettivo.

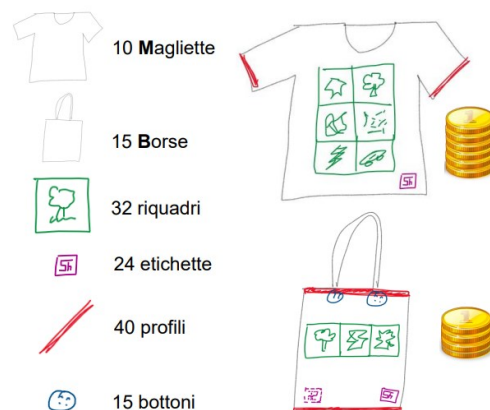
La ricerca operativa si occupa dunque di formalizzare un problema in un modello matematico e calcolare una soluzione ottima, quando possibile, o approssimata (detta anche subottima) per esso. Essa costituisce un approccio scientifico alla risoluzione di problemi complessi, si può ricondurre all'ambito della matematica applicata, ma presenta forti caratteristiche interdisciplinari relative in prevalenza a matematica, informatica, economia e finanza, ingegneria ed altre. Inoltre, la ricerca operativa ha molte applicazioni commerciali soprattutto negli ambiti economico, infrastrutturale, logistico, militare, della progettazione di servizi e di sistemi di trasporto e nelle tecnologie. Nel caso particolare di problemi di carattere economico, la funzione da massimizzare può coincidere con il massimo profitto ottenibile o con il minor costo da sostenere.”

Esempio pratico di ottimizzazione: creazione di magliette e borse con un certo numero di riquadri. L'obiettivo sarebbe di creare un programma per massimizzare il profitto date queste risorse.

Serve un *criterio* per capire come ragionare sulle risorse. Per esempio, posso ragionare ordinando in base al profitto, sommando le risorse che si devono usare e dividendo per quantità.

Altri ragionamenti possibili:

- scelte greedy (migliori sulla base di un input) sulla base delle magliette e delle borse
- bruteforce/approccio esaustivo-esponenziale (provo tutte le combinazioni possibili)



Obiettivo:



Decisioni:

- Quante M?
- Quante B?

- scelgo una quantità casuale e cerco di vedere quale tra tutte si avvicina di più al valore che mi serve

Problemi di ottimizzazione

Determinare la migliore configurazione di sistemi complessi sotto condizioni di utilizzo di risorse scarse

- Mix ottimo di produzione
- Pianificazione della produzione, schedulazione di processi
- Determinazione dei turni del personale
- Determinazione di percorsi ottimali
- Organizzazione dei flussi di dati in una rete di telecomunicazione
- Individuazione di sequenze genomiche
- Pianificazione e gestione operativa di reti di trasporto (Amazon)

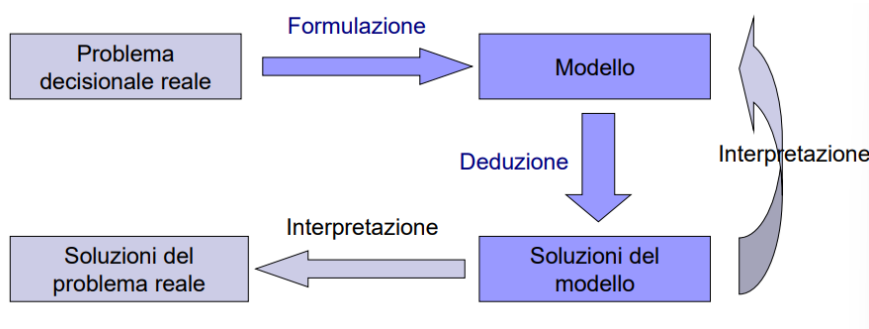
Si cerca in effetti di:

- Generare delle soluzioni ammissibili, quindi valide nel contesto d'uso
- Proporre delle soluzioni ragionevoli, quindi contestualmente utili alla situazione analizzata

Tuttavia:

- Occorre certificare che la soluzione proposta sia ottima (migliore in assoluto)
- Occorre capire il valore intrinseco delle risorse (quindi, determinare quelle più o meno utili)
- Occorre valutare se la soluzione proposta sia stabile (quindi, rimanga tale anche quando in dati variano)
- Occorre determinare soluzioni ottime in problemi simili (in modo modellistico, quindi individuando pattern tipici e ripetuto, quindi algoritmico)

Il metodo della Ricerca Operativa è basato su:



■ Formulazione: modelli matematici, su grafo, di simulazione, teoria dei giochi, data-driven (intelligenza artificiale)

■ Deduzione: metodi quantitativi, algoritmi efficienti.

Nell'uso di un modello, non si trova una soluzione del problema; questo ovviamente ha dei limiti, in quanto la soluzione deve essere interpretata sulla base della realtà (se è un buon modello, si adatta bene).

Vediamo un esempio reale per capire l'utilizzo della matematica:

“Un coltivatore ha a disposizione 11 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg di semi di lattuga, 18 t di tuberi, 145 mc di fertilizzante. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata per la coltivazione di lattuga è di 3000 €/ettaro e quella delle patate è di 5000 €/ettaro. Il consumo di risorse per ogni tipo di coltivazione è di 7 kg di semi e 10 mc di fertilizzante per ettaro di lattuga, e 3 t di tuberi e 20 mc di fertilizzante per le patate. Stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.”

Dati utili:

Scritto da Gabriel

- Lattuga e Patate (quali risorse)
- Semi, Tuberi e Fertilizzante (quali sottotipi di risorse e incrocio possibile dei dati)

Di fatto, in pratica, è come il problema di prima.

Non si può adottare un approccio forza bruta (quantità continue e non discrete). In realtà, questo ci aiuta.

Costruzione del modello

- Cosa bisogna decidere? $\Rightarrow$ variabili decisionali (incognite che rappresentano i dati)
- Qual è l'obiettivo? $\Rightarrow$ funzione obiettivo (ciò che vogliamo ottenere, mette insieme le v. decisionali)
- Come sono caratterizzate le soluzioni ammissibili? $\Rightarrow$ vincoli del problema (relazioni tra incognite)
- Modelli matematici $\Rightarrow$ f. obiettivo e vincoli visti come relazioni matematiche tra le variabili decisionali

■ Variabili decisionali:

x_L : quantità in ettari da destinare a lattuga
 x_P : quantità in ettari da destinare a patate

■ Funzione obiettivo:

$\max 3000 x_L + 5000 x_P$ (ricavo totale da massimizzare)

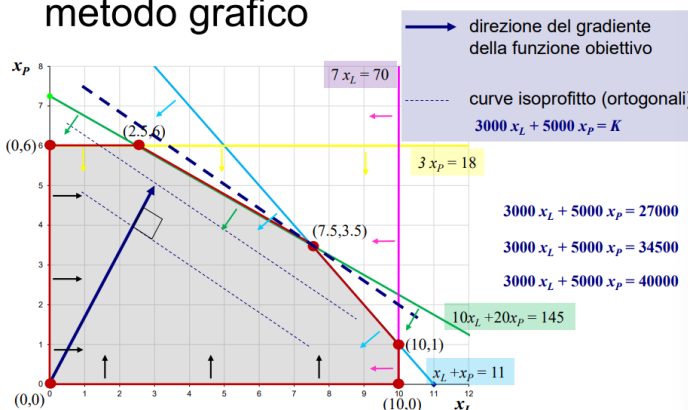
■ Sistema dei vincoli:

$x_L + x_P \leq 11$ (ettari disponibili)
 $7 x_L \leq 70$ (semi disponibili)
 $3 x_P \leq 18$ (tuberi disponibili)
 $10 x_L + 20 x_P \leq 145$ (fertilizzante disponibile)
 $x_L \geq 0, x_P \geq 0$ (dominio)

Aggiungiamo che quelle citate sotto come curve isoprofitto sono la differenza tra il ricavo ed i costi totali.

La soluzione empirica del problema è presente nel file *risolutore.xls* del primo laboratorio

Soluzione: metodo grafico



Per trovare la soluzione ottima, usiamo la funzione obiettivo.

Man mano che aumenta il guadagno, ho sempre più rette che vanno verso il *gradiente* (piano delle soluzioni ammissibili in rosso, possiamo dire il vettore che descrive il "movimento" delle derivate parziali delle rette); finché ci sono rette parallele, aumento il profitto, fino ad arrivare al limite della regione.

La soluzione ottima qui è 40000, in cui la soluzione ottima è il punto (7.5, 3.5).

Il metodo grafico funziona grazie a due motivi:

$\Rightarrow$ la *linearità* della funzione obiettivo

$\Rightarrow$ la *linearità* dei vincoli

Si parla in questi casi di modelli di programmazione lineare (PL) (almeno due variabili e funzioni isoprofitto sotto forma di rette).

Sotto queste ipotesi (come vedremo meglio in seguito), una soluzione si trova su un vertice della regione ammissibile: l'ultimo toccato traslando le rette isoprofitto nella direzione del gradiente.

Se ci sono più variabili, si cerca di passare ad un problema a livello geometrico, tale che si possa approssimare comunque algebricamente il problema.

In questo corso usiamo considereremo solo modelli di programmazione lineare PL, tale da avere un buon modello matematico su cui basarsi. Per risolvere questi problemi:

- Risolutore per fogli di calcolo (Excel, Calc etc.)
- Software di ottimizzazione
- ◆ Linguaggi di modellazione matematica (OPL, AMPL, Mosel, Lingo, GAMS etc.)
- ◆ Motori di ottimizzazione (Cplex, Gurobi, Xpress, Scip, CoinOR, GLPK, LPSolve etc.)

Sottolineato è AMPL, software oggetto dei nostri laboratori, mentre in corsivo i relativi risolutori utilizzati.

Programma di massima del corso

(da 6 CFU, prima era da 7; ciò che cambia sono il dimensionamento esplicito delle variabili big-M, i vincoli spuri e gli esercizi del modello dello zaino 0)

1. Problemi di ottimizzazione: modellazione e soluzione con software off-the-shelf (app già pronte da usare)
 - formulazione di modelli di programmazione matematica;
 - soluzione con l'utilizzo di pacchetti software (laboratorio).
2. Programmazione lineare:
 - teoria e metodo del simplesso;
 - teoria della dualità e applicazioni.
3. Ottimizzazione su grafi: modelli e algoritmi per
 - problema del cammino minimo;
 - problemi di flusso su reti (flusso massimo, flusso di costo minimo).
4. Introduzione alla Programm. Lineare Intera e all'Ottimizzazione Combinatoria
 - metodo del Branch & Bound per PL;
 - cenni sui metodi euristici e metaeuristici (ricerca locale e varianti)

Modelli di programmazione lineare

Diamo quindi esplicitamente il concetto di modello di programmazione lineare, quindi una relazione che esprime la soluzione ottima di un problema di ottimizzazione attraverso relazioni matematiche lineari (altri non lineari esistono, ma il metodo di soluzione sfrutta fortemente la linearità dei vincoli, sapendo che lo spazio definito come modello è un poliedro).

Gli elementi di un modello di programmazione sono:

- *insiemi*: raggruppano gli elementi del sistema;
- *parametri/coefficienti tecnologici*: sono i dati del problema e rappresentano delle quantità fissate che dipendono dai diversi elementi del sistema (e dalla tecnologia del sistema stesso);
- *variabili decisionali o di controllo*: sono le grandezze del sistema di cui non conosciamo il valore (assimilabili a delle incognite) e sulle quali possiamo agire per determinare diverse soluzioni alternative del problema;
- *vincoli (incognite)*: sono delle relazioni matematiche che descrivono le condizioni di ammissibilità delle soluzioni. Servono quindi per discriminare le combinazioni di valori delle variabili decisionali che rappresentano soluzioni accettabili del problema, da quelle che non lo sono;
- *funzione obiettivo (relazione sulle incognite)*: è la quantità da massimizzare o minimizzare, espressa come funzione delle variabili decisionali.

Concettualmente si ha che:

- Un *modello* di programmazione matematica *dichiara* le caratteristiche della soluzione cercata (che cosa), piuttosto che definire la strategia per la ricerca della soluzione stessa (*come*).
 - Per esempio, nel caso del modello della lattuga e patate, si sta dicendo:
 - Qual è la soluzione ottima tale che ogni vincolo sia rispettato, ottenendo il valore massimo possibile (non dicendo *come* la si trova)
- La *risoluzione* di un problema di ottimizzazione formulato con un modello di programmazione matematica consiste nella determinazione dei valori delle variabili che soddisfano tutti i vincoli e massimizzano o minimizzano il valore della funzione obiettivo
- I *modelli di programmazione lineare* sono una particolare classe di modelli di programmazione matematica in cui:
 - la funzione obiettivo è un'espressione lineare delle variabili decisionali;
 - i vincoli sono determinati da un sistema di equazioni e/o disequazioni lineari.

Nell'esercizio di modellazione, attenzione a inserire solo modelli lineari; le variabili, di qualsiasi natura esse siano, possono essere solamente moltiplicate per una costante e sommate fra loro.

Quella che segue è la formulazione (con le disuguaglianze che sono sempre $\leq$ o $\geq$ e non segni stretti, in quanto pericoloso da un punto di vista computazionale):

$$\min (\max) z = c_1x_1 + c_2x_2 + \dots + c_jx_j + \dots + c_nx_n \quad (+cost.)$$

soggetto a (s.t.)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1j}x_j + \dots + a_{1n}x_n &\geq (=, \leq) b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2j}x_j + \dots + a_{2n}x_n &\geq (=, \leq) b_2 \\ &\dots \end{aligned}$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mj}x_j + \dots + a_{mn}x_n \geq (=, \leq) b_m$$

$$x_j \in \mathbb{R}_+ (x_j \in \mathbb{Z}_+) \quad \forall j = 1, \dots, n$$

Si noti che s. t. intende "sotto il vincolo" (visto come "subject to/soggetto a")

dove:

z è la funzione obiettivo da minimizzare ($\min$) o massimizzare ($\max$);

$x_j \quad \forall j = 1, \dots, n$, sono le variabili decisionali (incognite) che possono essere reali (eventualmente non negative), intere (eventualmente non negative) o binarie ($x_j \in \{0, 1\}$);

$c_j \quad \forall j = 1, \dots, n$, sono i coefficienti di costo ($\min$) o profitto ($\max$) e sono costanti note del problema (parametri);

$a_{ij} \quad \forall i = 1, \dots, m \text{ e } \forall j = 1, \dots, n$, sono i coefficienti tecnologici (parametri);

$b_i \quad \forall i = 1, \dots, m$, sono i termini noti (parametri).

Programmazione lineare intera (PLI), quindi tutte le variabili possono assumere valori interi

Costanti note

I modelli di programmazione lineare sono utili avendo buona valenza operativa generale; noi ci concentriamo *sulla formulazione, non sulla risoluzione*, dato che possono essere facilmente implementati su calcolatore per ottenere la soluzione ottima. Privilegiamo la semplicità del modello per velocizzare la risoluzione. Vediamo degli esempi semplici di modellazione:

1) Gioco di assemblaggio (telecomandi) – [cfr. pag. 7 "Note di Programmazione Lineare", ma cambiano i valori della f.o.]

Per l'assemblaggio di telecomandi, si hanno a disposizione 10 moduli display, 18 moduli di logica di controllo, 12 moduli di trasmissione, 21 tastierini, 9 moduli di navigazione e 10 moduli led. I telecomandi sono di due tipi. Il tipo A richiede un display, un modulo di navigazione, 2 tastierini, 2 moduli di logica, un modulo di trasmissione e un led. Il tipo B richiede 2 display, 3 tastierini, 2 moduli di logica e 3 moduli di trasmissione. Considerando che il tipo A permette un guadagno netto di 4 euro e il tipo B di 6 euro, determinare la produzione che massimizza il guadagno.

1) Individuare le variabili decisionali

- # (numero) telecomandi tipo A, x_A
- # telecomandi tipo B, x_B

2) Determinare l'obiettivo (funzione obiettivo da ottimizzare)

Si vuole massimizzare la quantità da 4 euro per A e da 6 euro per tipo B, quindi:

$$\max \quad 4x_A + 6x_B$$

Descrivere le condizioni di ammissibilità delle soluzioni (sistemi di equazioni/disequazioni che esprimono i vincoli del problema)

Quindi, dati i profitti unitari, vogliamo fare in modo che, per le singole risorse, non si superi la disponibilità:

$$\begin{aligned} x_A + 2x_B &\leq 10 && \text{(display)} \\ x_A &\leq 9 && \text{(navigazione)} \\ 2x_A + 3x_B &\leq 21 && \text{(tastierini)} \\ 2x_A + 2x_B &\leq 18 && \text{(logica)} \\ x_A + 3x_B &\leq 12 && \text{(trasmissione)} \\ x_A &\leq 10 && \text{(led)} \end{aligned}$$

Sapendo che, ad esempio:

$$4[\text{€/tel}] \quad x[\text{tel}]$$

Le variabili devono essere intere e positive

4) Esplicitare il dominio delle variabili: $x_A \geq 0$, $x_B \geq 0$, $x_A \in \mathbb{Z}$, $x_B \in \mathbb{Z}$ (dominio)

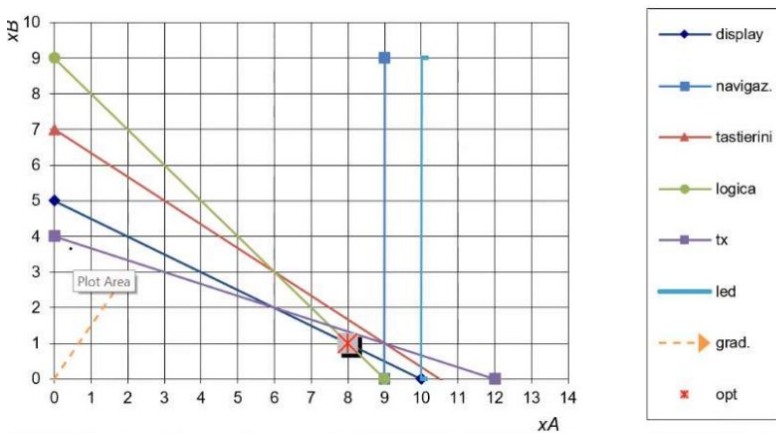
Si può risolvere il problema graficamente? Sì (esempio in Excel del prof), grazie al fatto che il vertice ottimo è intero. Riprendiamo l'esempio delle magliette, esprimibile anch'essa come modello di PLI.

Quando si va nella direzione del gradiente, quando il modello è rappresentato graficamente, va nella direzione del vertice ottimo, il quale deve essere l'ultimo punto toccato dalle linee di livello all'interno della zona della soluzione ammissibile (sapendo qual è l'ultimo punto toccato; esso, però, deve essere intero).

Con Excel, per esempio, si può usare la funzione *SOLVER* che trova la soluzione ottima.

Essa viene vista in laboratorio, but still: <https://support.microsoft.com/en-us/office/define-and-solve-a-problem-by-using-solver-5d1a388f-079d-43ac-a7eb-f63e45925040>

Quando si opera su due dimensioni, è più difficile, perché la soluzione ottima non sempre è vicina a dove siamo (essendo la soluzione intera, è un "punto che sta in mezzo" e non si riesce a caratterizzarlo bene, quindi complicando la risoluzione).



L'esempio della prima lezione sulle magliette ha questo testo:

Money makers

Un gruppo di ragazzi vuole ricavare il più possibile vendendo agli amici magliette e borse decorate. Sono disponibili 10 magliette di cotone e 15 borse di tela e, per la decorazione, 32 riquadri disegnati e 40 profili rossi. Su ogni maglietta vengono apposti 6 riquadri e 2 profili, e su ogni borsa 3 riquadri e 5 profili. Sono anche disponibili 15 bottoni, e ogni borsa ne utilizzerà due per la chiusura. Sono state anche preparate 22 etichette, da apporre una su ogni maglietta e due su ogni borsa. Considerando che ogni maglietta decorata è venduta a 24 euro e ogni borsa a 16 euro, e che gli amici compreranno tutte le magliette e le borse, determinare la produzione che massimizza il ricavo.

La relativa modellazione segue dalla parte di lab, ma la mostro *for the sake of completeness/per completezza*:

xM	xB		f.o.			
24	16		max			
2	6		144			
1	2	14	<=	22	etichette	8,00
6	3	30	<=	32	riquadri	2,00
0	2	12	<=	15	bottoni	3,00
2	5	34	<=	40	profili	6,00
1	0	2	<=	10	magliette	8,00
0	1	6	<=	15	borse	9,00

3) Dieta economica – [cfr. pag. 8 “Note di Programmazione Lineare”]

Un dietologo deve preparare una dieta che garantisca un apporto giornaliero di proteine, ferro e calcio di almeno 20 mg, 30 mg e 10 mg, rispettivamente. Il dietologo è orientato su cibi a base di verdura (5 mg/kg di proteine, 6 mg/kg di ferro e 5 mg/kg di calcio, al costo di 4 €/kg), carne (15 mg/kg di proteine, 10 mg/kg di ferro e 3 mg/kg di calcio, al costo di 10 €/kg) e frutta (4 mg/kg di proteine, 5 mg/kg di ferro e 12 mg/kg di calcio, al costo di 7 €/kg). Determinare la dieta di costo minimo.

Vincoli: $x_V, x_C, x_D \in R$ e $x_V, x_C, x_D \geq 0$ (non serve siano intere, in quanto anche qui la quantità proteica non può, logicamente, essere frazionaria; non mi serviranno 5,67 mg. di proteine, in concreto).

Una guida, spesso, nella modellazione è la funzione obiettivo.

In questo caso, dobbiamo superare le soglie delle proteine, del ferro e del calcio (quindi $\geq$)

Variabili decisionali

- x_1 : quantità in kg di cibi a base di verdura da includere nella dieta;
- x_2 : quantità in kg di cibi a base di carne da includere nella dieta;
- x_3 : quantità in kg di cibi a base di frutta da includere nella dieta.

Come si vede, alle variabili

x_1, x_2, x_3
corrispondono
 x_V, x_C, x_D

Modello PL

$$\min 4x_1 + 10x_2 + 7x_3 \quad (\text{costo giornaliero della dieta})$$

Esempio esteso calcolo proteine:

$$5(mg_p/kg_v)x_v(kg_v) + 15(mg_c/kg_c)x_c(kg_c) + 4(mg_f/kg_f)x_f(kg_f) \geq 20$$

Che diventa:

$$\begin{aligned} 5x_1 + 15x_2 + 4x_3 &\geq 20 \quad (\text{proteine}) \\ 6x_1 + 10x_2 + 5x_3 &\geq 30 \quad (\text{ferro}) \\ 5x_1 + 3x_2 + 12x_3 &\geq 10 \quad (\text{calcio}) \end{aligned}$$

Si può risolverlo anche questo con il metodo grafico, risultando un piano nello spazio.

Se al posto di avere il fruttivendolo di questo esercizio, avessimo delle scatolette:

- Vanno aggiunte delle costanti moltiplicative per il peso dei barattoli e si aggiunge un vincolo intero alle variabili; quindi, si avrebbe ad esempio $x_j \in Z_+$ (non $\in R_+$ come invece appare da slide)

Esercizi di Programmazione Lineare: Modello dieta e modello di costo minimo e vari esempi

Riportiamo l'esercizio (3), *Indagine di mercato (copertura di costo minimo, vedi sotto)* [cfr. pag. 8 "Note di Programmazione lineare"]

Un'azienda pubblicitaria deve svolgere un'indagine di mercato per lanciare un nuovo prodotto. Si deve contattare telefonicamente un campione significativo di persone: almeno 150 donne sposate, almeno 110 donne non sposate, almeno 120 uomini sposati e almeno 100 uomini non sposati. Le telefonate possono essere effettuate al mattino (al costo operativo di 1.1 €) o alla sera (al costo di 1.6 €). Le percentuali di persone mediamente raggiunte sono riportate nella seguente tabella

	Mattino	Sera
Donne sposate	30%	30%
Donne non sposate	10%	20%
Uomini sposati	10%	30%
Uomini non sposati	10%	15%
Nessuno	40%	5%

Le variabili decisionali sono:

- x_1 , telefonate al mattino
- x_2 , telefonate alla sera

Il dominio è sugli interi positivi, dunque:

$$x_1, x_2 \geq 0 \text{ e } x_1, x_2 \in \mathbb{Z}_+, \forall i \in \{1, 2, 3\}, x_i \in \mathbb{R}_+$$

L'obiettivo è di minimizzare il costo delle telefonate, dunque la funzione obiettivo:

$$Z = \min 1.1x_1 + 1.6x_2$$

affinché:

- $0.3x_1 + 0.3x_2 \geq 150$ per le donne sposate
- $0.1x_1 + 0.2x_2 \geq 110$ per le donne non sposate
- $0.1x_1 + 0.3x_2 \geq 120$ per gli uomini sposati
- $0.1x_1 + 0.15x_2 \geq 100$ per gli uomini non sposati

Come teniamo conto dei nessuno (le chiamate a vuoto)?

- Non abbiamo un caso migliore o peggiore delle telefonate, infatti non si ha controllo rispetto a quante vadano effettivamente a vuoto. Esse sono una "seccatura" già compresa nei vincoli, in quanto sono una percentuale di telefonate che non contribuisce a nulla (da un punto di vista di modello di programmazione lineare, un valore pari a 0 non fornisce alcuna informazione utile)

Altro esercizio → (4), *Trasporto di frigoriferi (problema di trasporto, vedi sotto)* [cfr. pag. 9 "Note di Programmazione lineare"]

Una ditta di produzione di elettrodomestici produce dei frigoriferi in tre stabilimenti e li smista in quattro magazzini intermedi di vendita. La produzione settimanale nei tre stabilimenti A, B e C è rispettivamente di 50, 70 e 20 unità. La quantità richiesta dai 4 magazzini è rispettivamente di 10, 60, 30 e 40 unità. I costi per il trasporto di un frigorifero tra gli stabilimenti e i magazzini 1, 2, 3 e 4 sono i seguenti:

- dallo stabilimento A: 6, 8, 3, 4 euro;
- dallo stabilimento B: 2, 3, 1, 3 euro;
- dallo stabilimento C: 2, 4, 6, 5 euro.

La ditta vuole determinare il piano di trasporti di costo minimo.

Variabili decisionali e insiemi:

$x_{(ij)}$: # frigo prodotti da stabilimento i e smistati nel magazzino j , $\forall i \in \{A, B, C\} \forall j \in \{1, 2, 3, 4\}$, $x_{(ij)} \in \mathbb{Z}_+$

$I = \{A, B, C\}$ stabilimenti;

$J = \{1, 2, 3, 4\}$ magazzini.

Funzione obiettivo $\rightarrow$ costo minimo

$$\begin{aligned} \min \quad & 6x_{A1} + 8x_{A2} + 3x_{A3} + 4x_{A4} + \\ & 2x_{B1} + 3x_{B2} + 1x_{B3} + 3x_{B4} + \\ & 2x_{C1} + 4x_{C2} + 6x_{C3} + 5x_{C4} \end{aligned}$$

Vincoli:

$$x_{A1} + x_{A2} + x_{A3} + x_{A4} \leq 50 \quad (\text{capacità produttiva stabilimento A})$$

$$x_{B1} + x_{B2} + x_{B3} + x_{B4} \leq 70 \quad (\text{capacità produttiva stabilimento B})$$

$$x_{C1} + x_{C2} + x_{C3} + x_{C4} \leq 20 \quad (\text{capacità produttiva stabilimento C})$$

$$x_{A1} + x_{B1} + x_{C1} \geq 10 \quad (\text{domanda magazzino 1})$$

$$x_{A2} + x_{B2} + x_{C2} \geq 60 \quad (\text{domanda magazzino 2})$$

$$x_{A3} + x_{B3} + x_{C3} \geq 30 \quad (\text{domanda magazzino 3})$$

$$x_{A4} + x_{B4} + x_{C4} \geq 40 \quad (\text{domanda magazzino 4})$$

Quanto visto fino ad ora sono problemi “classici”, cosiddetti modelli di mix ottimo di produzione.

$$\begin{aligned} \max \quad & \sum_{i \in I} P_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \leq Q_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I \end{aligned}$$

dove

I insieme dei beni che possono essere prodotti;

J insieme delle risorse disponibili;

P_i profitto (unitario) per il bene $i \in I$;

Q_j quantità disponibile della risorsa $j \in J$;

A_{ij} quantità di risorsa j necessaria per la produzione di un'unità del bene i .

Considerando gli esempi sopra proposti, rientrano in questo schema i problemi di assemblaggio di telefonini e del contadino.

Tradotto in termini testuali:

- si ha un insieme di beni in cui, in base anche alle stesse risorse disponibili, non si deve mai superare una certa quantità sui vincoli e si sa che si ottiene un certo profitto rispetto ai singoli beni (sapendo che, per dominio, queste quantità devono esistere e, come tali, devono essere intere positive).
- in generale, si cerca di massimizzare, dunque si ha una somma con un *max* davanti e ogni vincolo possiede un $\leq$ rispetto ai singoli vincoli

Altri problemi sono quelli di copertura di costo minimo (problemi di dieta/diet problems):

Modelli di copertura di costo minimo

$$\begin{aligned} \min \quad & \sum_{i \in I} C_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \geq D_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I \end{aligned}$$

dove

I insieme delle risorse da acquistare;

J insieme delle domande da coprire;

C_i costo (unitario) per l'utilizzo della risorsa $i \in I$;

D_j ammontare della domanda di $j \in J$;

A_{ij} capacità (unitaria) della risorsa i di soddisfare la domanda j .

Considerando gli esempi sopra proposti, rientrano in questo schema i problemi della dieta economica, dell'indagine di mercato, dei turni in ospedale, della localizzazione di servizi.

Tradotto in termini testuali:

- si ha un insieme di risorse per cui ognuna deve coprire un certo numero di domande; ognuna di queste, a prescindere dalla quantità effettiva, non si considera un costo ulteriore (infatti, per ogni singola risorsa e quantità si ha costo unitario), tale che, per minimo, si soddisfi una quantità che soddisfa una specifica domanda. In generale, dato che sia le risorse che le domande normalmente esistono, esse sono interi positivi per dominio.
- in generale, si cerca di minimizzare, dunque si ha una somma con un *min* davanti e ogni vincolo possiede un $\geq$ rispetto ai singoli vincoli

Altro problema classico individuato da uno schema generale sono i problemi di trasporto:

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in J} x_{ij} \leq O_i \quad \forall i \in I \\ & \sum_{i \in I} x_{ij} \geq D_j \quad \forall j \in J \\ & x_{ij} \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I, j \in J \end{aligned}$$

dove

I insieme dei centri di offerta;

J insieme dei centri di domanda;

C_{ij} costo (unitario) per il trasporto da $i \in I$ a $j \in J$;

O_i ammontare dell'offerta in $i \in I$;

D_j ammontare della domanda in $j \in J$.

Considerando gli esempi sopra proposti, rientra in questo schema il problema del trasporto di frigoriferi.

Tradotto in termini testuali:

- si considerano molteplici domini, in particolare sapendo che parlando di trasporti si considerano le numerosi variabili in gioco (per esempio, stabilimenti/impianti produttivi, magazzini, ecc.), che non inficiano sulla descrizione del modello in sé
- l'obiettivo, essendo un trasporto, è normalmente una somma minimizzata (per esempio nel caso di una distanza), considerando che si vuole massimizzare la produzione (dunque, ogni vincolo sarà seguito da un $\leq$)

Su un esempio concreto con indici rispetto a quest'ultimo modello (cioè, dei trasporti):

$$I = 8 \text{ e } J = 20$$

Avremmo 160 variabili decisionali ($20 * 8$) e 28 vincoli ($20 + 8$).

Domanda: Non avrebbe più senso mettere il segno “=” e non “≤” o “≥”?

Risposta: In realtà no, poiché sarebbe una quantità *al limite*, dunque approssimativamente arriviamo alla quantità detta. Normalmente, questo si intende dai vincoli del testo o, comunque, dalla natura del problema. Se necessario si mette, altrimenti non serve.

Altra domanda: Potrà capitare una situazione in cui uno dei vincoli è rispettato con il maggiore stretto “>”?

Risposta: È ammissibile come soluzione, ma dipende dalla natura in problema.

Nel modello sopra (trasposto), non è ammissibile in quanto non è un problema di minimo/ottimo; in questo caso, quindi, non esiste soluzione che rispetti tale vincolo.

Vediamo un altro problema (5), *Turni in ospedale* (problema di copertura di costo minimo)

[cfr. pag. 10 “Note di Programmazione lineare”]

Si vogliono organizzare i turni degli infermieri in ospedale. Ogni infermiere lavora 5 giorni consecutivi, indipendentemente da come sono collocati all'interno della settimana, e poi ha diritto a due giorni consecutivi di riposo. Le esigenze di servizio per i vari giorni della settimana richiedono la presenza di 17 infermieri il lunedì, 13 il martedì, 15 il mercoledì, 19 il giovedì, 14 il venerdì, 16 il sabato e 11 la domenica. Organizzare il servizio in modo da minimizzare il numero totale di infermieri da impegnare.

Variabili decisionali

Sapendo che i turni sono compresi per singolo giorno, impiegando una specifica quantità di infermieri, idealmente abbiamo:

- *Lun* (17) gli infermieri che lavorano il lunedì
- *Mar* (13) gli infermieri che lavorano il martedì,
- *Mer* (15) gli infermieri che lavorano il mercoledì,
- *Gio* (19) gli infermieri che lavorano il giovedì,
- *Ven* (14) gli infermieri che lavorano il venerdì,
- *Sab* (16) gli infermieri che lavorano il sabato,
- *Dom* (11) gli infermieri che lavorano la domenica

$$\begin{array}{ll}
 \min & \text{lun} + \text{mar} + \text{mer} + \text{gio} + \text{ven} + \text{sab} + \text{dom} \\
 \text{s.t.} & \\
 & \text{lun} + \text{gio} + \text{ven} + \text{sab} + \text{dom} \geq 17 \quad (\text{presenze lunedì}) \\
 & \text{lun} + \text{mar} + \text{ven} + \text{sab} + \text{dom} \geq 13 \quad (\text{presenze martedì}) \\
 & \text{lun} + \text{mar} + \text{mer} + \text{sab} + \text{dom} \geq 15 \quad (\text{presenze mercoledì}) \\
 & \text{lun} + \text{mar} + \text{mer} + \text{gio} + \text{dom} \geq 19 \quad (\text{presenze giovedì}) \\
 & \text{lun} + \text{mar} + \text{mer} + \text{gio} + \text{ven} \geq 14 \quad (\text{presenze venerdì}) \\
 & \text{mar} + \text{mer} + \text{gio} + \text{ven} + \text{sab} \geq 16 \quad (\text{presenze sabato}) \\
 & \text{mer} + \text{gio} + \text{ven} + \text{sab} + \text{dom} \geq 11 \quad (\text{presenze domenica}) \\
 & \text{lun}, \text{mar}, \text{mer}, \text{gio}, \text{ven}, \text{sab}, \text{dom} \in \mathbb{Z}_+ \quad (\text{dominio})
 \end{array}$$

(Si può, volendo, commentare come si fa normalmente codice anche in esame, ad esempio come un commento alla C++, quindi // ; nel mio file, per completezza, e negli esercizi da me risolti, commento in maniera estesa le scelte operate. All'esame si metta variabile e doppio slash e basta).

// x_i : # infermieri del turno i -esimo

Esempio del lunedì $\rightarrow x_1 + x_4 + x_5 + x_6 + x_7 \geq 17$

Esempio del martedì $\rightarrow 1x_1 + 1x_2 + 0x_3 + 0x_4 + x_5 + x_6 + x_7$

Scritto da Gabriel

Dominio $\rightarrow x_i \in \mathbb{Z}_+$ **Altro esercizio (6), Localizzazione di servizi (copertura di costo minimo)**

Una città è divisa in sei quartieri, dove si vogliono attivare dei centri unificati di prenotazione (CUP) per servizi sanitari. In ciascun quartiere è stata individuata una possibile località di apertura. Le distanze medie in minuti da ciascun quartiere a ciascuna delle possibili località è indicata in tabella.

	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6
Quartiere 1	5	10	20	30	30	20
Quartiere 2	10	5	25	35	20	10
Quartiere 3	20	25	5	15	30	20
Quartiere 4	30	35	15	5	15	25
Quartiere 5	30	20	30	15	5	14
Quartiere 6	20	10	20	25	14	5

Si desidera che nessun utente abbia un tempo medio di spostamento superiore a 15 minuti per arrivare al CUP più vicino e si vuole minimizzare il numero di CUP attivati.

Esercizi di Programmazione Lineare: Modello di trasporto e vari esempi

Riprendendo l'esercizio di cui sopra:

$$x_i \begin{cases} 1 & \text{se apro CUP } i\text{-esimo} \\ 0 & \text{altrimenti} \end{cases} \quad \begin{array}{l} \text{Insiemi} \\ I = \{1, 2, \dots, 6\} \text{ località.} \end{array}$$

Le variabili devono essere binarie. Si consideri che, a parità di costo, si utilizzano le variabili con lo stesso valore per ogni quartiere (perché non si sa se aprano effettivamente o meno), a patto che ogni singola variabile sia ≤ 15 .

Funzione obiettivo (ce ne sta sempre una sola): $\min x_1 + x_2 + \dots + x_6 \in \{0,1\}$

Vincoli:

$$\begin{array}{llllllllll} \min & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 \\ \text{s.t.} & & & & & & & & & & & \\ x_1 + x_2 & \geq 1 & \text{(quartiere 1)} & & & & & & & & & \\ x_1 + x_2 + x_6 & \geq 1 & \text{(quartiere 2)} & & & & & & & & & \\ x_3 + x_4 & \geq 1 & \text{(quartiere 3)} & & & & & & & & & \\ x_3 + x_4 + x_5 & \geq 1 & \text{(quartiere 4)} & & & & & & & & & \\ x_4 + x_5 & \geq 1 & \text{(quartiere 5)} & & & & & & & & & \\ x_2 + x_5 + x_6 & \geq 1 & \text{(quartiere 6)} & & & & & & & & & \\ & x_1, x_2, x_3, x_4, x_5, x_6 & \in & \{0,1\} & \text{(dominio)} & & & & & & & \end{array}$$

Ora, esercizio *Produzione e forza lavoro* (7) $\rightarrow$ modello di mix ottimo di produzione

[cfr. pag. 14 “Note di Programmazione lineare”]

Produzione e forza lavoro

Un'azienda produce i modelli I, II e III di un certo prodotto a partire dai materiali grezzi A e B, di cui sono disponibili 4000 e 6000 unità, rispettivamente. In particolare, ogni unità del modello I richiede 2 unità di A e 4 di B; un'unità del modello II richiede 3 unità di A e 2 di B; ogni unità del modello III richiede 5 unità di A e 7 di B. Il modello I richiede una forza lavoro doppia rispetto al modello II e tripla rispetto al modello III. La forza lavoro disponibile è in grado di produrre al massimo l'equivalente di 700 unità del modello I. Il settore marketing dell'azienda ha reso noto che la domanda minima per ciascun modello è rispettivamente di 200, 200 e 150 unità, al prezzo di 30, 20 e 50 euro. Si vuole massimizzare il profitto totale.

Variabili decisionali:

$$x_1, x_2, x_3 \geq 0$$

Funzione obiettivo:

$$\max 30x_1 + 20x_2 + 50x_3$$

$$s. t. x_1, x_2, x_3 \in R_+$$

$$\max 30x_I + 20x_{II} + 50x_{III}$$

s.t.

$$x_I \geq 200 \quad (\text{vincoli sulla domanda})$$

$$x_{II} \geq 200$$

$$x_{III} \geq 150$$

Vincoli:

(Per i materiali)

$$2x_1 + 3x_2 + 5x_3 \leq 4000$$

$$4x_1 + 2x_2 + 7x_3 \leq 6000$$

(Per la forza lavoro)

$$1x_1 + 1/2x_2 + 1/3x_3 \leq 700$$

(Per la domanda)

$$x_1 \geq 200$$

$$x_2 \geq 200$$

$$x_3 \geq 150$$

$$2x_I + 3x_{II} + 5x_{III} \leq 4000 \quad (\text{vincoli sui materiali})$$

$$4x_I + 2x_{II} + 7x_{III} \leq 6000$$

$$x_I + \frac{1}{2}x_{II} + \frac{1}{3}x_{III} \leq 700 \quad (\text{vincoli sulla forza lavoro})$$

$$x_i \in \mathbb{Z}_+, \forall i \in \{I, II, III\} \quad (\text{dominio})$$

Dominio: $x_i \in \mathbb{Z}_+, \forall i \in \{1, 2, 3\}$

Passiamo ad un altro esercizio: *Produzione e capacità eccedente (8), mix ottimo di produzione* [cfr. pag. 15 “Note di Programmazione lineare”]

Una società ha tre impianti con capacità produttiva eccedente. Tutti e tre gli impianti sono in grado di produrre schiume di lattice e si è deciso di sfruttare in questo modo la capacità produttiva in eccesso. Le schiume possono essere realizzate in tre diverse densità (bassa, media e alta) che forniscono un profitto netto unitario di 9, 10 e 12 euro. Gli stabilimenti 1, 2 e 3 hanno manodopera e capacità produttiva in eccesso per produrre 500, 600 e 300 quintali al giorno, indipendentemente dalla densità delle schiume. Comunque, la disponibilità dello spazio destinato all'immagazzinamento durante il processo produttivo limita la produzione. Gli stabilimenti 1, 2 e 3 hanno, rispettivamente, 900, 800 e 350 mq di magazzino disponibile per questo prodotto. Ogni quintale di schiuma prodotta al giorno in densità bassa, media o alta richiede 2, 1.5 e 1 mq, rispettivamente. Le previsioni di vendita indicano che si possono vendere al massimo 600, 800 e 500 quintali delle schiume di densità bassa, media e alta, rispettivamente. I sindacati hanno chiesto di mantenere un carico di lavoro uniforme e la direzione ha concordato che sarà utilizzata la medesima percentuale della capacità produttiva in eccesso. La direzione ci chiede di determinare come suddividere la produzione per massimizzare il profitto totale.

Variabili decisionali:

// $x_{(B/M/A)}$: quantità di schiuma B/M/A (cioè, rispettivamente, densità bassa/media/alta)

Scritto da Gabriel

// $x_{(ij)}$: quantità di schiuma di tipo i in stabilimento j

Funzione obiettivo:

$\max 9x_B + 10x_M + 12x_A$ (scritta qui a lato in forma estesa)

$$\begin{aligned} x_B &= x_{(B1)} + x_{(B2)} + x_{(B3)} \\ x_M &= x_{(M1)} + x_{(M2)} + x_{(M3)} \\ x_A &= x_{(A1)} + x_{(A2)} + x_{(A3)} \end{aligned}$$

$$\max \begin{aligned} &9(x_{1b} + x_{2b} + x_{3b}) + \\ &10(x_{1m} + x_{2m} + x_{3m}) + \\ &12(x_{1a} + x_{2a} + x_{3a}) \end{aligned}$$

s.t.

(Manodopera)

$$x_{(B1)} + x_{(M1)} + x_{(A1)} \leq 500$$

$$x_{(B2)} + x_{(M2)} + x_{(A2)} \leq 600$$

$$x_{(B3)} + x_{(M3)} + x_{(A3)} \leq 300$$

$$x_{1b} + x_{1m} + x_{1a} \leq 500$$

$$x_{2b} + x_{2m} + x_{2a} \leq 600$$

$$x_{3b} + x_{3m} + x_{3a} \leq 300$$

(disponibilità manodopera)

(Disponibilità magazzini)

$$2x_{(B1)} + 1.5x_{(M1)} + 1x_{(A1)} \leq 900$$

$$2x_{(B2)} + 1.5x_{(M2)} + 1x_{(A2)} \leq 800$$

$$2x_{(B3)} + 1.5x_{(M3)} + 1x_{(A3)} \leq 350$$

$$2x_{1b} + 1.5x_{1m} + 1x_{1a} \leq 900$$

$$2x_{2b} + 1.5x_{2m} + 1x_{2a} \leq 800$$

$$2x_{3b} + 1.5x_{3m} + 1x_{3a} \leq 350$$

(disponibilità magazzini)

$$x_{1b} + x_{2b} + x_{3b} \leq 600$$

$$x_{1m} + x_{2m} + x_{3m} \leq 800$$

$$x_{1a} + x_{2a} + x_{3a} \leq 500$$

(previsioni di vendita)

(Previsioni di vendita)

$$x_{(B1)} + x_{(B2)} + x_{(B3)} \leq 600$$

$$x_{(M1)} + x_{(M2)} + x_{(M3)} \leq 800$$

$$x_{(A1)} + x_{(A2)} + x_{(A3)} \leq 500$$

$$\frac{1}{500}(x_{1b} + x_{1m} + x_{1a}) = \frac{1}{600}(x_{2b} + x_{2m} + x_{2a}) \quad (\text{uniformità del carico di lavoro})$$

$$\frac{1}{500}(x_{1b} + x_{1m} + x_{1a}) = \frac{1}{300}(x_{3b} + x_{3m} + x_{3a})$$

$$x_{ij} \in \mathbb{R}_+, \forall i \in \{1, 2, 3\}, \forall j \in \{b, m, a\} \quad (\text{dominio})$$

$$x_B \leq 600, x_M \leq 800, x_A \leq 500$$

(Uniformità del carico di lavoro) $\rightarrow$ Si ragiona "in percentuale" ((n. di partenza * n. percentuale)/100)

Questo viene fatto sulla somma delle produzioni di schiuma di densità bassa, media, alta sulla base dei dati di manodopera e capacità in eccesso, dunque appunto 500, 600 e 300.

$$(x_{(B1)} + x_{(M1)} + x_{(A1)})/500 = (x_{(B2)} + x_{(M2)} + x_{(A2)})/600$$

$$(x_{(B1)} + x_{(M1)} + x_{(A1)})/500 = ((x_{(B3)} + x_{(M3)} + x_{(A3)}))/300$$

Dominio: $x_{(ij)} \geq 0, x_{(ij)} \in \mathbb{R}_+$

Altro esercizio, *Piani di investimento (9)* $\rightarrow$ schema multi-periodale (vedi sotto)

[cfr. pagg. 16/17 "Note di Programmazione lineare"]

Un finanziere ha due piani di investimento A e B disponibili all'inizio di ciascuno dei prossimi cinque anni. Ogni euro investito in A all'inizio di ogni anno garantisce, due anni più tardi, un profitto di 0,4 euro (e può essere immediatamente reinvestito). Ogni euro investito in B all'inizio di ogni anno dà, tre anni dopo, un profitto di 0,7 euro. In più, da un certo momento in avanti, sarà possibile sfruttare anche i piani di investimento C e D. Ogni euro investito in C all'inizio del secondo anno raddoppierà dopo 4 anni. Ogni euro investito in D all'inizio del quinto anno darà un profitto di 0,3 euro l'anno successivo. Anche per i piani B, C e D vale la possibilità di reinvestimento, come per il piano A. Il finanziere ha a disposizione 10000 euro e vuole sapere quale piano di investimento massimizza il capitale posseduto all'inizio del sesto anno.

Variabili decisionali:

$x_{(ij)}$: capitale investito nel piano i all'inizio dell'anno j , $\forall i \in \{A, B, C, D\} \forall j \in \{1, 2, 3, 4, 5\}$

In queste, non si considerano una serie di variabili, corrispondenti a piani di investimento non disponibili o investimenti che non rientrano all'inizio del sesto anno (e quindi, data la natura lineare del modello, varrebbero zero e non si considerano), nello specifico:

$$x_{(A5)}, x_{(B4)}, x_{(B5)}, x_{(C1)}, x_{(C3)}, x_{(C4)}, x_{(C5)}, x_{(D1)}, x_{(D2)}, x_{(D3)}, x_{(D4)}$$

Per la funzione obiettivo, si considera anche il ritorno dagli investimenti, per cui ogni anno si cerca di capire quanto è stato investito e quanto, potenzialmente, ritorna:

$$\max \quad 0.4(x_{A1} + x_{A2} + x_{A3} + x_{A4}) + 0.7(x_{B1} + x_{B2} + x_{B3}) + x_{C2} + 0.3x_{D5} + 10000$$

Vincoli:

$$x_{(A1)} + x_{(B1)} \leq 10000 \text{ // 1° anno}$$

$x_{(A2)} + x_{(B2)} + x_{(C2)} \leq 10000 - x_{(A1)} - x_{(B1)}$ //2° anno (con un eventuale reinvestimento del capitale del primo di A e di B e sapendo che reinvestendo su C all'inizio del secondo anno, si avrà un profitto dopo quattro)

$x_{(A3)} + x_{(B3)} \leq 10000 + 0.4x_{(A1)} - x_{(B1)} - x_{(A2)} - x_{(B2)} - x_{(C2)}$ //3° anno (sapendo che si ha il profitto di 0.4, due anni più tardi su A, quindi $A1+0.4$ e si tolgono tutti i reinvestimenti degli anni precedenti su A e su B. Inoltre, C3 è pari a 0 dato che non si ha guadagno e non conta qui, essendo modello lineare)

$x_{(A4)} \leq 10000 + 0.7x_{(B1)} + 0.4x_{(A1)} + 0.4x_{(A2)} - x_{(B2)} - x_{(C2)} - x_{(A3)} - x_{(B3)}$ //4° anno (sapendo che ci sono i profitti che arrivano da parte di A1 e A2 essendo passati due anni per entrambi e il profitto di B, dato che sono passati 3 anni, più i reinvestimenti di A, B, C)

$x_{(D5)} \leq 10000 + 0.4x_{(A1)} + 0.4x_{(A2)} + 0.4x_{(A3)} + 0.7x_{(B1)} + 0.7x_{(B2)} - x_{(C2)} - x_{(B3)} - x_{(A4)}$ //5° anno (sapendo che ci sono i profitti che arrivano da parte di A1, A2 e A3 essendo passati due anni per entrambi e il profitto di B1 e B2, dato che sono passati 3 anni, più i reinvestimenti di A, B, C)

Dominio:

$$x_{ij} \in \mathbb{R}_+, \forall i \in \{A, B, C, D\}, \forall j \in \{1, 2, 3, 4, 5\}$$

Il modello di questo tipo viene definito *schema multi-periodale*, considerando un tipo di vincoli che viene ripetuto in diversi periodi di tempo, considerando inoltre guadagni e reinvestimenti.

Esempio ulteriore, *Produzione su più linee (10)*

[cfr. pag. 19 "Note di Programmazione lineare"]

Un mangime è ottenuto miscelando una stessa quantità di tre componenti che possono essere lavorate su quattro linee di produzione differenti. Ogni linea è dotata di una limitata capacità di ore di lavorazione e una diversa produttività (unità di componente per ogni ora), come indicato nella seguente tabella:

Linea	Capacità	Produttività		
		componente 1	componente 2	componente 3
1	100	10	15	5
2	150	15	10	5
3	80	20	5	10
4	200	10	15	20

Si vuole determinare il numero di ore di lavorazione di ciascuna componente su ciascuna linea di produzione in modo da massimizzare la quantità di mangime complessivamente prodotta.

Variabili decisionali:

p_i = quantità componenti $i \in \{1, 2, 3\}$ prodotta

L'idea è che la quantità di mangime completata sia condizionata dalla componente disponibile nella minore quantità in quanto se volessimo massimizzare la produzione, si dovrebbe modulare la quantità massima di lavoro rispetto alle quantità minime impiegate.

$\max 3 * \min \{p_1, p_2, p_3\}$  Non sarebbe lineare, per ora

Ora, ci accorgiamo che la quantità di componente j dipende dalla linea di lavorazione i , e dunque la variabile decisionale diventa come segue:

x_{ij} = numero di ore di lavorazione di ciascun componente j rispetto alla linea i
con un dominio attuale di $\forall i \in \{1, 2, 3, 4\}, \forall j \in \{1, 2, 3\}$

Vincoli:

Capacità oraria delle linee:

$$\begin{aligned} x_{11} + x_{12} + x_{13} &\leq 100 && \text{(ragiona in verticale rispetto alla colonna delle capacità e per ognuno dei tre componenti)} \\ x_{21} + x_{22} + x_{23} &\leq 150 \\ x_{31} + x_{42} + x_{53} &\leq 80 \\ x_{31} + x_{42} + x_{53} &\leq 200 \end{aligned}$$

Disponibilità delle componenti:

(ragiona in verticale rispetto alla colonna della produttività delle componenti)

$$p_1 \leq 10x_{(11)} + 15x_{(21)} + 20x_{(31)} + 10x_{(41)}$$

$$p_2 \leq 15x_{(21)} + 10x_{(22)} + 5x_{(23)} + 15x_{(42)}$$

$$p_3 \leq 5x_{(13)} + 5x_{(23)} + 10x_{(33)} + 20x_{(43)}$$

Dominio:

$$x_{(ij)} \in R_+, p_j \in R_+, \forall i \in 1, 2, 3, 4, \forall j \in 1, 2, 3, y \in R$$

Vista come sopra, la funzione obiettivo non è lineare, tuttavia lo diventa aggiungendo al posto del pezzo evidenziato dal riquadro una variabile y

$\max \min \{p_1, p_2, p_3\}$  Posso anche chiamarla y

Vanno però aggiunti dei vincoli, dipendenti dalla stessa variabile:

$$y \leq p_1, y \leq p_2, y \leq p_3 \text{ (trasformata, con una variabile in più e tre vincoli in più, in modello lineare)}$$

Nello specifico, i vincoli diventerebbero come segue:

$$\begin{aligned} y &\leq 10x_{11} + 15x_{21} + 20x_{31} + 10x_{41} && \text{(componente 1 disponibile)} \\ y &\leq 15x_{12} + 10x_{22} + 5x_{32} + 15x_{42} && \text{(componente 2 disponibile)} \\ y &\leq 5x_{13} + 5x_{23} + 10x_{33} + 20x_{43} && \text{(componente 3 disponibile)} \end{aligned}$$

Tale che la funzione obiettivo diventi $\max y$

Conclusione Programmazione Lineare –Esercizi, F.o. in forma min-max, max-min e min-abs, Modelli con vincoli di tipo logico, Vincoli di Big-M

Nella formulazione di problemi sotto forma di modelli di programmazione lineare, potremmo incontrare problemi in cui, come obiettivo, si vuole minimizzare il massimo tra un insieme di valori (min-max) (o viceversa). Nell'esempio di ieri, la funzione obiettivo è nella forma (considerando un insieme di valori $e_1, \dots, e_n$ è nella forma *min max*).

Abbiamo diversi casi, in cui però non abbiamo espressioni lineari:

$$\begin{array}{l} \text{max min } \{e_1, e_2, \dots, e_n\} \\ \\ \text{max } y \\ y \leq e_1 \\ y \leq e_2 \\ \dots \\ y \leq e_n \end{array}$$

- **min** non è una funzione lineare
- **max** non è una funzione lineare
- **Valore assoluto** non è una funzione lineare

$$\begin{array}{l} \text{min max } \{e_1, e_2, \dots, e_n\} \\ \\ \text{min } y \\ y \geq e_1 \\ y \geq e_2 \\ \dots \\ y \geq e_n \end{array}$$

$$\begin{array}{l} \text{min } |e| \equiv \text{min max } \{e, -e\} \\ \text{min } y \\ y \geq e \\ y \geq -e \end{array}$$

Per linearizzare il sistema, aggiungiamo una variabile e dei vincoli, linearizzando problemi del tipo:

- *min max* $e_1, \dots, e_n$
- *max min* $e_1, \dots, e_n$

Similmente, esiste una classe il cui obiettivo è minimizzare il valore di una certa quantità e , avendo che la funzione obiettivo sarà della forma *min |e|*, si può vedere come *min max* $\{e, -e\}$, per ricondursi al caso min-max.

Abbiamo un altro esercizio, *Schedulazione just-in-time*, (11), *problema di costo minimo* [cfr. pagg. 20/21 "Note di Programmazione lineare"]

Un server computazionale deve pianificare l'esecuzione di cinque batch su di una macchina mono-processore. I batch durano rispettivamente 5, 7, 4, 7 e 10 minuti. La sequenza di esecuzione 1-2-3-4-5 è data e non ci può essere sovrapposizione temporale tra i batch. Il primo batch ha come ora di consegna desiderata le 10.32, il secondo le 10.38, il terzo le 10.42, il quarto le 10.52 e il quinto le 10.57. La consegna dei batch elaborati deve essere il più puntuale possibile: si paga una penale di 750 euro per ogni minuto di anticipo o ritardo nella consegna. Organizzare i tempi di esecuzione (al minuto) per minimizzare la penale totale.

Variabili decisionali:

i_j : ora di inizio del job $j \in 1,2,3,4,5$ y_j : minuti di anticipo/ritardo del job j , $\forall j \in \{1,2,3,4,5\}$

Funzione obiettivo:

Primo tentativo

$\min 750 |i_1 + 5 - 10:32| + |i_2 + 7 - 10:38| \dots \rightarrow$ non è lineare (val. assoluto e non permette di capire bene il tipo delle variabili (numeri, date, n. reali))

L'ora non è facilmente trattabile con operatori di somma e prodotto e trasformiamo le variabili in minuti, stabilendo il minuto dopo le 10 per cui un batch sia preso in carico dalla macchina. La funzione è del tipo *min-abs*, in quanto per ogni job deve essere associata una durata, una consegna e un'eventuale penalità.

Versione corretta

$\min 750(y_1 + y_2 + y_3 + y_4 + y_5)$

Una generica variabile assumerà, idealmente, come vincolo:

$y_1 \geq i_1 + 5 - 32 \quad y_1 \geq 32 - (i_1 + 5)$

$y_2 \geq \dots$

$y_1 \geq i_1 + 5 - 32 \quad y_1 \geq 32 - 5 - i_1$ (linearizzazione dei valori assoluti)

$y_2 \geq i_2 + 7 - 38 \quad y_2 \geq 38 - 7 - i_2$

$y_3 \geq i_3 + 4 - 42 \quad y_3 \geq 42 - 4 - i_3$

$y_4 \geq i_4 + 7 - 52 \quad y_4 \geq 52 - 7 - i_4$

$y_5 \geq i_5 + 10 - 57 \quad y_5 \geq 57 - 10 - i_5$

Come visto sopra, i vincoli assumono una

penalità del tipo $y_j = |i_j + p_j - d_j|$ e si vede la stessa cosa da un punto di vista di durata

$y_j = |d_j - (i_j + p_j)|$, considerando che i_j è il minuto di inizio del job j , p_j la sua durata e d_j il minuto di consegna desiderato (quindi, linearizzazione dei valori assoluti considerando durata e penalità).

Una piccola nota: per il fatto che la funzione obiettivo è di tipo *min-abs*, i vincoli vengono linearizzati come *+vincolo* e *-vincolo*, come si vede sopra (quindi, proprio i casi del valore assoluto).

Dominio: $i_j \in \mathbb{Z}_+$, $y_j \in \mathbb{R}_+$

Si parla anche della sovrapposizione tra i singoli job, quindi si considera la durata del batch precedente sommata al minuto dopo le 10 di inizio del batch j (non essendoci una i_6 non si considera $i_6 \geq i_5 + 10$).

$i_2 \geq i_1 + 5$ (precedenza e non sovrapposizione tra i batch)

$i_3 \geq i_2 + 7$

$i_4 \geq i_3 + 4$

$i_5 \geq i_4 + 7$

$i_j \in \mathbb{Z}_+, \forall j \in \{1, 2, 3, 4, 5\}$ (dominio)

$y_j \in \mathbb{R}, \forall j \in \{1, 2, 3, 4, 5\}$

Altro esercizio, Localizzazione con costi fissi (con vincoli aggiuntivi) (13), problema di mix ottimo di produzione

Una catena della Grande Distribuzione Organizzata (GDO) dispone di un budget W per l'apertura di nuovi ipermercati in Italia. Gli studi preliminari hanno individuato un insieme I di possibili localizzazioni. Per l'apertura di un ipermercato nella localizzazione i bisogna sostenere un costo fisso F_i (acquisto del terreno, oneri amministrativi etc.) e un costo variabile pari a C_i ogni 100 mq di ipermercato. Per ogni localizzazione, è prevista una dimensione massima U_i e, nel caso di apertura, una dimensione minima pari a L_i . Una volta aperto e entrato a regime, ciascun ipermercato localizzato in i produrrà entrate per R_i ogni 100 mq. Determinare l'insieme di localizzazioni in cui aprire dei nuovi ipermercati e dimensionare gli ipermercati stessi in modo da massimizzare i ricavi complessivi, tenendo conto che non possono essere aperti più di K ipermercati.

Idealmente, pensiamo ad una variabile binaria, legata all'apertura di un ipermercato nella localizzazione i , $\forall i \in I$:

$$y_i = \begin{cases} 1 & \text{se decido di aprire in } i \in I \\ 0 & \text{altrimenti} \end{cases}$$

x_i : dimensione in centinaia di mq dell'ipermercato localizzato in i , $\forall i \in I$;
 y_i : **variabile logica (binaria)** legata all'apertura di un ipermercato nella localizzazione i , $\forall i \in I$, cioè:
 $y_i = \begin{cases} 1 & \text{se viene aperto un ipermercato nella localizzazione } i \ (x_i > 0) \\ 0 & \text{altrimenti} \end{cases}$

x_i : dimensione in centinaia di mq dell'ipermercato localizzato in i , $\forall i \in I$

Funzione obiettivo:

$$\max \sum_{i \in I} R_i x_i y_i \quad \text{s.t.}$$

Ovviamente, tutto ciò che spendo deve essere $\leq W$, cioè del budget.

$$\sum_{i \in I} C_i x_i y_i + F_i y_i \leq W \quad (C_i \text{ è il costo per ogni 100 mq, } F_i \text{ è costo fisso del terreno, } R_i \text{ è il ricavo})$$

$$x_i \in \mathbb{R}_+, y_i \in \{0,1\}, \forall i \in I$$

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

Essa sarebbe la funzione indicatrice (step) ma, essendo non lineare (dato che usa $x_i y_i$ come termine, che sarebbe quadratico, non posso usarla).

Con $y_i = 1$ allora si paga il costo fisso ed è possibile guadagnare R_i in funzione obiettivo a seconda del valore di x_i .

Con $y_i = 0$ non si paga il costo fisso, non si guadagna R_i in funzione obiettivo o si paga il costo variabile per la localizzazione i .

Formulazione semanticamente corretta ma sbagliata: NON lineare!

Insiemi

I : possibili localizzazioni.

Variabili decisionali

x_i : dimensione in centinaia di mq dell'ipermercato localizzato in i , $\forall i \in I$;

y_i : **variabile logica (binaria)** legata all'apertura di un ipermercato nella localizzazione i , $\forall i \in I$, cioè:

$$y_i = \begin{cases} 1 & \text{se viene aperto un ipermercato nella localizzazione } i \ (x_i > 0) \\ 0 & \text{altrimenti} \end{cases}$$

Modello non lineare(!!!)

$$\begin{aligned} \max \quad & \sum_{i \in I} R_i x_i y_i \\ \text{s.t.} \quad & \sum_{i \in I} C_i x_i y_i + F_i y_i \leq W \quad (\text{budget}) \\ & x_i \in \mathbb{R}_+, y_i \in \{0,1\} \quad \forall i \in I \end{aligned}$$

I possibili valori per y_i sono 2: se $y_i = 1$, allora si paga il costo fisso ed è possibile guadagnare R_i in funzione obiettivo e pagare il costo variabile, a seconda del valore di x_i . Se $y_i = 0$ non si paga il costo fisso e non si guadagna R_i in funzione obiettivo né si paga il costo variabile per la localizzazione i (indipendentemente dal valore di x_i , che sarà da ignorare in questo caso). Pertanto il modello rappresenta bene l'insieme delle soluzioni ammissibili, **ma non è lineare**, per la presenza del termine $x_i y_i$ che moltiplica due variabili tra loro (y_i , anche se binaria, è una variabile!). Pertanto, il modello è sbagliato nel contesto dei modelli PL.

Per linearizzare il modello precedente, è necessario introdurre un vincolo esplicito che forzi $x_i = 0$ qualora $y_i = 0$ rendendo effettiva nel modello la descrizione data per le variabili y_i . Questo sarà introdotto dopo.

Per ora, ragioniamo con un termine solo:

$$\begin{aligned} \max \quad & \sum_{i \in I} R_i x_i \\ \text{s.t.} \quad & \\ & \sum_{i \in I} C_i x_i + F_i y_i \leq W \\ & x_i \in \mathbb{R}_+ \quad y_i \in \{0,1\} \end{aligned}$$

Ipotizzando una possibile soluzione, in cui aprono y_1 e y_2 (possono aprire entrambe una delle due non apre affatto). L'ottimizzatore, così come è stata scritto, non avrà nessun motivo per percorrere l'apertura ad 1. Non si sa infatti se y_i debba essere > 0 e contemporaneamente ci possa essere $x_i = 1$, poste le altre variabili del problema.

$$\begin{aligned} y_2 = 1 \quad x_2 &= 70 \\ y_2 = 0 \quad x_2 &= 70.001 \end{aligned}$$

Idealmente (a commento)

$$//y_1 \simeq x_1 \text{ (se } x_1 > 0, y_1 = 1)$$

Non si scriva una cosa del genere (simil linguaggio di programmazione, non è lineare)

$$\rightarrow \text{if } x_1 > 0 \text{ then } y_1 = 1$$

La formulazione operata fino ad ora non è lineare. Riscriviamo dunque il modello linearmente:

$$\begin{aligned} \max \quad & \sum_{i \in I} R_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} C_i x_i + F_i y_i \leq W \text{ (budget)} \end{aligned}$$

Idealmente dovremmo scrivere: $x_i \leq 0 \Leftrightarrow y_i = 0$

Invece, prendiamo y_i (che vale 1) e lo moltiplichiamo per una costante M sufficientemente grande (non si metta M tra le variabili):

$x_i \leq M * y_i \Rightarrow M$ è sufficientemente grande (ed y_i non deve mai superare la dimensione tale da rendere il vincolo irrilevante)

Quindi, il vincolo aggiunto è lineare anch'esso:

$$x_i \leq M y_i \quad \forall i \in I$$

Tale vincolo è noto con il nome di *Big-M* (big-M constraint). Questo serve a sancire nel modello la relazione logica che esiste tra x_i ed y_i , rendendo effettiva ("attivando") la variabile binaria y_i . Il completamento dei vincoli è come appare a lato:

$$\begin{aligned} \max \quad & \sum_{i \in I} R_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} C_i x_i + F_i y_i \leq W \quad \text{(budget)} \\ & x_i \leq M y_i \quad \forall i \in I \quad \text{(attivazione delle variabili binarie)} \\ & x_i \in \mathbb{R}_+, \quad y_i \in \{0,1\} \quad \forall i \in I \end{aligned}$$

Idealmente, esprimendo l'apertura in due posti diversi come tabella di verità, sarebbe come inserire la funzione *NAND* (questa formulazione, tabella sotto alla mano, viene risolta con il Big-M constraint):

Y_A	Y_B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Esprimo un vincolo logico attraverso l'espressione lineare (usando variabili intere vincolate ad assumere valori tra 0 ed 1, estremi compresi):

$$y_A + y_B \leq 1$$

Dominio:

$$x_i \in \mathbb{R}_+ \quad y_i \in \{0,1\}$$

Nella dispensa, esiste una variante dello stesso problema che però introduce vincoli aggiuntivi:

Una catena della Grande Distribuzione Organizzata (GDO) dispone di un budget W per l'apertura di nuovi ipermercati in Italia. Gli studi preliminari hanno individuato un insieme I di possibili localizzazioni. Per l'apertura di un ipermercato nella localizzazione i bisogna sostenere un costo fisso F_i (acquisto del terreno, oneri amministrativi etc.) e un costo variabile pari a C_i ogni 100 mq di ipermercato. Per ogni localizzazione, è prevista una dimensione massima U_i e, nel caso di apertura, una dimensione minima pari a L_i . Una volta aperto e entrato a regime, ciascun ipermercato localizzato in i produrrà entrate per R_i ogni 100 mq. Determinare l'insieme di localizzazioni in cui aprire dei nuovi ipermercati e dimensionare gli ipermercati stessi in modo da massimizzare i ricavi complessivi, tenendo conto che non possono essere aperti più di K ipermercati.

Insiemi

I : possibili localizzazioni.

Variabili decisionali

x_i : dimensione in centinaia di mq dell'ipermercato localizzato in i , $\forall i \in I$;

y_i : **variabile logica (binaria)** legata all'apertura di un ipermercato nella localizzazione i , $\forall i \in I$, cioè:

$$y_i = \begin{cases} 1 & \text{se viene aperto un ipermercato nella localizzazione } i \ (x_i > 0) \\ 0 & \text{altrimenti} \end{cases}$$

Modello PLIM

$$\begin{aligned} \max \quad & \sum_{i \in I} R_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} C_i x_i + F_i y_i \leq W && (\text{budget}) \\ & x_i \leq U_i y_i \quad \forall i \in I && (\text{dimensione massima / attivazione delle variabili binarie}) \\ & x_i \geq L_i y_i \quad \forall i \in I && (\text{dimensione minima dell'ipermercato}) \\ & \sum_{i \in I} y_i \leq K && (\text{numero massimo di ipermercati}) \\ & x_i \in \mathbb{R}_+, y_i \in \{0, 1\} \quad \forall i \in I \end{aligned}$$

Se scrivessimo modelli non lineari, saremmo costretti ad utilizzare metodi di programmazione *non* lineare.

Infatti, si deve considerare che le variabili decisionali possono essere soggette a vincoli di tipo logico, più o meno espliciti (che sono in correlazione tra di loro *a patto che si verifichino determinate condizioni che influenzano la decisione che devo prendere*).

Queste variabili hanno due valori possibili come dominio, quindi 0 oppure 1 (variabili binarie); queste sono combinate in modo lineare, secondo espressioni di relazione logica riportate in tabella, che mostra esempi corretti/esempi sbagliati.

SBAGLIATO			Corretto	
formulazioni <i>NON LINEARI</i> !!!			Vincoli	Domini
if $x_1 > 0$ then $x_2 = 0$ and if $x_2 > 0$ then $x_1 = 0$	$\text{nand}(y_1, y_2)$	$x_1 x_2 = 0$ $y_1 y_2 = 0$	$x_1 \leq M y_1$ $x_2 \leq M y_2$ $y_1 + y_2 \leq 1$	$x_1, x_2 \geq 0$ $y_1, y_2 \in \{0, 1\}$ $(M \rightarrow \infty)$
if $x > 0$ then $y = 1$	$(x > 0) \rightarrow (y = 1)$	$x(1 - y) = 0$	$x \leq M y$	$x \geq 0, y \in \{0, 1\}$
$y_1 = 1$ or $y_2 = 1$	$y_1 \vee y_2$	$(1 - y_1)(1 - y_2) = 0$	$y_1 + y_2 \geq 1$	$y_1, y_2 \in \{0, 1\}$
$y_1 = 1$ and $y_2 = 1$	$y_1 \wedge y_2$	$y_1 y_2 = 1$	$y_1 + y_2 = 2$	$y_1, y_2 \in \{0, 1\}$
$y_1 = 1$ only if $y_2 = 1$	$y_1 \rightarrow y_2$	$y_1(1 - y_2) = 0$	$y_1 \leq y_2$	$y_1, y_2 \in \{0, 1\}$
$y_1 = 1$ only if $y_2 = 0$	$y_1 \rightarrow \overline{y_2}$	$y_1 y_2 = 0$	$y_1 \leq (1 - y_2)$	$y_1, y_2 \in \{0, 1\}$
$y_1 = 1$ xor $y_2 = 1$	$y_1 \neq y_2$		$y_1 + y_2 = 1$	$y_1, y_2 \in \{0, 1\}$
etc. etc. etc.			etc. etc. etc.	
Queste formulazioni sono semanticamente corrette ma			Queste formulazioni sono	
NON ACCETTABILI			corrette a patto di	
in un modello di programmazione lineare			- SPECIFICARE I DOMINI	
			- ATTIVARE le var. logiche	

Attivare le variabili: relazionare le variabili facendo in modo di linearizzare il modello facendole valere solo 0 od 1 (f. di attivazione). Si usano quindi le variabili attivate ed utilizzo i vincoli $\in [0, 1]$

Altro esercizio, Portfolio Optimization (14), problema di mix ottimo di produzione

Un cliente affida ad un'agenzia finanziaria 100 000 euro da impiegare in fondi di investimento. I fondi attualmente offerti dal mercato sono  cinque , come riassunto in tabella:

Nome	Tipo	Moody's rating	Durata (anni)	Rendita alla maturazione
A	privato	Aa	9	4,5%
B	pubblico	A	15	5,4%
C	stato	Aaa	4	5,1%
D	stato	Baa	3	4,4%
E	privato	Ba	2	6,1%

Si sa che i fondi pubblici e dello stato sono tassati del 30% alla fine del periodo. Il cliente chiede di riservare almeno il 40% del capitale a fondi pubblici e dello stato e vuole che la durata media dell'investimento non superi i 5 anni.

Le regole del mercato impongono che l'investimento in C precluda la possibilità di investire in D, e viceversa. Inoltre, è possibile investire in E solo se si sono investiti almeno 10 000 euro in A. Infine, trasformando il Moody's rating in una scala numerica (Aaa = 1, Aa = 2, A = 3, Baa = 4 e Ba = 5), il valore medio del rischio dell'investimento non deve superare 1,5. Si vuole massimizzare la rendita finale dell'investimento.

Variabili decisionali:

x_i : € investiti in $i \in \{A, \dots, E\}$

Funzione obiettivo:

$$\max (0.45x_A + 0.54 * 0.7x_B + 0.7 * 0.51x_C + 0.7 * 0.44x_D + 0.61x_E) * 0.1$$

(sapendo che moltiplichiamo per 0.7 i fondi pubblici e dello stato dato che, per questi, ci sono le tasse del 30%, e grazie alla moltiplicazione per 0.7 stiamo togliendo in percentuale il 30%. Moltiplichiamo infine tutto per 0.1 perché esprimiamo tutto come percentuale)

s. t.

$$x_A + x_B + x_C + x_D + x_E \leq 100000 \text{ (non supero il budget per l'investimento)}$$

$$x_B + x_C + x_D \geq 0.4 * 100000 \text{ (si riserva il 40% del capitale per fondi pubblici e dello stato)}$$

$$\frac{x_A * 9 + x_B * 15 + x_C * 4 + x_D * 3 + x_E * 2}{x_A + x_B + x_C + x_D + x_E} \leq 5$$

(euro per anni + euro per anni)... / n. anni (durata media)

Scritta in questo modo, l'espressione sarebbe non lineare; scriviamo quindi la durata media come segue nella pagina dopo.

$$9x_A + 15x_B + 4x_C + 3x_D + 2x_E \leq 5(x_A + x_B + x_C + x_D + x_E) \text{ (durata media)}$$

$$2x_A + 3x_B + x_C + 4x_D + 5x_E \leq 1,5(x_A + x_B + x_C + x_D + x_E) \text{ (Moody's medio)}$$

(qui nel Moody's si vede a che rating corrisponde ciascun fondo e si moltiplica il fondo per la scala)

Introduciamo variabili binarie (dette anche logiche), fatto perché, se si investe in C, non si investe in D:

// $x_C \ x_D$

$x \in R_+$

// $y_i = 1$ se $x_i > 0$

// $y_i = 0$ altrimenti

$$y_C + y_D \leq 1 \text{ (dal testo si preclude infatti la possibilità che, investendo in C, allora si possa investire in D)}$$

Ora, attiviamo y :

$$x_C \leq My_C$$

$$x_D \leq My_D$$

Scritto da Gabriel

Ora consideriamo “è possibile investire in E solo se si sono investiti almeno 10000 euro in A”.
Se decidessi di investire in E, oppure in altre variabili, vengono aggiunti i relativi vincoli

// $y = 1$ se investissi in E
 // $y = 0$ altrimenti
 $x_A \geq 10000 y_E$ ~~if $y_E = 1$~~

// $y_E \simeq x_E$
 $x_E \leq M y_E$

Domini:

$x_i \in R_+$ $y_i \in \{0,1\}$

A conclusione di questo, diciamo che l'introduzione delle variabili logiche è necessariamente accompagnata ad una loro attivazione, anch'essa lineare. Infatti, la definizione di per sé non fa parte del modello. I vincoli di attivazione devono essere validi ed ammissibili (concretamente, significa che, se una variabile vale 1, un'altra varrà 0 e cose di questo tipo).

Laboratorio 1 - Utilizzo di solver per programmazione matematica

Come detto, un modello *dichiara* le caratteristiche della soluzione ottima in linguaggio matematico. Composto da insiemi, parametri, variabili decisionali e vincoli, con l'obiettivo di massimizzare o minimizzare una funzione obiettivo.

Le seguenti componenti sono usati:

- Un *solver* (o *risolutore*) è un software che riceve in input una descrizione di un problema di ottimizzazione e produce in output la soluzione ottima del modello e informazioni ad essa collegate.
- Un *modellatore* fornisce un'interfaccia verso un risolutore (cioè, un algoritmo). Hanno alcuni obiettivi:
 - o Disporre di un linguaggio semplice:
 - ad alto livello;
 - simile a quello di modellazione (linguaggio matematico);
 - formalmente strutturato;
 - possibilità di commenti.
 - o Consentire la separazione tra implementazione del modello e implementazione delle tecniche di soluzione
 - o Dialogare con diversi solver (strutture di I/O standard).
 - o Mantenere la separazione tra modello e dati del problema: per cambiare l'istanza basta cambiare i dati, non il modello.
 - o Linguaggio per script.

Ci sono varie configurazioni:

<u>Modellatori</u>	<u>Solver</u>
Foglio elettronico	Risolutore per fogli elettronici
AMPL	Cplex, Gurobi, Minos

Descrivono il modello sotto utilizzando le formule di un foglio elettronico. Alcune caratteristiche:

- Facilità di utilizzo (diffusione, interfaccia «familiare»)
- Integrazione con tool di presentazione
- Rigidità di utilizzo (no separazione modello/dati)
- Motori di ottimizzazione poco efficienti

Alcuni strumenti semplici:

- Risolutore (solver) integrato in Microsoft Excel
 - o <http://www.solver.com/excel-solver-help>
- Solver di LibreOffice
 - o <https://help.libreoffice.org/latest/it/text/scalc/01/solver.html?DbPAR=CALC>

Ora:

- Si apre Excel, schiacciando su File, Opzioni, Impostazioni Avanzate, Componenti Aggiuntivi e su Gestisci si clicca su “Vai” selezionano “Componenti aggiuntivi di Excel” e compare “Componente Aggiuntivo Risolutore”
- Si apre la scheda “Dati” e appare ora l’opzione “Risolutore”
- Nella scheda “Parametri Risolutore” si inseriscono le celle utili, nel tab “Soggette ai vincoli” si inseriscono con aggiungi uno per uno i vincoli e poi su “Selezionare un metodo di risoluzione” si sceglie “Simplex LP”

Riprendiamo il primo esempio visto a lezione (con annessa riproduzione in Excel):

Un coltivatore ha a disposizione 11 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg di semi di lattuga, 18 t di tuberi, 145 t di concime. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata per la coltivazione di lattuga è di 3000 €/ettaro e quella delle patate è di 5000 €/ettaro. L'assorbimento delle risorse per ogni tipo di coltivazione è di 7 kg di semi e 10 t di concime per ettaro di lattuga, e 3 t di tuberi e 20 di concime per le patate. Stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.

■ **Variabili decisionali:**

x_L : quantità in ettari da destinare a lattuga
 x_P : quantità in ettari da destinare a patate

■ **Funzione obiettivo:**

$$\max 3000 x_L + 5000 x_P$$

■ **Sistema dei vincoli:**

$x_L + x_P \leq 11$ (ettari disponibili)
 $7 x_L \leq 70$ (semi disponibili)
 $3 x_P \leq 18$ (tuberi disponibili)
 $10 x_L + 20 x_P \leq 145$ (concime disponibile)
 $x_L \geq 0, x_P \geq 0$ (dominio)

x_L	x_P		f.o.		
3000	5000		max		
7,5	3,5		40000		
1	1	11	<=	11 ettari	0,00
7	0	52,5	<=	70 semi	17,50
0	3	10,5	<=	18 tuberi	7,50
10	20	145	<=	145 concime	0,00

Ora, proviamo a spiegare un minimo come impostare dei modelli su Excel.

- Magliette e Borse (Young Money Makers) (testo e risultato)

Un gruppo di ragazzi vuole ricavare il più possibile vendendo agli amici magliette e borse decorate. Sono disponibili 10 magliette di cotone e 15 borse di tela e, per la decorazione, 32 riquadri disegnati e 40 profili rossi. Su ogni maglietta vengono apposti 6 riquadri e 2 profili, e su ogni borsa 3 riquadri e 5 profili. Sono anche disponibili 15 bottoni, e ogni borsa ne utilizzerà due per la chiusura. Sono state anche preparate 22 etichette, da apporre una su ogni maglietta e due su ogni borsa. Considerando che ogni maglietta decorata è venduta a 24 euro e ogni borsa a 16 euro, e che gli amici compreranno tutte le magliette e le borse, determinare la produzione che massimizza il ricavo.

xA	xB		f.o.		
24	16		max		
2	6		144		
1	2	14	<=	22	etichette
6	3	30	<=	32	riquadri
2	5	34	<=	40	bottoni
0	2	12	<=	15	profili
1	0	2	<=	10	magliette
0	1	6	<=	15	borse

In pratica:

- I vincoli del problema dicono di non dover superare 22 etichette, 10 magliette di cotone, 15 borse di tela, 32 riquadri disegnati e 15 profili.
- Per ognuno sono noti dei valori (es. etichette da apporre 1 su ogni maglietta e 2 su ogni borsa, oppure i riquadri, da apporre 6 su ogni maglietta e 3 su ogni borsa, ecc.)

Ognuno di questi valori per vincolo

(celle di sx) viene impostato per vincolo come minore/uguale del risultato a destra (es. celle che contengono 1 e 2 <= 22, celle che contengono 6 e 3 <= 32, ecc.), tutto secondo il modello.

Quindi, aprendo la scheda "Dati" e cliccando su "Risolutore", ciò viene impostato su "Aggiungi", dove si seleziona con il simbolo apposito le celle come da me indicato e si seleziona "<=" sui vincoli.

Ora, sempre su Risolutore, viene impostata la funzione obiettivo (questo, nella cella che nell'immagine si vede sotto a max, quindi nella scheda "Parametri Risolutore" e cliccando sul simbolo $\uparrow$ si seleziona quella cella; similmente, si imposti con lo stesso pulsante su "Modificando le celle variabili", tale da considerare le celle sotto i parametri della funzione obiettivo (per il problema in oggetto, 24 e 16).

Fatto ciò, deve essere impostata la colonna in grigio; in pratica, si veda il ragionamento ad esempio sulla prima cella:

SOMMA				✖ ✓ f_x		=A4*A\$3+B4*B\$3	
	A	B	C	D	E	F	
1	xA	xB		f.o.			
2	24	16		max			
3	2	6		144			
4	1	2	=A4*A\$3+	<=	22	etichette	

Vengono selezionate le celle variabili impostate poco fa e i singoli parametri per ogni vincolo, eseguendo la somma tra il prodotto della variabile della funzione obiettivo e quella del problema in oggetto.

Ciò viene fatta identicamente per ogni cella (trascinando col pulsante in basso a dx della cella in cui si ha scritto questa cosa fino all'ultima, quindi quella che di lato possiede 0 ed 1).

Ultima cosa da aggiungere (come Vincolo sulla scheda "Parametri Risolutore"); le variabili (celle con 2 e 6, che sono parametri che essendo variabili posso modificarmi a mano), devono essere intere.

Letteralmente, si vada su "Parametri Risolutore" e, premendo Aggiungi, si selezioni nella finestra che appare "int" al posto di <= questa volta, e si è a posto.

Fatti tutti questi passaggi, è possibile premere senza problemi "Risolvi", risultando tutto esattamente come mostrato sopra.

Le slide di questa lezione fanno riferimento anche a due modelli di cui si presenta testo e soluzione (volendo, da fare secondo quanto presente dal testo e adattato in modo simile a quanto ben spiegato da me; consiglio quello della Dieta piuttosto che quello del Trasporto, presenti comunque entrambi su Mega):

- Problema della dieta

Un dietologo deve preparare una dieta che garantisca un apporto giornaliero di proteine, ferro e calcio di almeno 20 mg, 30 mg e 10 mg, rispettivamente. Il dietologo è orientato su cibi a base di verdura (5 mg/kg di proteine, 6 mg/kg di ferro e 5 mg/kg di calcio, al costo di 4 €/kg), carne (15 mg/kg di proteine, 10 mg/kg di ferro e 3 mg/kg di calcio, al costo di 10 €/kg) e frutta (4 mg/kg di proteine, 5 mg/kg di ferro e 12 mg/kg di calcio, al costo di 7 €/kg). Determinare la dieta di costo minimo.

xV	xC	xF		f.o			
4	10	7		min	min $4x_1 + 10x_2 + 7x_3$		
3	1,2	0		24			
5	15	4	33	>=	20	proteine	$5x_1 + 15x_2 + 4x_3 \geq 20$
6	10	5	30	>=	30	ferro	$6x_1 + 10x_2 + 5x_3 \geq 30$
5	3	12	18,6	>=	10	calcio	$5x_1 + 3x_2 + 12x_3 \geq 10$
1	0	0	3	<	3	max verdure	
0	1	0	1,2	<	3	max carne	
0	0	1	0	<	3	max frutta	

- Problema del trasporto di frigoriferi

Una ditta di produzione di elettrodomestici produce dei frigoriferi in tre stabilimenti e li smista in quattro magazzini intermedi di vendita. La produzione settimanale nei tre stabilimenti A, B e C è rispettivamente di 50, 70 e 20 unità. La quantità richiesta dai 4 magazzini è rispettivamente di 10, 60, 30 e 40 unità. I costi per il trasporto di un frigorifero tra gli stabilimenti e i magazzini 1, 2, 3 e 4 sono i seguenti:

- dallo stabilimento A: 6, 8, 3, 4 euro;
- dallo stabilimento B: 2, 3, 1, 3 euro;
- dallo stabilimento C: 2, 4, 6, 5 euro.

La ditta vuole determinare il piano di trasporti di costo minimo.

Trasporto di frigoriferi					
Costi unitari		Destinazioni			
		1	2	3	4
Origini	A	6	8	3	4
	B	2	3	1	3
	C	2	4	6	5
Come la funzione obiettivo min $6x_{A1} + 8x_{A2} + 3x_{A3} + 4x_{A4} + 2x_{B1} + 3x_{B2} + 1x_{B3} + 3x_{B4} + 2x_{C1} + 4x_{C2} + 6x_{C3} + 5x_{C4}$					
Quantità trasportata		Destinazioni			
		1	2	3	4
Origini	A	0	0	10	40
	B	0	50	20	0
	C	10	10	0	0
	Tot	10	60	30	40
		ug	ug	ug	
Domanda		10	60	30	40
Costo complessivo		420			

Programmazione Lineare e Metodo del Simplex: Vertici (e vertice ottimo), poliedri, forma standard

Un *modello di programmazione matematica* è un oggetto che rappresenta un problema di ottimizzazione, in cui abbiamo una funzione obiettivo f che può essere minimizzata/massimizzata con variabili reali rappresentanti un certo insieme di vincoli definiti come funzioni. In generale:

- si considerano variabili intere (discrete) $\rightarrow$ programmazione lineare
- si considerano variabili continue $\rightarrow$ metodo del simplex

In termini matematici:

$$\begin{aligned} \min(\max) \quad & f(x) \\ \text{s.t.} \quad & g_i(x) = b_i \quad (i = 1 \dots k) \\ & g_i(x) \leq b_i \quad (i = k + 1 \dots k') \\ & g_i(x) \geq b_i \quad (i = k' + 1 \dots m) \\ & x \in \mathbb{R}^n \end{aligned}$$

dove

- $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ è un vettore di n variabili reali (ciascun vettore rappresenta una potenziale soluzione del problema);
- f e g_i sono funzioni $\mathbb{R}^n \rightarrow \mathbb{R}$
- $b_i \in \mathbb{R}$

Un problema di Programmazione Lineare (PL) è un problema di ottimizzazione in cui la funzione obiettivo f e tutti i vincoli g_i sono funzioni lineari delle variabili x_j (in questa fase consideriamo solo variabili reali; quanto diremo non vale in caso di variabili intere/binarie):

$$\begin{aligned} \min(\max) \quad & c_1x_1 + c_2x_2 + \dots + c_nx_n \\ \text{s.t.} \quad & a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \quad (i = 1 \dots k) \\ & a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \leq b_i \quad (i = k + 1 \dots k') \\ & a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geq b_i \quad (i = k' + 1 \dots m) \\ & x_i \in \mathbb{R} \quad (i = 1 \dots n) \end{aligned}$$

La soluzione di un problema PL è data da:

- una *soluzione ammissibile*: $x \in \mathbb{R}^n$ che soddisfa tutti i vincoli
- una *regione ammissibile*: insieme delle x ammissibili
- *soluzione ottima* x^* [*min*]: $c^T x^* \leq c^T x, \forall x \in \mathbb{R}^n, x$ ammissibile.

Risolvere un problema PL significa riconoscere uno dei seguenti tre casi (non sempre avremo una soluzione ammissibile):

- è *inammissibile* (l'insieme ammissibile è vuoto)
- è *illimitato* (ammette ogni possibile soluzione che fa aumentare/minimizzare la f. obiettivo, quindi insieme delle x ammissibili)
- *ammette soluzione ottima* (con valore finito all'interno della regione ammissibile che ottimizza la funzione obiettivo; al massimo può esistere, in alcuni casi strani, un *sup* tale da poter approssimare la soluzione "alla meglio")

La regione ammissibile, a livello geometrico, è un *poliedro* (intersezione di un numero *finito* di semispazi [ciascuna delle due parti in cui uno spazio resta diviso da un piano in esso contenuto] chiusi ed iperpiani [sottospazio lineare di dimensioni $n - 1$ rispetto allo spazio che lo contiene] in $\mathbb{R}^n$, i quali sono regioni *convesse* (tutto ciò che non "ha buchi").

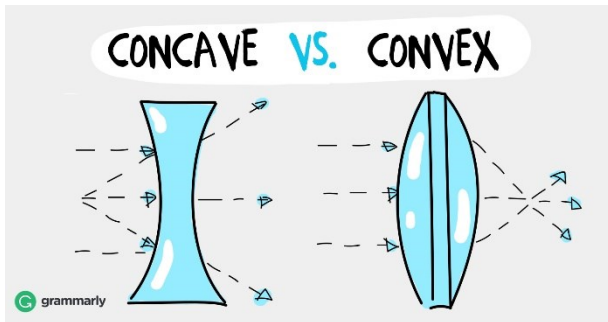


Figura 1: Differenza concavo/convesso visivamente

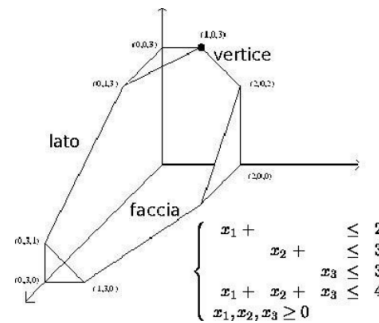


Figure 1: Un poliedro in $\mathbb{R}^3$

Problema di PL: $\min(\max)\{c^T x : x \in P\}$, P è un poliedro in $\mathbb{R}^n$.

Vertici di un poliedro: definizione

- $z \in \mathbb{R}^n$ è *combinazione convessa* di due punti $x, y \in \mathbb{R}^n$, se $\exists \lambda \in [0,1]: z = \lambda x + (1 - \lambda)y$ (z è esprimibile come *somma pesata* di questi due punti, sapendo che i pesi non sono compresi tra 0 ed 1 ma la loro somma fa 1)
- $z \in \mathbb{R}^n$ è *combinazione convessa stretta* di due punti $x, y \in \mathbb{R}^n$, se $\exists \lambda \in (0,1): z = \lambda x + (1 - \lambda)y$
- $v \in P$ è *vertice del poliedro* P se non può essere espresso come *combinazione convessa stretta* di due punti distinti dello stesso poliedro $\nexists x, y \in P, \lambda \in (0,1): x \neq y, v = \lambda x + (1 - \lambda)y$

Nello specifico:

- una *combinazione convessa* è il punto $z \in \mathbb{R}^n$ che per un insieme di punti $x^1, x^2, \dots, x^k \in \mathbb{R}^n$, si sa che esistono k valori scalari non negativi $\lambda_1, \lambda_2, \dots, \lambda_k$ tale che la somma di tutti i $\lambda_i = 1$.

Rappresentazione dei poliedri

$z \in \mathbb{R}^n$ è *combinazione convessa* di $x^1, x^2, \dots, x^k$ se

$\exists \lambda_1, \lambda_2, \dots, \lambda_k \geq 0$:

$$\sum_{i=1}^k \lambda_i = 1 \text{ e } z = \sum_{i=1}^k \lambda_i x^i$$

(ciò significa che esprimo una regione poliedro vicina ai punti di partenza, rappresentando tutti i punti nel quadrilatero presente)

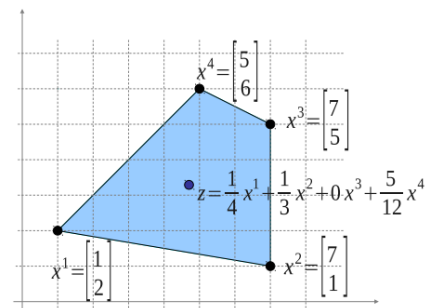


Figure 3: Combinazione convessa in $\mathbb{R}^2$

Il *teorema di rappresentazione dei poliedri Minkowski-Weyl* enuncia che un poliedro è dato da una combinazione convessa limitata dei suoi vertici e, a sua volta, un poliedro (conoscendo i vertici) è insieme convesso.

Teorema 1 (Rappresentazione dei poliedri [Minkowski-Weyl] - caso limitato): Dato un poliedro limitato $P \subseteq \mathbb{R}^n$, e indicando con $v^1, v^2, \dots, v^k$ ($v^i \in \mathbb{R}^n$) i vertici di P , si ha $x \in P \Rightarrow x = \sum_{i=1}^k \lambda_i v^i$ con $\lambda_i \geq 0, \forall i = 1..k$ e $\sum_{i=1}^k \lambda_i = 1$. In altri termini, **ogni punto di P si può ottenere come combinazione convessa dei suoi vertici.**

“Se esiste una soluzione ottima, esiste almeno un vertice ottimo”; intuitivamente, conoscendo i vertici, cerco la soluzione nella regione ammissibile guardando solo quelli.

Teorema: esistenza di un vertice ottimo (versione “min”)

(in altri termini, come trovare la soluzione ottima “senza fare disegni”, visto che non sempre le dimensioni dei piani di applicazione lo permettono)

Problema PL $\min\{c^T x : x \in P\}$, P non vuoto e limitato:

- PL ammette soluzione ottima
- esiste almeno un vertice ottimo

Dimostrazione (discorsiva):

Considero un insieme di vertici e calcolo l'insieme $c^T v_1, c^T v_2, \dots$ fino a trovare la soluzione ottima

(vertice migliore), vista come $v^* = \operatorname{argmin}_{v \in V} c^T v^k$

(*argmin* è l'argomento del minimo, quindi in generale è l'insieme dei valori in cui la funzione raggiunge il minimo; a noi interessa il vertice ottimo).

Spiegata a parole: considerata una somma di vertici vicini allo spazio di definizione, sapendo che lo spazio è convesso, per ogni vertice del poliedro, il vertice v^* è minimo del problema (ed è ottimo).

$$c^T x = c^T \sum_{i=1}^k \lambda_i v^i = \sum_{i=1}^k \lambda_i c^T v^i \geq \sum_{i=1}^k \lambda_i c^T v^* = c^T v^* \sum_{i=1}^k \lambda_i = c^T v^*$$

In sintesi, $\forall x \in P, c^T v^* \leq c^T x$, cioè v^* è una soluzione ottima corrispondente ad un vertice del poliedro. ■

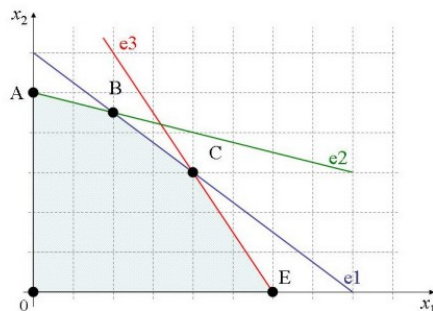
Possiamo limitare la ricerca dell'ottimo ai “soli” vertici.

Vediamo un esempio molto easy di PL (con f. obiettivo e vincoli a lato)

Esempio 1 Una piccola ditta di profumi realizza due nuove fragranze a partire da 3 essenze: rosa, mugheretto e viola. Per realizzare un litro di fragranza 1 sono richiesti 3 centilitri di rosa, 1 centilitro di mugheretto e 3 centilitri di viola. Per realizzare un litro di fragranza 2 sono richiesti 4 centilitri di rosa, 4 centilitri di mugheretto e 2 centilitri di viola. La disponibilità in magazzino per le tre essenze è di 24, 20 e 18 centilitri per rosa, mugheretto e viola rispettivamente. Sapendo che l'azienda realizza un profitto di 13 e 10 euro per ogni litro venduto di fragranza 1 e 2 rispettivamente, determinare le quantità ottimali delle due fragranze da produrre.

$$\begin{array}{llll} \max & 13x_1 & + & 10x_2 \\ \text{s.t.} & 3x_1 & + & 4x_2 \leq 24 \quad (e1) \\ & x_1 & + & 4x_2 \leq 20 \quad (e2) \\ & 3x_1 & + & 2x_2 \leq 18 \quad (e3) \\ & x_1 & , & x_2 \geq 0 \end{array}$$

Esempio: vertici come intersezione



$$\begin{array}{llll} \max & 13x_1 & + & 10x_2 \\ \text{s.t.} & 3x_1 & + & 4x_2 \leq 24 \quad (e1) \\ & x_1 & + & 4x_2 \leq 20 \quad (e2) \\ & 3x_1 & + & 2x_2 \leq 18 \quad (e3) \\ & x_1 & , & x_2 \geq 0 \end{array}$$

$$B = e_1 \cap e_2 \quad (2,9/2) \quad 71$$

$$C = e_1 \cap e_3 \quad (4,3) \quad 82$$

$$E = e_3 \cap (x_2 = 0) \quad (6,0) \quad 78$$

C ottimo (dato dall'intersezione di due rette, ricavabile dall'intersezione di iperpiani)

La ricerca dell'ottimo può avvenire sui soli vertici e ciascun vertice è ottenibile dall'intersezione di due delle rette che definiscono i semipiani corrispondenti ai vincoli (sulle disponibilità di esistenze o di non negatività delle variabili).

Ad esempio, il vertice $B = (2,9/2)$ è ottenuto dall'intersezione della retta $x_1 + 4x_2 = 20$ con la retta $3x_1 + 4x_2 = 24$; il vertice $E = (6,0)$ è ottenuto dall'intersezione delle rette $x_2 = 0$ e $3x_1 + 2x_2 = 18$ etc. Calcolando il valore della funzione obiettivo in ciascun vertice e scegliendo il valore massimo, si ottiene l'ottimo in corrispondenza del vertice C: $x_1 = 4, x_2 = 3$ con valore della funzione obiettivo pari a 82 (profitto massimo pari a 82). Ovviamente, la rappresentazione grafica dei vertici non è sempre possibile e, per poter procedere alla ricerca dell'ottimo sui vertici, è necessario poterli determinare per via algebrica.

Caratterizzazione algebrica dei vertici

Rappresentiamo in modo equivalente le soluzioni che ci interessano come *equazioni* (tralasciando per il momento i vincoli di non negatività delle variabili)

$$\begin{array}{rclclcl} 3x_1 & + & 4x_2 & + & s_1 & & = & 24 \\ x_1 & + & 4x_2 & & & + & s_2 & = & 20 \\ 3x_1 & + & 2x_2 & & & & + & s_3 & = & 18 \end{array}$$

dove s_1, s_2, s_3 sono variabili ausiliarie che indicano la possibilità di soddisfare i rispettivi vincoli originari all'uguaglianza (se assumono valore 0) o in modo stretto (se assumono valori > 0).

Si può risolvere il sistema fissando il valore a 0 in maniera opportuna (entro la regione ammissibile per non violare i vincoli), ottenendo sulla base di 3 equazioni e 5 incognite, $+\infty^{5-2=3}$ soluzioni.

Si immagina la risoluzione di questi sistemi con il *metodo di Gauss-Jordan* per la soluzione di sistemi $Ax = b$: eseguire delle operazioni elementari sulla matrice aumentata $[A|b]$ in modo da ottenere in A una sottomatrice identità di dimensioni pari a $\rho(A) = \rho(A|b)$

("rho" $\rightarrow \rho$, cioè il *raggio spettrale*, estremo superiore della norma che viene usato per capire se un metodo iterativo è convergente verso la soluzione di un problema)

(Una matrice aumentata è una matrice ottenuta aggiungendo le colonne di due matrici date, di solito allo scopo di eseguire le stesse operazioni elementari di riga su ciascuna delle matrici date).

L'*eliminazione di Gauss-Jordan* è un algoritmo che può essere utilizzato per risolvere sistemi di equazioni lineari e per trovare l'inversa di qualsiasi matrice invertibile. Si basa su tre operazioni elementari:

- scambiare le posizioni di due righe (riga i con la riga j)
- moltiplicare la riga i per uno scalare non nullo.
- sostituire alla riga i , la riga i più α volte la riga j ($\alpha \in R$)

Le operazioni elementari sulla matrice aumentata $[A|b]$ non alterano l'insieme delle soluzioni ammissibili del sistema $Ax = b$. Nel sistema, intendendo come soluzioni le s_i , si hanno:

$$\begin{aligned} s_1 &= 24 - 3x_1 - 4x_2 \\ s_2 &= 20 - x_1 - 4x_2 \\ s_3 &= 18 \end{aligned}$$

Si basa sul *teorema di Rouché-Capelli*, dunque:

$Ax = b$ ammette soluzioni $\Leftrightarrow \rho(A) = \rho(A|b) = r(\infty^{(n-r)} \text{ soluzioni})$

(perché cerchiamo di diagonalizzare la matrice rispetto al rango r ed $(n - r)$ sono i gradi di libertà, fissando in arbitrario il valore di $(n - r)$ variabili).

Nel sistema di prima:

$$\begin{array}{rclclcl} 3x_1 & + & 4x_2 & + & s_1 & & = & 24 \\ x_1 & + & 4x_2 & & & + & s_2 & = & 20 \\ 3x_1 & + & 2x_2 & & & & + & s_3 & = & 18 \end{array}$$

Si ottiene quindi (fissando $s_1, s_2 = 0$)

$$\begin{array}{rclclcl} 3x_1 & + & 4x_2 & & & & = & 24 \\ x_1 & + & 4x_2 & & & & = & 20 \\ 3x_1 & + & 2x_2 & + & s_3 & = & 18 \end{array}$$

Rango: $r = 3$

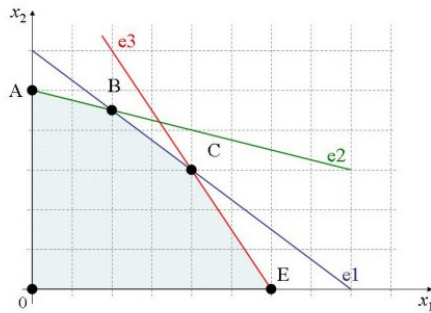
Numero di incognite: $n = 5$

Soluzioni: $\infty^{(n-2)}$

Gradi di libertà: $2 \rightarrow$ (numero di variabili indipendenti, *from my understanding*)

Prendo il sistema e cerco le soluzioni mettendo a zero due variabili "a caso", per esempio s_2 ed s_3 .

Prima faccio il conto, poi vado sul piano. Si ottiene quindi:



$$s_1 = s_2 = 0 \quad (2, 9/2, 0, 0, 3) \quad B$$

$$x_1 = s_2 = 0 \quad (0, 5, 4, 0, 8) \quad A$$

...

$$s_2 = s_3 = 0 \quad (3.2, 4.2, -2.4, 0, 0)$$

non ammissibile!

Mettendo a zero, le disuguaglianze diventano uguaglianze, tale da ricavarsi l'intersezione. Essendo un sistema quadrato, metterò esattamente due variabili a zero, ottenendo un punto di intersezione.

In questo modo, voglio l'intersezione tra la retta 2 e la retta 3. Il vincolo viene violato con -2.4 in quanto s_1 andrebbe scelto negativo. Soluzioni: $x_1 = 2, x_2 = 9/2, s_1 = 0, s_2 = 0, s_3 = 3$

La soluzione corrisponde al vertice B . In effetti, porre $s_1 = s_2 = 0$ significa, da un punto di vista geometrico, saturare i vincoli ($e1$) ed ($e2$): la soluzione si troverà quindi all'intersezione delle corrispondenti rette. Un'altra soluzione può essere ottenuta fissando a 0 le variabili x_1 e s_2 , che porta alla soluzione $x_1 = 0, x_2 = 5, s_1 = 4, s_2 = 0, s_3 = 8$, corrispondente al vertice A .

Si noti che le variabili da porre a 0 devono essere opportunamente scelte. Ad esempio, ponendo $x_1 = s_1 = 0$, si ottiene la soluzione $x_1 = 0, x_2 = 6, s_1 = 0, s_2 = -4, s_3 = 6$ che non corrisponde ad un vertice del poliedro: la soluzione ottenuta non è infatti ammissibile, dato che $s_2 < 0$ indica che il vincolo ($e2$) è violato.

Dato un problema PL qualsiasi, esso può essere messo nella seguente forma, detta forma standard:

$$\min \quad c_1x_1 + c_2x_2 + \dots + c_nx_n$$

$$\text{s.t.} \quad a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \quad (i = 1 \dots m)$$

$$x_i \in \mathbb{R}_+ \quad (i = 1 \dots n)$$

- vincoli sono delle equazioni (+/- variabili slack/surplus, cioè rispettivamente variabili aggiunte/sottratte ad una disuguaglianza per trasformarla in uguaglianza)
- variabili ≥ 0 (sostituzione di variabili)
- funzione obiettivo di minimo senza costi additivi e moltiplicativi ($X - 1$)
- $b_i \geq 0$ ($X - 1$)

Tutto questo sapendo che:

- la funzione obiettivo è di minimo e senza costanti additive o moltiplicative (si moltiplicano per -1 le funzioni di massimizzazione; le costanti additive e moltiplicative positive possono essere trascurate, le costanti moltiplicative negative possono essere eliminate cambiando il verso di ottimizzazione);
- tutte le variabili sono positive o nulle (si effettuano sostituzioni di variabili per le variabili libere o negative);
- tutti i vincoli sono delle equazioni (si aggiunge una variabile positiva di slack per i vincoli di $\leq$ e si sottrae una variabile positiva di surplus per i vincoli di $\geq$);
- i termini noti b_i sono tutti positivi o nulli (si moltiplicano per -1 i vincoli con termine noto < 0).

Programmazione Lineare e Metodo del Simplexso

Esercizio 1 Mettere in forma standard il seguente problema di PL:

$$\begin{aligned} \max \quad & 5(-3x_1 + 5x_2 - 7x_3) + 34 \\ \text{s.t.} \quad & -2x_1 + 7x_2 + 6x_3 - 2x_1 \leq 5 \\ & -3x_1 + x_3 + 12 \geq 13 \\ & x_1 + x_2 \leq -2 \\ & x_1 \leq 0 \\ & x_2 \geq 0 \end{aligned}$$

Definiamo la funzione obiettivo z_{min} :

$$z_{min} = +3x_1 - 5x_2 + 7x_3$$

s. t.

$$-4x_1 + 7x_2 + 6x_3 \leq 5 \text{ (sto sommando } -2x_1 \text{ e } -2x_1) \Rightarrow -4x_1 + 7x_2 + 6x_3 + s_1 = 5$$

(Noi vogliamo l'uguaglianza; usiamo una variabile di slack [riempimento] per renderla tale. Ciò viene fatto per entrambe le equazioni in cui vengono inserite s_1, s_2)

$$-3x_1 + x_3 \geq 1 \Rightarrow -3x_1 + x_3 - s_2 = 1 \text{ (13 - 12 fa -1 e introduco } s_2)$$

$$x_1 + x_2 \leq -2 \Rightarrow x_1 + x_2 + s_3 = -2$$

$$x_1 \leq 0, x_2 \geq 0, x_3 \text{ libera}, s_1, s_2, s_3 \geq 0$$

Abbiamo x_1 negativa; introduciamo delle variabili cappello tale da non avere le variabili negative. Ciò viene fatto per la variabile x_1 che è negativa ma anche per x_3 che è variabile libera.

$$\hat{x}_1 = -x_1 \quad \hat{x}_1 \geq 0$$

$$x_3 = x'_3 - x''_3 \quad x'_3 \geq 0 \quad x''_3 \geq 0$$

A questo punto, operiamo per sostituzione (partendo direttamente da z_{min} , quindi non si consideri 5 o 34):

$$\min \quad -3\hat{x}_1 - 5x_2 + 7x'_3 - 7x''_3$$

$$\text{s.t.} \quad 4\hat{x}_1 + 7x_2 + 6x'_3 - 6x''_3 + s_1 = 5$$

$$3\hat{x}_1 + x'_3 - x''_3 - s_2 = 1$$

$$\hat{x}_1 - x_2 - s_3 = 2$$

$$\hat{x}_1 \geq 0, x_2 \geq 0, x'_3 \geq 0, x''_3 \geq 0, s_1 \geq 0, s_2 \geq 0, s_3 \geq 0.$$

Per non avere termini noti negativi, allora introduciamo delle variabili positive per raggiungere quello che vogliamo, scrivendolo in forma standard (attenzione ad andare con calma nei passaggi)

Similmente (da s. t. in poi), sostituiamo le variabili cappello a $-4x_1 + 7x_2 + 6x_3 \leq 5$, $-3x_1 + x_3 - s_2 = 1$ e $x_1 + x_2 + s_3 = -2$ (in quest'ultima, moltiplichiamo tutto per -1 tale da non avere termini noti negativi, diventando come si vede), definendo infine i domini.

Per alcuni richiami teorici

[Richiami di algebra lineare $\rightarrow$ cfr. pag. 12/46 "Note su Programmazione Lineare e Metodo del Simplexso"]

Come detto, partiamo da *soluzioni di base per sistemi di equazioni lineari* (in cui si mettono a zero alcune variabili [fuori base], scegliendo analogamente le variabili da *non* mettere a zero [in base]; assumiamo inoltre che avremo un sistema *sottodeterminato* [più colonne n che righe m] e che la matrice abbia rango massimo).

- Assunzioni: sistema $Ax = b, A \in R^{(m \times n)}, \rho(A) = m, m < n$
- *base* di A : sottomatrice quadrata di rango massimo, $B \in R^{(m \times n)}$ ottenuta scegliendo m colonne linearmente indipendenti della matrice A .
- Dato un sistema $Ax = b$ si scelga una base B della matrice A . Le colonne della matrice A e le variabili del vettore x possono essere riordinati opportunamente in modo da poter scrivere:

$$A = [B|F] \quad B \in \mathbb{R}^{m \times m}, \det(B) \neq 0 \quad x = \begin{bmatrix} x_B \\ x_F \end{bmatrix}, x_B \in \mathbb{R}^m, x_F \in \mathbb{R}^{n-m}$$

dove

- B è l'insieme delle colonne di A che formano la base;
- F l'insieme delle restanti colonne;
- x_B il vettore delle variabili corrispondenti alle colonne in base (*variabili di base*);
- x_F il vettore delle variabili corrispondenti alle colonne fuori base (*variabili non di base o fuori base*).

Di conseguenza, il sistema $Ax = b$ si può scrivere in forma a blocchi:

$$Ax = b \implies [B|F] \begin{bmatrix} x_B \\ x_F \end{bmatrix} = Bx_B + Fx_F = b$$

Osservando che la matrice di base B è invertibile (ha rango massimo) possiamo premoltiplicare per B^{-1} ottenendo:

$$x_B = B^{-1}b - B^{-1}Fx_F$$

Scegliendo una matrice di base B' diversa da B , cioè scegliendo un diverso insieme di m colonne di A linearmente indipendenti, si ottiene una nuova soluzione del sistema:

$$\text{con } x_F = 0, \text{ una soluzione di base: } x = \begin{bmatrix} x_B \\ x_F \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$$

Dato un sistema di equazioni $Ax = b$, le soluzioni ottenute scegliendo una base B della matrice e ponendo $x_B = B^{-1}b$ e $x_F = 0$ si dicono *soluzioni di base*. Altre soluzioni di base si ottengono scegliendo una *diversa* base di A .

Le variabili a 0 sono $n - m$ e si hanno soluzioni di base *degeneri* (quindi, se il vettore $x_B = B^{-1}b$ ha almeno una componente nulla).

Formalmente, una soluzione di base è detta *ammissibile* se viene soddisfatta la condizione di non negatività $x_B = B^{-1}b \geq 0$.

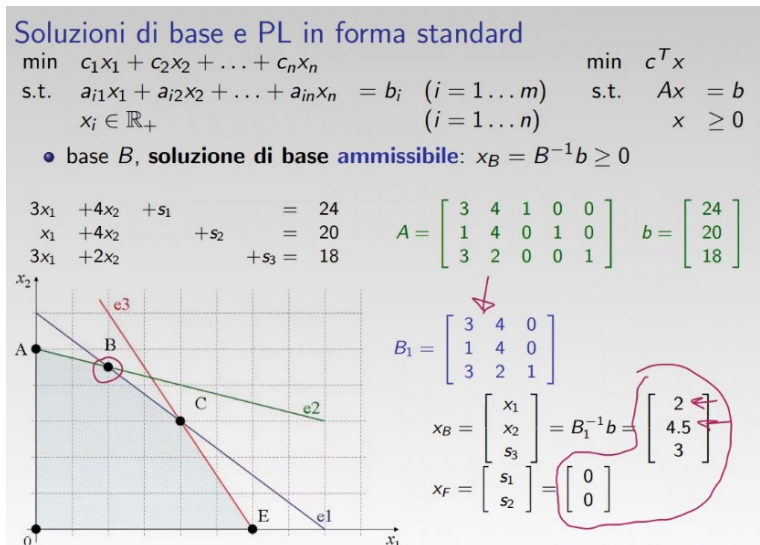
Consideriamo un problema di PL in forma standard (forma estesa e forma compatta, sx e dx):

$$\begin{array}{ll} \min & c_1x_1 + c_2x_2 + \dots + c_nx_n \\ \text{s.t.} & a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \quad (i = 1 \dots m) \\ & x_i \in \mathbb{R}_+ \quad (i = 1 \dots n) \end{array} \quad \begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}$$

oppure:

$$\min \{c^T x : Ax = b, x \geq 0\}$$

Troviamo la soluzione di base, prendendo delle colonne linearmente indipendenti (quindi, tali da metterle a zero, variabili del tipo s_i per dire); essa, *scegliendo bene*, corrisponde ad un vertice (per esempio, alcune soluzioni sono negative e dunque non ammissibili).



Nello schema sopra, scegliamo le colonne linearmente indipendenti (cioè, fissate a zero), scegliendo di volta in volta le basi e, operando bene, si trovano i vertici del poliedro e vertice ottimo:

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_3 \end{bmatrix} = B_1^{-1}b = \begin{bmatrix} 2 \\ 4.5 \\ 3 \end{bmatrix}$$

$$x_F = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^T = (2 \ 9/2 \ 0 \ 0 \ 3) \rightarrow \text{vertice B}$$

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_2 \end{bmatrix} = B_2^{-1}b = \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}$$

$$x_F = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^T = (4 \ 3 \ 0 \ 2 \ 0) \rightarrow \text{vertice C}$$

$$x_B = \begin{bmatrix} x_1 \\ s_1 \\ s_2 \end{bmatrix} = B_3^{-1}b = \begin{bmatrix} 6 \\ 6 \\ 14 \end{bmatrix}$$

$$x_F = \begin{bmatrix} x_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^T = (6 \ 0 \ 6 \ 14 \ 0) \rightarrow \text{vertice E}$$

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_1 \end{bmatrix} = B_4^{-1}b = \begin{bmatrix} 18/5 \\ 21/5 \\ -18/5 \end{bmatrix}$$

$$x_F = \begin{bmatrix} s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^T = (18/5 \ 21/5 \ -18/5 \ 0 \ 0) \rightarrow \text{n.a.}$$

Qua a dx abbiamo una soluzione non ammissibile, in quanto negativa.

Soluzioni di base ammissibili $\rightarrow n - m$ variabili a 0 $\rightarrow$

intersezione di un numero opportuno di iperpiani $\rightarrow$ vertici

$$PL \min \{c^T x : Ax = b, x \geq 0\} \quad P = \{x \in R^n : Ax = b, x \geq 0\}$$

Teorema: corrispondenza tra vertici e soluzioni ammissibili di base
(caratterizzazione algebrica dei vertici di un politopo)

x soluzione ammissibile di base di $Ax = b \iff x$ vertice di P

Corollario: soluzione ammissibile di base ottima

Se P non vuoto e limitato, allora esiste almeno una soluzione ottima coincidente con una soluzione ammissibile di base

Politopo:

In matematica, figura (graficamente non rappresentabile) di uno spazio euclideo a più di tre dimensioni (iperspazio), analoga di un poligono nel piano e di un poliedro nello spazio tridimensionale.

Algoritmo per PL (caso limitato): schema di principio (esaustivo/bruteforce)

- 1) metti il problema in forma standard $\min\{c^T x : Ax = b, x \geq 0\}$
- 2) $incumbent = +\infty$
- 3) repeat
 - a. genera una combinazione di m colonne di A
 - b. sia B la relativa sottomatrice di A
 - c. if $\det(B) \neq 0$ then calcola $x_B = B^{-1}b$ else continue
 - d. if $x_B \geq 0$ and $c_B^T x_B + c_F^T 0 < incumbent$ then aggiorna la soluzione
- 4) until (combinazioni di colonne esaurite)

(Incumbent solution $\rightarrow$ La migliore soluzione attuale trovata durante una procedura di ricerca algoritmica).

Complessità fino a:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \quad (\text{numero massimo di soluzioni ammissibili di base e di vertici del poliedro ammissibile}) \quad (\text{decisamente non efficiente})$$

$\Rightarrow$ metodo del simplesso: è un metodo iterativo che permette di esplorare in modo guidato, mediamente "più efficiente", l'insieme delle soluzioni ammissibili di base, a partire da una soluzione ammissibile di base data, e tali che siano miglioranti, quindi appunto migliorino (o, se andasse male, non peggiorino) la soluzione all'iterazione precedente, in termini della f.o.

Vediamo l'esempio del problema dei profumi espresso in *forma standard*:

$$\begin{aligned} \min \quad & z = -13x_1 - 10x_2 \\ \text{s.t.} \quad & 3x_1 + 4x_2 + s_1 = 24 \\ & x_1 + 4x_2 + s_2 = 20 \\ & 3x_1 + 2x_2 + s_3 = 18 \\ & x_1, x_2, s_1, s_2, s_3 \geq 0 \end{aligned}$$

una **soluzione ammissibile di base** iniziale (vertice B):

- $B = \begin{bmatrix} 3 & 4 & 0 \\ 1 & 4 & 0 \\ 3 & 2 & 1 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$
- $x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 9/2 \\ 3 \end{bmatrix} \quad x_F = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- $z_B = c^T x = c_B^T x_B + c_F^T x_F = -71$

Spiegando un po' le cose:

- La matrice B (base) considera i coefficienti per colonna delle singole equazioni (con l'ultima colonna che considera 1 in ultima riga in quanto corrisponderebbe ad una soluzione ammissibile di base data dal problema, cioè s_3)
- La matrice F (fuori base) considera le colonne di s_1 ed s_2
- x_B ed x_F valgono sulla base di quanto appena descritto, considerando che si pongono a 0 le variabili fuori base, ottenendo quindi la soluzione.
- Ora esprimiamo il valore della funzione obiettivo in corrispondenza della base corrente

Esempio

Nuova base $\Rightarrow$ variabile fuori base aumenta con **effetti su x_B e su z_B**

$$\begin{aligned} x_B &= B^{-1}b - B^{-1}F x_F \\ z &= c^T x = c_B^T x_B + c_F^T x_F = c_B^T B^{-1}b + (c_F^T - c_B^T B^{-1}F) x_F \end{aligned}$$

Esprimere x_B e z in funzione delle variabili **fuori base**

Il passaggio ad una nuova base comporta la variazione (in aumento) di almeno una delle variabili attualmente fuori base. Tale variazione avrà degli effetti sul valore delle variabili attualmente in base e sul valore della funzione obiettivo.

Per semplificare i calcoli a mano, usiamo metodo di **Gauss-Jordan**:

$$\begin{aligned} Ax = b & \rightsquigarrow [B \ F \ | \ b] \rightsquigarrow [I \ \bar{F} \ | \ \bar{b}] \\ x_B &= \bar{b} - \bar{F} x_F \quad z = \dots \end{aligned}$$

Esprimiamo pertanto il valore della funzione obiettivo e delle variabili attualmente in base nei termini delle variabili fuori base, sfruttando le equazioni dei vincoli.

z esprime: "uno scalare valore della funzione obiettivo in corrispondenza della soluzione di base [quindi sarà -71 se la base è quella di prima] viene sommato ad alcuni coefficienti delle variabili fuori base". Inoltre, "dato il sistema standard e data una base, rigiro il sistema scrivendolo in modo tale che le variabili base siano funzione delle sole variabili base e la funzione obiettivo viene scritta in funzione delle variabili fuori base".

Esempio

$$\begin{array}{cccccc|c}
 x_1 & x_2 & s_3 & s_1 & s_2 & \bar{b} \\
 \hline
 \begin{pmatrix} 3 & 4 & 0 & 1 & 0 \\ 1 & 4 & 0 & 0 & 1 \\ 3 & 2 & 1 & 0 & 0 \end{pmatrix} & & & & & \begin{pmatrix} 24 \\ 20 \\ 18 \end{pmatrix} \\
 (R_1/3) & 1 & 4/3 & 0 & 1/3 & 0 & 8 \\
 (R_2 - R_1/3) & 0 & 8/3 & 0 & -1/3 & 1 & 12 \\
 (R_3 - R_1) & 0 & -2 & 1 & -1 & 0 & -6 \\
 \\
 (R_1 - 1/2 R_2) & 1 & 0 & 0 & 1/2 & -1/2 & 2 \\
 (3/8 R_2) & 0 & 1 & 0 & -1/8 & 3/8 & 9/2 \\
 (R_3 + 3/4 R_2) & 0 & 0 & 1 & -5/4 & 3/4 & 3
 \end{array}$$

Applichiamo Gauss-Jordan rispetto alle colonne date dei vincoli, le variabili base (s_3) e le variabili fuori base. Ora, cerco di far venir fuori, al posto delle singole colonne, quelle della matrice identità, tali da ricavarsi i valori della f. o.

$$\begin{array}{lcl}
 x_1 & = & 2 - 1/2 s_1 + 1/2 s_2 \\
 x_2 & = & 9/2 + 1/8 s_1 - 3/8 s_2 \\
 s_3 & = & 3 + 5/4 s_1 - 3/4 s_2 \\
 \\
 z = -13x_1 - 10x_2 & = & -71 + 21/4 s_1 - 11/4 s_2
 \end{array}$$

La funzione obiettivo vale -71 , ricavata mettendo a 0 s_1 ed s_2 (dall'espressione della funzione obiettivo, capisco quale variabile scegliere da aumentare e trovare una soluzione ammissibile).

Con le equazioni sotto questa forma, è subito evidente che, per migliorare il valore della funzione obiettivo, si potrebbe lasciare $s_1 = 0$ e aumentare il valore della variabile s_2 : per ogni unità di aumento di s_2 si ottiene una diminuzione della funzione obiettivo di $11/4$. Converrebbe quindi aumentare s_2 quanto più possibile. La variazione di s_2 , però, si deve riflettere sulle variabili attualmente in base, in modo che la soluzione rimanga ammissibile (vincoli di uguaglianza rispettati). Infatti:

$$\begin{array}{lcl}
 x_1 & = & 2 + 1/2 s_2 \\
 x_2 & = & 9/2 - 3/8 s_2 \\
 s_3 & = & 3 - 3/4 s_2
 \end{array} \quad (\text{quello che cambia è avere } s_1 \text{ a zero nelle tre equazioni})$$

È quindi evidente che, all'aumentare di s_2 , x_1 tende ad aumentare, mentre sia x_2 sia s_3 diminuiscono. Se s_2 assume valori troppo elevati, per garantire che i vincoli di uguaglianza siano rispettati, x_2 e/o s_3 potrebbero diventare negative, portando ad una soluzione che, pur rispettando i tre vincoli di uguaglianza, viola i vincoli di non negatività delle variabili e sarebbe quindi non ammissibile.

Si intende quindi che le variabili siano ≥ 0 e si risolvano le disuguaglianze rispetto alla variabile che aumento (in questo caso s_2). Qui un riassunto:

Esempio

$$z = -71 + 21/4 s_1 - 11/4 s_2$$

- Conviene aumentare s_2 (e lasciare s_1 a 0)
- Per mantenere i vincoli di uguaglianza:

$$\begin{array}{lcl}
 x_1 & = & 2 + 1/2 s_2 \\
 x_2 & = & 9/2 - 3/8 s_2 \\
 s_3 & = & 3 - 3/4 s_2
 \end{array}$$

- Per mantenere i vincoli di non negatività:

$$\begin{array}{lcl}
 x_1 \geq 0 & \Rightarrow & 2 + 1/2 s_2 \geq 0 \Rightarrow s_2 \geq -4 \text{ sempre!} \\
 x_2 \geq 0 & \Rightarrow & 9/2 - 3/8 s_2 \geq 0 \Rightarrow s_2 \leq 12 \\
 s_3 \geq 0 & \Rightarrow & 3 - 3/4 s_2 \geq 0 \Rightarrow s_2 \leq 4
 \end{array}$$

- Soluzioni ammissibili e miglioranti ponendo $s_1 = 0$ e $0 \leq s_2 \leq 4$
- $s_2 = 4 \Rightarrow s_3 = 0$: soluzione ammissibile di base e migliorante

Se s_2 vale 10, allora x_1 vale 7 e x_2 vale $3/4$, s_3 vale $-18/4$ (non ammissibile)

Si potrebbero ottenere soluzioni non di base, con più variabili a 0 rispetto a quelle consentite.

Ricerca operativa semplice (per davvero)

Dalla condizione più stringente, si ottiene quindi il valore limite 4. Ora, ponendo $s_2 = 4$, si ha $s_3 = 0$: abbiamo quindi una nuova soluzione di base

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix} \quad x_F = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$z = -82$$

che corrisponde al vertice C . Ripetiamo il ragionamento con la nuova base. Effettuando le sostituzioni per esprimere il valore della funzione obiettivo e delle variabili in base nei termini delle variabili attualmente fuori base, si ottiene (per semplicità possiamo partire dalle ultime relazioni ottenute):

$$\begin{aligned} \text{v.3 } s_2 &= -4/3(s_3 - 3 - 5/4s_1) = 4 + 5/3 s_1 - 4/3 s_3 \\ \text{v.1 } x_1 &= 2 - 1/2s_1 + 1/2(4 + 5/3s_1 - 4/3s_3) = 4 + 1/3 s_1 - 2/3 s_3 \\ \text{v.2 } x_2 &= 9/2 + 1/8s_1 - 3/8(4 + 5/3s_1 - 4/3s_3) = 3 - 1/2 s_1 - 1/2 s_3 \\ \text{f.o. } z &= -71 + 21/4s_1 - 11/4(4 + 5/3s_1 - 4/3s_3) = -82 + 2/3 s_1 + 11/3 s_3 \end{aligned}$$

Dall'ultima relazione è evidente come la funzione obiettivo non possa essere migliorata, visto che qualsiasi variazione ammissibile (in aumento) delle variabili attualmente fuori base ne farebbe aumentare il valore. Pertanto possiamo concludere che la soluzione corrente è una soluzione ottima di base (e quindi una delle soluzioni ottime) del problema dei profumi. Il problema è stato pertanto risolto esplorando solo due tra le $\binom{5}{3} = 10$ possibili soluzioni di base. Cerchiamo adesso di generalizzare i passi che ci hanno permesso tale risultato.

Esempio

Nuova base ammissibile (s_2 al posto di s_3 in base):

$$B = \begin{bmatrix} 3 & 4 & 0 \\ 1 & 4 & 1 \\ 3 & 2 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (\text{vertice } C) \quad x_B = \begin{bmatrix} x_1 \\ x_2 \\ s_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix}$$

$$x_F = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad z_B = c^T x = c_B^T x_B + c_F^T x_F = -82$$

Iteriamo il ragionamento: x_B e z in funzione di x_F :

$$\begin{aligned} x_1 &= 4 + 1/3 s_1 - 2/3 s_3 \\ x_2 &= 3 - 1/2 s_1 - 1/2 s_3 \\ s_3 &= 4 + 5/3 s_1 - 4/3 s_3 \\ z &= -82 + 2/3 s_1 + 11/3 s_3 \end{aligned}$$

Soluzione ottima! Visitate 2 delle $\binom{5}{3} = 10$ possibili basi

Forma canonica di un problema PL

PL $\min\{z = c^T x : Ax = b, x \geq 0\}$ è in **forma canonica rispetto alla base B** di A se tutte le variabili in base e la funzione obiettivo sono scritte esplicitamente **nei termini delle sole variabili fuori base**:

$$\begin{aligned} z &= \bar{z}_B + \bar{c}_{F_1} x_{F_1} + \bar{c}_{F_2} x_{F_2} + \dots + \bar{c}_{F_{(n-m)}} x_{F_{(n-m)}} \\ x_{B_i} &= \bar{b}_i - \bar{a}_{iF_1} x_{F_1} - \bar{a}_{iF_2} x_{F_2} - \dots - \bar{a}_{iF_{(n-m)}} x_{F_{(n-m)}} \quad (i = 1 \dots m) \end{aligned}$$

$\bar{z}_B$ scalare (valore f.o. per la soluzione di base)

$\bar{b}_i$ scalare (valori variabili di base)

B_i indice della i -esima variabile in base (ce ne sono m)

F_j indice della j -esima variabile fuori base (ce ne sono $n - m$)

$\bar{c}_{F_j}$ coefficiente della j -esima variabile fuori base in funzione obiettivo (**costo ridotto della variabile rispetto alla base B**)

$-\bar{a}_{iF_j}$ coefficiente della j -esima variabile fuori base nel vincolo che esprime la i -esima variabile in base

Posso capire se la soluzione è ottima o migliorabile; data una base B , prendo il problema, lo pongo in forma standard e trovo una forma canonica rispetto a quella base (quindi, questo a sx). Così, si può anche capire subito se è illimitato.

Tramite i costi ridotti, vediamo quanto aumenterebbe la f. o. rispetto alla base corrente (incremento marginale rispetto alle v. fuori base)

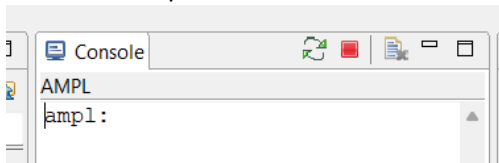
Tramite i vincoli, posso capire il limite oltre il quale non è migliorabile la funzione obiettivo (cambiando base).

Laboratorio 2 – AMPL per programmazione matematica (da slide “Introduzione ai risolutori Excel e AMPL”)

Preliminari:

- Scaricare AMPL ed unzipparlo
- Andare in percorso in locale che contenga *ampl.exe* dentro un path simile a:
 - o *C:\(percorso)\ampl.mswin64\amplide*

In AMPL è disponibile una console:



Selezionando “New Folder”, si apre una cartella e salviamo il file *contadino.mod*.

(Assicurarsi di fare doppio clic sulla cartella dove si sceglie di salvare il file .mod, in questo modo seleziona solo quella e il file viene eseguito). Scriviamo i vincoli del modello sotto come segue:

- dichiariamo le variabili con *var*
- dichiariamo la funzione obiettivo con *maximize-max* e *minimize-min*
- dichiariamo i vincoli con *subject to/s.t.*

■ Variabili decisionali:

x_L : quantità in ettari da destinare a lattuga

x_P : quantità in ettari da destinare a patate

■ Funzione obiettivo:

$\max 3000 x_L + 5000 x_P$ (ricavo totale da massimizzare)

■ Sistema dei vincoli:

$x_L + x_P \leq 11$ (ettari disponibili)
 $7 x_L \leq 70$ (semi disponibili)
 $3 x_P \leq 18$ (tuberi disponibili)
 $10 x_L + 20 x_P \leq 145$ (fertilizzante disponibile)
 $x_L \geq 0, x_P \geq 0$ (dominio)

```
var xL; #ettari da coltivare a lattuga
var xP; #ettari patate

maximize resa: 3000*xL+5000*xP; #f.o.

subject to ettari: xL+xP <= 11;
subject to semi: 7*xL <= 70;
s.t. tuberi: 3*xP<=18;
s.t. concime: 10*xL+20*xP <= 145;
s.t. domL: xL >= 0;
s.t. domP: xP >= 0;
```

Su console, se immesse variabili, si immette *reset*. Poi, per eseguire il file si lancia il comando:

- *model contadino.mod;* (questo esegue uno ad uno i comandi del file .mod).
- *option solver cplex;* (seleziona il solver d’uso di PL)
- *solve;* (risolve il problema)
- Se volessi vedere le variabili lancio *display (variabili)*

Nota:

- se si scrivono male i comandi, si va in modalità *ampl?* si scriva “;” per finire il comando (quando si ha *ampl?* significa che non si è finito di scrivere un particolare comando) e si reimmettano tutti i comandi

Comandi per la soluzione del modello (da console):

```
option solver cplex; # scelta del motore di
                    ottimizzazione (*)
solve; # risolve il modello

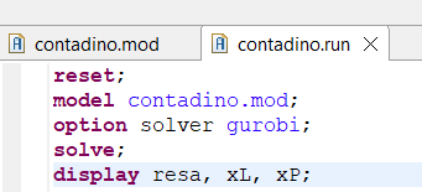
display xL, xP; # visualizza il valore
                (ottimo) delle variabili
```

(*) possibili solver (vedi eseguibili disponibili nella cartella di installazione):

PLI: *cplex (o cplexamp), gurobi, xpress ...*

PNL: *minot (default solver), baron ...* (non gestiscono variabili intere)

Si scrive poi il file `.run` nella stessa cartella che carica i comandi di risoluzione di un modello in uno script e viene eseguito tramite il comando `include contadino.run`:



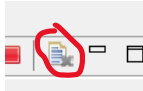
Risolvere il problema dei telecomandi con AMPL

Per le variabili intere:

```
var xA integer;
```

[Risorse]

- `telecomandi.mod`
- `telecomandi.dat`
- `telecomandi.run`

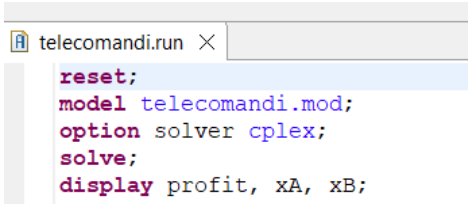


Il tastino indicato in questa piccola foto è il *clean* della console; non esiste un comando apposito in linea per farlo (not really nice)

Attenzione: mettere a *tutte le variabili e comandi* (soprattutto anche sulla console) il punto e virgola e un nome di riferimento.

```
var xA integer;
var xB integer;

maximize profit: 3*xA+6*xB;
s.t. display: xA+2*xB <= 10;
s.t. nav: xA <= 9;
s.t. tastierini: 2*xA + 3*xB <= 21;
s.t. logica: 2*xA + 3*xB <= 18;
s.t. trasmissione: xA + 3*xB <= 22;
s.t. led: xA <= 10;
```



Comandi per farlo andare:
`include telecomandi.run`

Le variabili possono avere dei bounds (lower/L e upper/U), dichiarandole come segue:

```
var nomeVar  [ >= LB ] [ , <= UB ]
               [integer/binary];
```

- I modelli precedenti includono i «dati» del problema:
 - Se cambiano i dati bisogna cambiare il modello (.mod)
 - Poca leggibilità
 - Difficile riportare modifiche del modello
- Separare modello e dati
 - File « .mod » con il modello «generale» e la definizione dei parametri del problema
 - File « .dat » con i dati attribuiti ai parametri
- Per uno stesso modello possiamo utilizzare diversi file dati (ad esempio «contadino» e «telecomandi»)

Vogliamo cercare di prendere i modelli e scriverli in una forma adatta ad AMPL, per esempio:

Modello in AMPL: sintassi base

Modelli di mix ottimo di produzione

$$\begin{aligned} \max \quad & \sum_{i \in I} P_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \leq Q_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ | \{0,1\}] \quad \forall i \in I \end{aligned}$$

I insieme dei beni che possono essere prodotti;
 J insieme delle risorse disponibili;
 P_i profitto (unitario) per il bene $i \in I$;
 Q_j quantità disponibile della risorsa $j \in J$;
 A_{ij} quantità di risorsa j necessaria per la produzione di un'unità del bene i .

```
#DICHIARAZIONE INSIEMI
set Prodotti;
set Risorse;

#DICHIARAZIONE PARAMETRI
param maxNumProd;      # massimo numero prodotti
param P {Prodotti};    # profitto unitario
param Q {Risorse};     # disponibilità risorsa
param A {Prodotti,Risorse}; # risorsa per unità di pr.

var x {Prodotti} >=0 , <= maxNumProd;

maximize profitto: sum {i in Prodotti} P[i]*x[i];

subject to disponib {j in Risorse}:
    sum {i in Prodotti} A[i,j]*x[i] <= Q[j];
```

Espressioni indicizzanti

Insiemi (set) → Un insieme è semplicemente una lista degli elementi che lo costituiscono. Serve ad esempio per dichiarare un parametro indicizzato su un insieme, ma anche per varie altre cose.

Parametri (param) → I parametri sono i dati (non variabili) che compaiono nel modello che descrivono il nostro problema di ottimizzazione; si mettono le graffe per intendere un vettore di parametri

Creiamo un file *mixOpt.mod*, tali da crearci fisicamente modelli e dati e separarli, per sicurezza e strutturazione. Per esempio, voglio avere un modello in cui le componenti fisse sono variabili (non decisionali, ma dati che variano *prima* di risolvere il problema e trovare il valore ottimo con il solver). In questo modo, scriviamo insiemi e parametri rispetto all'immagine del modello di mix ottimo di produzione:

```
mixOpt.mod ×
set I; #prodotti
set J; #risorse

param maxNumProd;
param P{I};
param Q{J};
param A{I,J};

var x {i in I} >= 0 <= maxNumProd;

maximize profitto: sum{i in I} P[i]*x[i];
s.t. dispon{j in J}: sum{i in I} A[i, j]*x[i] <= Q[j];
```

(Attenzione: se come me capita che sparisca la console, l'unico modo per ristabilire le view è cliccare sul menù "Window" e cliccare "Reset and restart IDE").

La struttura di sintassi base segue (esempio del contadino):

Dati in AMPL: ^{sintassi} sintassi base

```
set Prodotti := lattuga patata;
set Risorse := ettari semi tuberi concime;
```

```
param maxNumProd := 7;
```

```
param P :=
lattuga 3000
patata 5000
;
```

```
param Q :=
ettari 11
semi 70
tuberi 18
concime 145
;
```

```
A{Prodotti=righe , Risorse=colonne}
```

```
param A : ettari semi tuberi concime :=
lattuga 1 7 0 10
patata 1 0 3 20
;
```

■ Variabili decisionali:

x_L : quantità in ettari da destinare a lattuga

x_P : quantità in ettari da destinare a patate

■ Funzione obiettivo:

$\max 3000 x_L + 5000 x_P$ (ricavo totale da massimizzare)

■ Sistema dei vincoli:

$x_L + x_P \leq 11$ (ettari disponibili)

$7 x_L \leq 70$ (semi disponibili)

$3 x_P \leq 18$ (tuberi disponibili)

$10 x_L + 20 x_P \leq 145$ (fertilizzante disponibile)

$x_L \geq 0, x_P \geq 0$ (dominio)

Da mia osservazione, *maxNumProd* vale 7 in quanto abbiamo per ciascuno dei 2 prodotti almeno 4 varietà di risorse disponibili, per un totale complessivo di 6. Di conseguenza, 7 viene impostato come upper bound-limite superiore di riferimento sulle quantità.

A tale scopo, creiamo il file *mixOpt_contadino.dat*, con il codice che segue:

```
mixOpt_contadino.dat X
set I := lat pat;
set J := ett sem tub con;

param maxNumProd := 7;

param P :=
ett 11
sem 70
tub 18
con 145;
#ricordarsi che gli insiemi non sono ordinati
#dunque possiamo mettere le coppie
#[nome indice - valore]
#si ricordi che gli spazi vengono ignorati; si può scrivere in linea singola

param Q := lat 3000 pat 5000;

param A := lat ett 1 latt sem 7 latt tub 0 lat con 10
#prima riga: stiamo guardando i vincoli vedendo quanto vale xL
pat sem 0 pat ett 1 pat tub 3 pat con 20;
#seconda riga: stiamo guardando i vincoli vedendo quanto vale xP
```

Poi, carichiamo tutto con:

- *model mixOpt.mod*
- *data minOpt_contadino.dat*

Se inserisco il comando *expand*, riesco a visualizzare il modello esteso.

Posso inoltre cancellare solo un singolo file, per esempio il file dei dati con → *reset data*

Metodo del simplesso: cambio base, costi ridotti, variabili entranti ed uscenti, condizione di illimitatezza, passi metodo del simplesso

Dato un problema di PL in forma canonica rispetto ad una base B , la funzione obiettivo esprime come varierebbe il valore corrente della funzione obiettivo se si modificassero i valori delle variabili fuori base. In questo senso, il coefficiente $\bar{c}_{F_j}$ corrisponde al costo della variabile fuori base x_{F_j} quando la funzione obiettivo è nella forma canonica e viene detto costo ridotto, definito come incremento marginale unitario della f.o.

Si noti che le variabili in base non compaiono nella funzione obiettivo in forma canonica e assumiamo che il loro costo ridotto sia pari a 0.

Definizione 11 (Costo ridotto). *Dato un problema di PL in forma canonica rispetto a una base B , il coefficiente $\bar{c}_j$ della variabile x_j nella funzione obiettivo si dice costo ridotto della variabile x_j rispetto alla base B .*

Il costo ridotto $\bar{c}_{F_j}$ rappresenta l'incremento marginale del costo complessivo (funzione obiettivo di minimo) per ogni unità di variazione in aumento della variabile x_{F_j} . La definizione *ridotto* deriva dal fatto che, come vedremo, $\bar{c}_{F_j}$ si può facilmente calcolare sottraendo al costo originario c_{F_j} una quantità che dipende dalla base B .

Teorema di soluzione ottima

- Sia dato un problema di programmazione lineare e una sua base ammissibile B . Se tutti i costi ridotti rispetto alla base B sono non negativi (≥ 0) allora la soluzione di base associata a B è ottima.

Operazione di cambio base

- Se il test di ottimalità dà esito negativo, allora possiamo migliorare la soluzione corrente cambiando la base. Le operazioni di cambio base di nostro interesse sono quelle che consentono di passare dalla base ammissibile B ad un'altra base $\bar{B}$ con le seguenti caratteristiche:
 1. $\bar{B}$ è una base *adiacente* a B (due basi si dicono adiacenti se differiscono per una sola colonna);
 2. la soluzione di base associata a $\bar{B}$ migliora il valore della funzione obiettivo;
 3. la soluzione di base associata a $\bar{B}$ rimane una soluzione ammissibile.

Si riassume tutto nei seguenti passi:

Metodo del simplesso: cambio base

- Da base B ammissibile a $\bar{B}$ **adiacente, ammissibile, migliorante**
- **Una** colonna ($\approx$ variabile) entrante al posto di una uscente

- Variabile **entrante** (miglioramento): *qualsiasi* $x_h : \bar{c}_h < 0$

$$z = \bar{z}_B + \bar{c}_h x_h = \bar{z}_B \leq \bar{z}_B$$

- Variabile **uscente** (ammissibilità): [regola del quoziente minimo]

$$x_{B_i} \geq 0 \Rightarrow \bar{b}_i - \bar{a}_{ih} x_h \geq 0, \forall i \Rightarrow x_h \leq \frac{\bar{b}_i}{\bar{a}_{ih}}, \forall i : \bar{a}_{ih} > 0$$

$$t = \arg \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$$

$$x_h = \frac{\bar{b}_t}{\bar{a}_{th}} \geq 0 \Rightarrow x_{B_t} = 0 \text{ [} x_{B_t} \text{ esce!]}$$

La prima condizione corrisponde alla scelta di due variabili (e relative colonne) interessate al cambio base: una variabile x_{B_t} nella base di partenza che esca dalla base e una variabile x_{F_h} fuori dalla base di partenza che entri nella nuova base.

Ora dobbiamo capire:

- la variabile che *entra* in base, che è guidata dalla condizione di trovare una soluzione associata a $\tilde{B}$ tale da migliorare la funzione obiettivo. Richiamando la forma della funzione obiettivo:

$$z = \bar{z}_B + \bar{c}_{F_1} x_{F_1} + \bar{c}_{F_2} x_{F_2} + \dots + \bar{c}_{F_{(n-m)}} x_{F_{(n-m)}}$$

Scelta della variabile entrante x_h : facciamo entrare in base una qualsiasi variabile con costo ridotto strettamente negativo: $x_h: \bar{c}_h < 0$

- o Nella pratica, la prima variabile con costo ridotto negativo
- la variabile che *esce* dalla base, per cui scegliendo x_h , tutte le altre variabili vengono lasciate a 0 e si cerca di aumentare il più possibile il valore di x_h , in modo da rendere ammissibile la soluzione e rispettare i vincoli del problema (al fine di migliorare la funzione obiettivo).
 - o Nella pratica, la variabile che rispetta la regola (qui sotto) del rapporto minimo

Può succedere, comunque, che x_h assuma valori troppo elevati e, in tal caso, renda inammissibile la soluzione, perché x_{B_i} potrebbe assumere valori negativi; bisogna sempre tenere conto dei vincoli di non negatività. Il rispetto dell'ammissibilità della nuova base si risolve con una serie di rapporti come segue:

$$x_h \leq \frac{\bar{b}_i}{\bar{a}_{ih}} \quad \forall \quad i = 1 \dots m : \bar{a}_{ih} > 0$$

Questo serve a *limitare* l'aumento di x_h al valore di *rapporto minimo*.

$$\theta = \min \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : i = 1 \dots m, \bar{a}_{ih} > 0 \right\} \quad \begin{array}{l} x_h \geq 0 \rightarrow x_h \text{ entra in base ad un valore non negativo} \\ x_{B_t} = 0 \rightarrow x_{B_t} \text{ esce dalla base} \end{array}$$

Facciamo quindi uscire dalla base una qualsiasi delle variabili con $\bar{a}_{ih} > 0$ e che corrispondono al minimo valore $\frac{\bar{b}_i}{\bar{a}_{ih}}$:

$$t = \arg \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$$

$$\text{ponendo } x_h = \theta = \frac{\bar{b}_t}{\bar{a}_{th}}$$

Il cambio di base porta la f.o. a migliorare, ma esiste un caso particolare quando $\bar{a}_{ih} \leq 0$. All'aumentare del valore di x_h , tutte le variabili attualmente in base aumentano o restano invariate. Non si hanno limiti alla crescita di x_h e quindi alla diminuzione della f.o.

Metodo del simplesso: condizione di illimitatezza

- Consideriamo una qualsiasi x_h : $\bar{c}_h < 0$.

$$z = \bar{z}_B + \bar{c}_h x_h$$

$$x_{B_i} = \bar{b}_i - \bar{a}_{ih} x_h \quad (i = 1 \dots m)$$

- Se $\bar{a}_{ih} \leq 0, \forall i = 1 \dots m$, soluzione ammissibile con $x_h \rightarrow +\infty$

In altri termini, facendo aumentare arbitrariamente x_h si ottengono delle soluzioni (non di base) con valore della funzione obiettivo arbitrariamente basso (tendente a $-\infty$)

Condizione di illimitatezza di un PL

Esiste una base per cui

$$\exists x_h : (\bar{c}_h < 0) \wedge (\bar{a}_{ih} \leq 0, \forall i = 1 \dots m)$$

N.B.: condizione sufficiente

Tutte queste osservazioni permettono di sintetizzare il metodo del simplesso in una serie di passi, poi sintetizzati graficamente:

- 0) *Inizializzazione*, partendo da un problema in forma standard e da una base ammissibile B
- 1) *Passaggio alla forma canonica*, scrivendo il modello in forma canonica rispetto alla base corrente B
- 2) *Test di ottimalità*, se tutti i costi ridotti delle variabili fuori base sono > 0 oppure $= 0$ allora la soluzione della base corrente B è ottima
- 3) *Test di illimitatezza*, se esiste una qualsiasi variabile fuori base allora il problema è illimitato e l'algoritmo termina
- 4) *Scelta della variabile entrante per il cambio base*, scegliendo una variabile x_h con costo ridotto $\bar{c}_h < 0$
- 5) *Scelta della variabile uscente per il cambio base*, scegliendo x_{B_t} con
$$t = \arg \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$$
- 6) *Cambio base e iterazione*, aggiornare la base corrente eliminando la colonna di x_{B_t} e la sostituendo con la colonna di x_h . Da qui torno al passo 1.

Metodo del simplesso: sintesi

Init: PL in forma standard $\min \{c^T x : Ax = b, x \geq 0\}$ e una base ammissibile di partenza B

repeat

mettere in **forma canonica** rispetto a B

$$z = \bar{z}_B + \bar{c}_{F_1} x_{F_1} + \bar{c}_{F_2} x_{F_2} + \dots + \bar{c}_{F_{(n-m)}} x_{F_{(n-m)}}$$

$$x_{B_i} = \bar{b}_i - \bar{a}_{iF_1} x_{F_1} - \bar{a}_{iF_2} x_{F_2} - \dots - \bar{a}_{iF_{(n-m)}} x_{F_{(n-m)}} \quad (i = 1 \dots m)$$

if $(\bar{c}_j \geq 0, \forall j)$ **then** PL ottimo con B base ottima: **continue**

if $(\exists h : \bar{c}_h < 0 \text{ and } \bar{a}_{ih} \leq 0, \forall i)$ **then** PL illimitato: **continue**

variabile **entrante**: una $x_h : \bar{c}_h < 0$

variabile **uscente**: x_{B_t} con $t = \arg \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$

$B \leftarrow B \oplus A_h \ominus A_{B_t}$ [**cambio base**]

until (PL ottimo o illimitato)

Il passaggio alla forma canonica può essere ottenuto utilizzando il metodo di Gauss-Jordan per la soluzione di sistemi di equazioni lineari del tipo
 $Ax = b$.

Il metodo consiste nel costruire la matrice estesa del sistema $[A|b]$ e nel sostituire ciascuna riga con combinazioni lineari delle righe della stessa matrice estesa fino ad ottenere una forma canonica dalla quale sia facile ricavare una soluzione del sistema.

Normalmente, si considera nel pivoting lo stesso elemento che si fa entrare/uscire dalla base stessa.

Il tableau del semplice

$$z = c^T x \quad Ax = b$$

- Per ottenere la forma canonica "a mano" con Gauss-Jordan

- Funzione obiettivo come vincolo** (z nuova variabile):

$$z = c_1 x_1 + c_2 x_2 + \dots + c_n x_n \rightsquigarrow c_1 x_1 + c_2 x_2 + \dots + c_n x_n - z = 0$$

	x_{B_1}	...	x_{B_m}	x_{F_1}	...	$x_{F_{n-m}}$	z	$\bar{b}$
riga 0	c_B^T			c_F^T			-1	0
riga 1	B			F			0	b
$\vdots$							$\vdots$	
riga m							0	

- Operazioni elementari su righe (inclusa z): x_B (e z) in funzione di x_F

Questa forma a tabella serve per facilitare le operazioni di sostituzione, partendo da un problema PL in forma standard, poi rappresentato da un tableau iniziale. Una volta scelta una base ammissibile di partenza B , si considera la partizione $A = [B|F]$ che si riflette sul tableau (dopo aver scambiato opportunamente le colonne).

	x_{B_1}	...	x_{B_m}	x_{F_1}	...	$x_{F_{n-m}}$	z	$\bar{b}$
riga 0	0	...	0	$\square$	...	$\square$	-1	$\square$
riga 1	1		0	$\square$	...	$\square$	0	$\square$
$\vdots$				$\square$	...	$\square$	$\vdots$	$\square$
riga m	0		1	$\square$	...	$\square$	0	$\square$

Tableau in forma canonica

- Operazioni elementari su righe (inclusa z): x_B (e z) in funzione di x_F

La forma canonica prevede una serie di coefficienti al posto dei "quadrantini", in particolare $\bar{c}_{F_i}$ e $\bar{a}_{iF_i}$, come si può vedere subito anche sotto. In particolare, effettuando una serie di sostituzioni, otteniamo proprio il **tableau in forma canonica**:

Tableau e forma canonica

	x_{B_1}	...	x_{B_m}	x_{F_1}	...	$x_{F_{n-m}}$	z	$\bar{b}$
$-z$	0	...	0	$\bar{c}_{F_1}$	...	$\square$	-1	$\square$
x_{B_1}	1		0	$\bar{a}_{1F_1}$	...	$\square$	0	$\square$
x_{B_i}				$\square$	...	$\square$	$\vdots$	$\square$
x_{B_m}	0		1	$\square$	...	$\square$	0	$\square$

$$z = \bar{z}_B + \bar{c}_{F_1} x_{F_1} + \bar{c}_{F_2} x_{F_2} + \dots + \bar{c}_{F_{(n-m)}} x_{F_{(n-m)}}$$

$$x_{B_i} = \bar{b}_i - \bar{a}_{iF_1} x_{F_1} - \bar{a}_{iF_2} x_{F_2} - \dots - \bar{a}_{iF_{(n-m)}} x_{F_{(n-m)}} \quad (i = 1 \dots m)$$

Definendo questo tableau come forma schematizzata della forma canonica di un problema PL si ha che:

Tableau e forma canonica

- l'ultima colonna del tableau riporta la soluzione del problema rispetto alla base corrente, cioè le variabili in base e, nella prima riga, l'opposto della funzione obiettivo.
- le colonne delle variabili in base corrispondono alla matrice identità sormontata da una riga di zeri (costi ridotti variabili in base)
- le colonne delle variabili fuori base corrispondono ai coefficienti $\bar{a}_{iF_j}$ della forma canonica e, nella prima riga, riportano i costi ridotti

	x_{B_1}	...	x_{B_m}	x_{F_1}	...	$x_{F_{n-m}}$	z	$\bar{b}$
$-z$	0	...	0	$\bar{c}_{F_1}$	...	$\bar{c}_{F_{n-m}}$	-1	$-\bar{z}_B$
x_{B_1}	1		0	$\bar{a}_{1F_1}$	...	$\bar{a}_{1F_{n-m}}$	0	$\bar{b}_1$
x_{B_i}		...		$\bar{a}_{iF_1}$	...	$\bar{a}_{iF_{n-m}}$	$\vdots$	$\bar{b}_i$
x_{B_m}	0		1	$\bar{a}_{mF_1}$	...	$\bar{a}_{mF_{n-m}}$	0	$\bar{b}_m$

$$z = \bar{z}_B + \bar{c}_{F_1} x_{F_1} + \bar{c}_{F_2} x_{F_2} + \dots + \bar{c}_{F_{(n-m)}} x_{F_{(n-m)}}$$

$$x_{B_i} = \bar{b}_i - \bar{a}_{iF_1} x_{F_1} - \bar{a}_{iF_2} x_{F_2} - \dots - \bar{a}_{iF_{(n-m)}} x_{F_{(n-m)}} \quad (i = 1 \dots m)$$

Quindi, sto diagonalizzando, al fine di ottenere i numeri che voglio, tale da ottenere i numeri del simplesso da usare nel tableau.

Il tableau in forma canonica, come osservato, contiene tutti i dati necessari per:

- valutare l'ottimalità della base corrente: se tutti i costi ridotti nella prima riga sono ≥ 0 ;
- valutare l'illimitatezza del problema: se esiste una colonna con costo ridotto negativo nella prima riga e con restanti coefficienti tutti ≤ 0 ;
- effettuare l'operazione di cambio base

Per tutti i passi descritti dal simplesso sul tableau:

- la verifica dei costi ridotti (ottimalità) si fa sulla riga di $-z$, tale che si abbiano valori ≥ 0
- la verifica di illimitatezza, per cui se sotto ad un certo $\bar{c}_h$ si hanno tutti numeri negativi, allora mi fermo perché il problema è illimitato
- per la variabile entrante, se trovo un numero $c_h < 0$, allora posso far entrare in base una variabile x_h
- per la variabile uscente, verifico i rapporti tra i b_i (visibili nell'ultima colonna a destra) e gli a_{ih} (visibili nella colonna della variabile entrante x_h)

Il simplesso sul tableau

	x_{B_1}	...	x_{B_t}	...	x_{B_m}	x_{F_1}	...	x_h	...	$x_{F_{n-m}}$	z	$\bar{b}$
$-z$	0	...	0	...	0	$\bar{c}_{F_1}$	...	$\bar{c}_h$	...	$\bar{c}_{F_{n-m}}$	-1	$-\bar{z}_B$
x_{B_1}	1	...	0	...	0	$\bar{a}_{1F_1}$	...	$\bar{a}_{1h}$	...	$\bar{a}_{1F_{n-m}}$	0	$\bar{b}_1$
<u>x_{B_t}</u>	0	...	1	...	0	$\bar{a}_{tF_1}$	...	$\bar{a}_{th}$	...	$\bar{a}_{tF_{n-m}}$	$\vdots$	$\bar{b}_t$
x_{B_m}	0	...	0	...	1	$\bar{a}_{mF_1}$	...	$\bar{a}_{mh}$	...	$\bar{a}_{mF_{n-m}}$	0	$\bar{b}_m$

- verifica ottimalità: costi ridotti sulla riga $-z$ con valori ≥ 0
- verifica illimitatezza: colonna h con $c_h < 0$ e $a_{ih} \leq 0$
- variabile **entrante**: x_h : $c_h < 0$
- variabile **uscente** : x_t , $t = \arg \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$

Attenzione: La tabella del tableau è un artificio nostro per fare i conti di sostituzione in modo ordinato, ma il tableau è un modo per implementare il simplesso sperando di non confonderci troppo. Non eseguire il simplesso in modo puramente meccanico (un minimo di teoria occorre saperla, almeno per fare l'esercizio teorico del simplesso, comunque molto semplice).

Il simplesso sul tableau

	x_{B_1}	...	x_{B_t}	...	x_{B_m}	x_{F_1}	...	x_h	...	$x_{F_{n-m}}$	z	$\bar{b}$
$-z$	0	...	0	...	0	$\bar{c}_{F_1}$	...	$\bar{c}_h$	...	$\bar{c}_{F_{n-m}}$	-1	$-\bar{z}_B$
x_{B_1}	1	...	0	...	0	$\bar{a}_{1F_1}$	...	$\bar{a}_{1h}$	...	$\bar{a}_{1F_{n-m}}$	0	$\bar{b}_1$
x_{B_t}	0	...	1	...	0	$\bar{a}_{tF_1}$	...	$\bar{a}_{th}$	...	$\bar{a}_{tF_{n-m}}$	...	$\bar{b}_t$
x_{B_m}	0	...	0	...	1	$\bar{a}_{mF_1}$	...	$\bar{a}_{mh}$	...	$\bar{a}_{mF_{n-m}}$	0	$\bar{b}_m$

Individuato chi entra e chi esce, ridiagonalizziamo la matrice rispetto alla nuova base.

Quindi, vogliamo fare in modo compaiano sempre le colonne della matrice identità per essere considerata in forma canonica.

- cambio base: operazione di **pivot** su $\bar{a}_{th}$

$$\bar{a}_{tj} \leftarrow \frac{\bar{a}_{tj}}{\bar{a}_{th}} \quad \forall \text{ colonna } j$$

$$\bar{a}_{ij} \leftarrow \bar{a}_{ij} - \frac{\bar{a}_{tj}}{\bar{a}_{th}} \bar{a}_{ih} \quad \forall \text{ riga } i \neq t, \text{ colonna } j$$

In particolare, il cambio di base procede secondo tre passi:

- individuo una colonna h che abbia sulla prima riga valore negativo
- effettuo i rapporti $\frac{\bar{b}_i}{\bar{a}_{ih}}$ per ogni riga i
- scelgo la riga t che corrisponde al minimo dei rapporti (variabile che lascia la base)

	x_{B_1}	...	x_{B_t}	...	x_{B_m}	x_{F_1}	...	x_h	...	$x_{F_{n-m}}$	$\bar{b}$
$-z$	0	...	0	...	0	$\bar{c}_{F_1}$	...	$\bar{c}_h$	...	$\bar{c}_{F_{n-m}}$	$-\bar{z}_B$
x_{B_1}	1	...	0	...	0	$\bar{a}_{1F_1}$	...	$\bar{a}_{1h}$	...	$\bar{a}_{1F_{n-m}}$	$\bar{b}_1$
x_{B_t}	0	...	1	...	0	$\bar{a}_{tF_1}$	...	$\bar{a}_{th}$	...	$\bar{a}_{tF_{n-m}}$	$\bar{b}_t$
x_{B_m}	0	...	0	...	1	$\bar{a}_{mF_1}$	...	$\bar{a}_{mh}$	...	$\bar{a}_{mF_{n-m}}$	$\bar{b}_m$

Per applicare le considerazioni a lato, facciamo un'operazione di pivot sull'elemento indicato dalla freccia in azzurro (data una colonna e una riga, significa effettuare operazioni sulle righe che trasformino l'elemento in posizione (t, h) in 1 e tutti i restanti elementi della colonna h in 0). Il nuovo tableau è come segue:

	x_{B_1}	...	x_{B_t}	...	x_{B_m}	x_{F_1}	...	x_h	...	$x_{F_{n-m}}$	$\bar{b}$
$-z$	0	...	$\bar{c}_{B_t}$	...	0	$\bar{c}_{F_1}$	...	0	...	$\bar{c}_{F_{n-m}}$	$-\bar{z}_B$
x_{B_1}	1	...	$\bar{a}_{1B_t}$	...	0	$\bar{a}_{1F_1}$	...	0	...	$\bar{a}_{1F_{n-m}}$	$\bar{b}_1$
x_h	0	...	$\bar{a}_{tB_t}$	...	0	$\bar{a}_{tF_1}$	...	1	...	$\bar{a}_{tF_{n-m}}$	$\bar{b}_t$
x_{B_m}	0	...	$\bar{a}_{mB_t}$	...	1	$\bar{a}_{mF_1}$	...	0	...	$\bar{a}_{mF_{n-m}}$	$\bar{b}_m$

Si noti che l'operazione di pivot si esegue sempre scegliendo:

- per colonna la variabile che entra
- per riga la variabile che esce

Si noti inoltre che la t -esima colonna della matrice identità è corrispondente ad x_h (quindi, la riga t si riferisce a x_h). L'ordine delle operazioni deve sempre:

- mantenere la matrice identità nei vincoli (va bene anche sparsa, quindi anche con colonne in posizioni anche non vicine) e mantenerla con gli zeri sopra (costi ridotti, quindi), per la forma standard
- mantenere la colonna di zeri rispetto a z e preservando il suo valore (qui si cerca di tenerlo a -1)

- Risolvere con il metodo del simplesso il problema del contadino, ricordando il suo modello PL:

$$\begin{array}{rcll} \max & 3000x_L & + & 5000x_P \\ & x_L & + & x_P & \leq & 12 \\ & 7x_L & & & \leq & 70 \\ & & & 3x_P & \leq & 18 \\ & 10x_L & + & 20x_P & \leq & 160 \\ & x_L & , & x_P & \geq & 0 \end{array}$$

-1) semplifica

$$\begin{array}{l} z'_{max} = 3x_L + 5x_P \quad (1000) \\ x_L + x_P \leq 12 \\ x_L \leq 10 \\ x_P \leq 6 \\ x_L + 2x_P \leq 16 \\ x_L, x_P \geq 0 \end{array}$$

Qui divide per 1000 rispetto a 3000 e 5000. Poi, semplicemente semplifica le disuguaglianze.

Successivamente, imposta il problema in forma standard e mette la funzione in forma di minimo. Si trasformano i vincoli di disuguaglianza in vincoli di uguaglianza con le variabili di slack. Si ricordi di inserire che tutte le variabili sono ≥ 0 , *comprese le variabili di slack*. Per la forma standard ci domandiamo:

- la f.o. è di minimo?
- sono tutti vincoli di uguaglianza?
- tutte le variabili sono ≥ 0 ?
- tutti i vincoli sono ≥ 0 ?

A tutte, si risponde sì.

0) Forma STD

$$\begin{array}{rcll} \min & z_{min} & = & -3x_L - 5x_P \\ \text{s.t.} & x_L + x_P + s_1 & = & 12 \\ & x_L + s_2 & = & 10 \\ & x_P + s_3 & = & 6 \\ & x_L + 2x_P + s_4 & = & 16 \\ & x_L, x_P, s_1, s_2, s_3, s_4 & \geq & 0 \end{array}$$

Rein v = v x ≥ 0 v s ≥ 0

Successivamente, si va a costruire il tableau, tracciando la prima riga per la funzione obiettivo, poi mettendo tutte le altre righe per i vincoli, riportando i singoli coefficienti, in particolare per individuare le colonne linearmente indipendenti (cioè, quelle uguali a zero).

Diciamo inoltre che la prima riga sotto le variabili corrisponde ai coefficienti della funzione obiettivo, mentre le successive righe sono le disuguaglianze. La colonna $\bar{b}$ sono tutti i termini noti e, come valore iniziale della f.o. si mette -1 .

0100) BASE DI PARTENZA $B = [s_1 \ s_2 \ s_3 \ s_4]$ è AUMENTATA

x_L	x_P	s_1	s_2	s_3	s_4	z_{row}	$\bar{b}$
-3	-5	0	0	0	0	-1	0
1	1	1	0	0	0	0	12
1	0	0	1	0	0	0	10
0	1	0	0	1	0	0	6
1	2	0	0	0	1	0	16

Parto con la prima iterazione, usando tutte come base tutte le variabili di slack, mettendo in forma canonica rispetto alla base. In questo caso, costruiamo la matrice aumentata/tableau usando i coefficienti di prima.

Solo dopo aver messo in forma canonica, scrivo a fianco delle colonne le variabili in base (che significa evidenziare in verde quali sono le variabili a cui mi sto riferendo, qui nello specifico s_1, s_2, s_3, s_4)

Mi faccio poi delle domande:

- è ammissibile? Sì $\rightarrow$ tutti i $\bar{b}_i$ sono ≥ 0 (cioè, tutta l'ultima colonna di destra)
- è ottima? Non lo so $\rightarrow$ Esiste qualche coefficiente di costo ridotto < 0 (condizione sufficiente di ottimalità; se fossero tutti uguali a zero, allora sarebbe ottima)
- è illimitata? Non lo so $\rightarrow$ Vado a vedere se in corrispondenza di colonne con costi ridotti strettamente minori di 0 abbiamo sotto una colonna tutta negativa $\rightarrow$ Esiste almeno un valore strettamente positivo in corrispondenza di qualche valore negativo e quindi non possiamo concludere con certezza (condizione sufficiente)
- chi entra in base? $\rightarrow x_P \rightarrow$ Una qualsiasi variabile con costo ridotto negativo e scelgo tra x_L e x_P
- chi esce dalla base? Calcoliamo l'argomento del rapporto minimo tra i rapporti di $\bar{b}$ e gli $\bar{a}_{ih}$ (per righe rispetto alla colonna dell'elemento del pivot); si noti che non si considerano le variabili negative (non tanto per la divisione, ma lo 0 semplicemente dice "non varia la funzione obiettivo"; se avessimo variabili < 0 vale lo stesso). Nel rapporto, in generale, si prende semplicemente il numero con rapporto minimo.

Tutto quanto appena descritto è la prima iterazione:

ITER 1: $B = [s_1 \ s_2 \ s_3 \ s_4]$

x_L	x_P	s_1	s_2	s_3	s_4	z_{row}	$\bar{b}$
-3	-5	0	0	0	0	-1	0
s_1	1	1	0	0	0	0	12
s_2	1	0	1	0	0	0	10
s_3	0	1	0	1	0	0	6
s_4	1	0	0	0	1	0	16

1) è in F.C. rispetto a z
 2) è ammissibile? Sì (tutti $\bar{b}_i \geq 0$)
 3) è ottima? NON SÌ
 4) è illimitata? NON SÌ
 5) entra? tra x_L e x_P scelgo x_P
 6) esce? $\arg \min \left\{ \frac{12}{1}, \frac{10}{1}, \frac{6}{0}, \frac{16}{2} \right\} = \arg \{6\} = s_3$

Esce quindi s_3 e passo alla iterazione 2; s_3 ha lasciato il posto ad x_P e tutto il resto (s_1, s_2, s_4) rimane in base. Ora, però, andiamo a riscrivere la matrice aumentata, usando Gauss-Jordan per le matrici facendo operazioni sulle righe.

Meglio basarsi sempre sulla riga del pivot (qui x_P) per fare le operazioni in modo da non ottenere numeri strani (quindi $\rightarrow$ riga sulla quale opero = riga sulla quale opero + (un coefficiente * la riga del pivot)). In questo caso, nelle operazioni di pivoting, R_2 ed R_3 rimangono uguali perché già nella forma giusta di matrice identità.

Dopo le operazioni, verifico se ci sono errori di calcolo:

- 1) la forma è canonica? → Sì → Verifico se ho la matrice identità sormontata da zeri dove serve (quindi, nella posizione del pivot, qui x_p). Fatto questo, etichetto le linee (con s_1, s_2, x_p, s_4)
- 2) la soluzione è ammissibile? → Sì → Le variabili della colonna di $\bar{b}$ sono tutte ≥ 0
Attenzione che un errore imperdonabile è avere variabili negative nella colonna di $\bar{b}$, cosa che non dovrebbe mai succedere partendo da una base canonica
- 3) la soluzione è ottima? → Non so (stessi motivi di sopra)
- 4) la soluzione è illimitata? → Non so (stessi motivi di sopra)
- 5) chi entra in base? → x_L → L'unica variabile con costo ridotto negativo
- 6) chi esce in base? → s_4 → La variabile con rapporto minimo

ITER 2: $B = [s_1 \ s_2 \ x_p \ s_4]$

	x_L	x_p	s_1	s_2	s_3	s_4	z.r.	$\bar{b}$
$R_0 = R_0 + 5R_3$	-3	0	0	0	5	0	-1	30
$R_1 = R_1 - R_3$	1	0	1	0	-1	0	0	6
$R_2 = R_2$	0	0	1	0	0	0	0	10
$R_3 = R_3$	0	1	0	0	1	0	0	6
$R_4 = R_4 - 2R_3$	1	0	0	0	-2	1	0	4

1) F.C. ? $s_1'!$
 2) A.T.R. ? $s_4'!$
 3) O.T.T. ? NON SO
 4) I.C.E.R. ? NON SO
 5) entra? x_L
 6) esce $\arg \min \left\{ \frac{6}{1}, \frac{10}{1}, \frac{6}{1}, \frac{4}{-2} \right\} = \arg \{4\} = s_4$

Andiamo alla terza iterazione, con x_L che entra in base. Come prima, si eseguono le operazioni rispetto alla riga del pivot, eseguendo somme algebriche di righe.

Mi faccio di nuovo le stesse domande: (con motivazioni spiegate sopra e anche dette più sinteticamente dal prof, si faccia riferimento a sopra).

- 1) la forma è canonica? → Sì
- 2) la soluzione è ammissibile? → Sì
- 3) la soluzione è ottima? → Non so
- 4) la soluzione è illimitata? → Non so
- 5) chi entra in base? → s_3
- 6) chi esce in base? → s_1

ITER 3: $B = [s_1 \ s_2 \ x_p \ x_L]$

	s_3	s_1	s_2	x_p	x_L	s_4	z.r.	$\bar{b}$
$R_0 = R_0 + 3R_4$	0	0	0	0	-1	3	-1	42
$R_1 = R_1 - R_4$	0	0	1	0	1	-1	0	2
$R_2 = R_2 - R_4$	0	0	0	1	2	-1	0	6
$R_3 = R_3$	0	1	0	0	1	0	0	6
$R_4 = R_4$	1	0	0	0	-2	1	0	4

1) F.C. ? $s_1'!$
 2) A.T.R. ? $s_4'!$
 3) O.T.T. ? NON SO
 4) I.C.E.R. ? NON SO
 5) entra? s_3
 6) esce $\arg \min \left\{ \frac{2}{1}, \frac{6}{2}, \frac{6}{1}, \frac{4}{-2} \right\} = \arg \{4\} = s_4$

Quindi, la quarta iterazione con s_3 che entra in base, eseguendo le operazioni di somma sulla base del pivot come già detto per le altre e raccomandato dal prof.

Mi faccio di nuovo le stesse domande:

- 1) la forma è canonica? → Sì
- 2) la soluzione è ammissibile? → Sì → Le variabili della colonna di $\bar{b}$ sono tutte ≥ 0

3) la soluzione è ottima? → Sì → Tutti i costi ridotti sono ≥ 0 e mi fermo

ITER 4: $\bar{B} [s_3, s_2, x_p, x_L]$

$$\begin{array}{l}
 R_0' = R_0 + R_1 \\
 R_1' = R_1 \quad s_3 \\
 R_2' = R_2 - 2R_1 \quad s_2 \\
 R_3' = R_3 - R_1 \quad x_p \\
 R_4' = R_4 + 2R_1 \quad x_L
 \end{array}
 \begin{array}{cccccc|c|c}
 0 & 0 & 1 & 0 & 0 & 2 & -1 & 44 \\
 0 & 0 & 0 & 1 & 1 & -1 & 0 & 2 \\
 0 & 0 & -2 & 1 & 0 & -1 & 0 & 2 \\
 0 & 1 & -1 & 0 & 0 & 1 & 0 & 4 \\
 1 & 0 & 2 & 0 & 0 & -1 & 0 & 8
 \end{array}$$

1) FC? 8!
 2) ADRISS? 8!
 3) STURTA? 8! !

Ora devo dare il valore della soluzione ottima, poiché il problema è ora in forma standard rispetto al problema di partenza. Quindi diamo i valori della funzione obiettivo:

$$z_{MIN} = -44 \quad z_{MAX} = +44 \quad z_{MAX} = 44000 \\
 x_L = 8 \quad x_P = 4 \quad s_1 = 0 \quad s_2 = 2 \quad s_3 = 2 \quad s_4 = 0$$

(con il 44000 che salta fuori perché all'inizio avevamo diviso per mille e:

In merito ai vincoli:

- x_L all'ottimo vale 8 (colonna di $\bar{B}$)
- x_P all'ottimo vale 4 (colonna di $\bar{B}$)
- s_1 all'ottimo vale 0 (è fuori base)
- s_2 all'ottimo vale 2 (colonna di $\bar{B}$)
- s_3 all'ottimo vale 2 (colonna di $\bar{B}$)
- s_4 all'ottimo vale 0 (è fuori base)

Sui vincoli dobbiamo fare delle precisazioni, riferite al problema di partenza (noi lo convertiamo in forma standard) ed esclusivamente alle variabili di slack. Possiamo parlare di due tipi di vincoli:

- *laschi*, nel momento in cui abbiamo una variabile di slack > 0
 - o Si dice anche "soddisfatti col minore/maggiore stretto nella soluzione ottima"
- *saturi*, nel momento in cui abbiamo una variabile di slack pari a 0 (uguale a 0)
 - o Si dice anche "soddisfatti all'uguaglianza nella soluzione ottima"

$s_1 = 0$ significa che la prima variabile di slack è zero e il primo vincolo del PL in forma standard è soddisfatto all'uguaglianza → tecnicamente, il vincolo è saturo

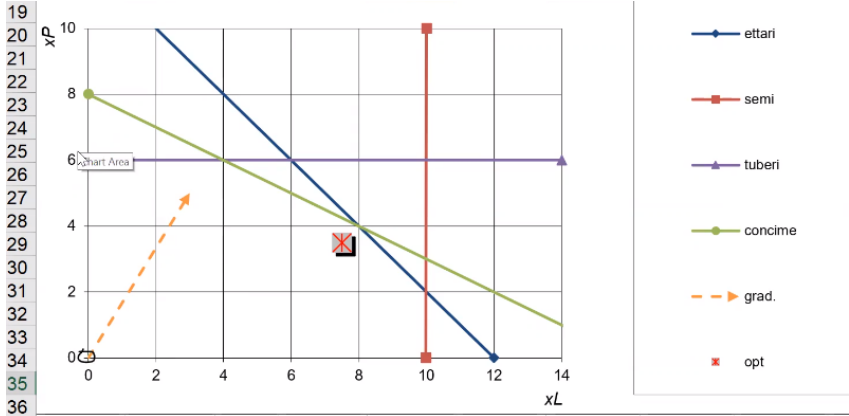
$s_2 = 2$ significa che il secondo vincolo è soddisfatto col minore stretto (basta che abbia un valore *positivo*) → tecnicamente, il vincolo è lasco

$s_3 = 2$ significa che il terzo vincolo è soddisfatto col minore stretto → tecnicamente, il vincolo è lasco

$s_4 = 0$ significa che la quarta variabile di slack è zero e il primo vincolo del PL in forma standard è soddisfatto all'uguaglianza → tecnicamente, il vincolo è saturo

Metodo del simplesso: casi notevoli simplesso e regola anticiclo di Bland

Nel metodo del simplesso, si visita una sequenza di soluzioni adiacenti che corrispondono geometricamente a vertici adiacenti, cercando di migliorare la funzione obiettivo. Rimaniamo grazie alla regola del rapporto minimo. Questo nel problema del contadino è visibile da qui:



Per evitare di fare errori gravi/di calcolo, ad ogni passo ci chiediamo se è ammissibile oppure se è in forma canonica (avere valori diversi da 1 sul pivot e non avrei più la forma canonica rispetto alla base che abbiamo). Nota: se parto da una base ammissibile, il problema deve sempre essere ammissibile.

[cfr. Esercizio 10 dispense “Esercizi risolti simplesso”]

Risolvere il seguente problema di programmazione lineare partendo dalla base $\{x_4, x_5, x_6\}$ oppure dalla base $\{x_1, x_5, x_6\}$.

$$\begin{array}{ll} \max & x_1 + 2x_2 + 2x_3 \\ \text{s.t.} & 2x_1 + x_2 - 2x_3 \geq 1 \\ & x_1 + 2x_3 \leq 2 \\ & -x_1 - 2x_2 - x_3 \leq 3 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

Riscriviamo il problema in forma standard. Siccome $x_2 \leq 0$, si introduce una nuova variabile $\hat{x}_2 = -x_2$ con $\hat{x}_2 \geq 0$. Attenzione che si inverte il segno della f.o. in quanto si passa da $\max$ a $\min$.

$$\begin{array}{ll} \min & -x_1 + 2\hat{x}_2 - 2x_3 \\ \text{s.t.} & 2x_1 - \hat{x}_2 - 2x_3 - x_4 = 1 \\ & x_1 + 2x_3 + x_5 = 2 \\ & -x_1 + 2\hat{x}_2 - x_3 + x_6 = 3 \\ & x_1, \hat{x}_2, x_3, x_4, x_5, x_6 \geq 0 \end{array}$$

Similmente, si cambia segno della variabile x_2 essendo che passa da negativa a positiva e, nei casi con termine noto negativo, si cambia segno a tutta l'uguaglianza. In ultimo, quando ho il segno $\geq$, va inserita una variabile di slack negativa.

Mettiamo sul tableau per vedere cosa succede, scegliendo come base tra le due $[x_4, x_5, x_6]$, che però risulta non ammissibile avendo x_4 negativo.

x_1	$\hat{x}_2$	x_3	x_4	x_5	x_6	z_{min}	b_i
-1	2	-2	0	0	0	-1	0
2	-1	-2	1	0	0	0	1
1	0	2	0	1	0	0	2
-1	2	-1	0	0	1	0	3

$B = [x_4, x_5, x_6]$
 $x_4 = -1 < 0$
 $x_5 = 2$
 $x_6 = 3$

Ecco perché il prof potrebbe dare varie basi di partenza, dato non tutte possono essere ammissibili; infatti, qui si parte da $B = [x_1 \ x_5 \ x_6]$, essendo l'altra non ammissibile. Segnalo le operazioni di pivoting:

Operazioni: $R_1 \leftarrow R_1/2, R_2 \leftarrow R_2 - R_1/2, R_3 \leftarrow R_3 + R_1/2$.

Partiamo dalla prima iterazione, mettendo in forma canonica rispetto a B :

ITER 1 $B = [x_1 \ x_5 \ x_6]$

F.C. - Rispetto a B

$R_0' = R_0 + R_1$	0	3/2	-3	-1/2	0	0	-1	1/2
$R_1' = R_1/2$	1	-1/2	-1	-1/2	0	0	0	1/2
$R_2' = R_2 - R_1/2$	0	1/2	3	1/2	0	0	0	3/2
$R_3' = R_3 + R_1/2$	0	3/2	-2	-1/2	0	1	0	7/2

1) È F.C.? Sì!
 2) È Ammissibile? Sì!
 3) È illimitato? Non so
 4) È ottimo? Non so
 5) Entra? tra x_3 e x_4 , x_3
 6) Esce? argmin $\{ \frac{1}{2}, \frac{3}{2}, \frac{7}{2} \} = \frac{1}{2} = x_5$

- 1) F.C? → Sì, ci sono le colonne della matrice identità
- 2) Ammissibile? → Sì, le variabili della colonna di $\bar{b}$ sono tutte ≥ 0
- 3) Illimitata? Non so → Esiste sempre almeno un coefficiente > 0 per i costi ridotti in corrispondenza di costi ridotti negativi e non posso concludere
- 4) Ottimo? Non so → Esiste qualche costo ridotto < 0 e non posso concludere
- 5) Entra? → Decido tra le variabili di costo ridotto negativo e decido tra x_3 e x_4 e scelgo x_3 (il motivo è spiegato formalmente dopo, ma accontentiamoci qui di dire $x_3 < x_4$)
- 6) Esce? Decido sulla base del rapporto minimo tra $\bar{b}_i$ e x_i , dove x_i è la posizione su cui si è fatti pivoting, quindi x_3 . Facendo i rapporti, il minimo è x_5 con gli altri due che sono negativi e sono scartati.

Andiamo all'iterazione 2, considerando come pivot l'elemento(3) di seconda riga e terza colonna, eseguendo le operazioni di pivoting che seguono:

Operazioni: $R_2 \leftarrow R_2/3, R_1 \leftarrow R_1 + R_2/3, R_3 \leftarrow R_3 + 2/3 R_2, R_0 \leftarrow R_0 + R_2$.

ITER 2 $B = [x_1 \ x_3 \ x_6]$

$R_0' = R_0 + R_2$	0	2	0	0	1	0	-1	2
$R_1' = R_1 + R_2/3$	1	-1/3	0	-1/3	1/3	0	0	1
$R_2' = R_2/3$	0	1/6	1	1/6	1/3	0	0	1/2
$R_3' = R_3 + 2/3 R_2$	0	1/6	0	-1/6	2/3	1	0	3/2

$z_{max} = -z_{min} = -(-2) = 2$
 $x_1^* = 1 \quad x_2^* = 0 \quad x_3^* = 1/2$
 $x_4^* = 0 \quad x_5^* = 0 \quad x_6^* = 9/2$

1) → F.C.? Sì
 2) → Ammissibile? Sì
 3) → illimitato? No
 4) → ottimo? Sì

- 1) È in forma canonica → Sì, ci sono le colonne della matrice identità
- 2) È ammissibile? Sì, le variabili della colonna di $\bar{b}$ sono tutte ≥ 0
- 3) È illimitato → No → Non ci sono colonne con costi ridotti < 0 , pertanto non mi pongo il problema se nella colonna sotto ci siano coefficienti negativi
- 4) È ottima? Sì → Tutti i costi ridotti sono ≥ 0

In merito ai vincoli:

- x_1 all'ottimo vale 1 (colonna di $\bar{B}$)
- x_2 all'ottimo vale 0 (è fuori base)
- x_3 all'ottimo vale 1/2 (colonna di $\bar{B}$)
- x_4 all'ottimo vale 0 (è fuori base)
- x_5 all'ottimo vale 0 (è fuori base)
- x_6 all'ottimo è 9/2 (colonna di $\bar{B}$)

Leggendolo bene;

- $x_4 = 0$ è vincolo saturo, poiché ha valore zero
- $x_5 = 0$ è vincolo saturo poiché ha valore zero
- $x_6 = \frac{9}{2}$ è vincolo lasco, poiché ha valore maggiore di zero

Esistono metodi come quello delle *due fasi*, in cui si scrive un problema artificiale in cui la f.o. è somma di queste variabili artificiali.

- Se il valore della funzione obiettivo è > 0 , si ha un problema inammissibile
- Se il valore della funzione obiettivo, allora $y = 0$, metto tutte le y fuori base e ottengo x_B come soluzione del problema di partenza

In tal caso, risolvo il problema a partire dalla base ottenuta.

Prendiamo un insieme di esempi notevoli del metodo del simplesso.

Prima di tutto, un *problema illimitato*.

- Risolvere con il metodo del simplesso il problema:

$$\begin{aligned} \max \quad & 2x_1 + 5x_2 \\ & x_1 - 4x_2 \leq 8 \\ & -x_1 + x_2 \leq 6 \\ & -3x_1 + 2x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Mettiamo il tableau in forma standard e poi facciamo entrare in base x_2 e poi facendo uscire dalla base il minimo dei rapporti (nello specifico, $5/2$ cioè x_5)

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	-2	-5	0	0	0	-1	0
x_3	1	-4	1	0	0	0	8
x_4	-1	1	0	1	0	0	6
x_5	-3	2	0	0	1	0	5

rapporti $\left\{ \frac{8}{-4}, \frac{6}{1}, \frac{5}{2} \right\}$

Facciamo le operazioni di pivot mettendo in forma canonica rispetto alla nuova base, con x_2 che entra in base e poi faccio pivot su quell'elemento:

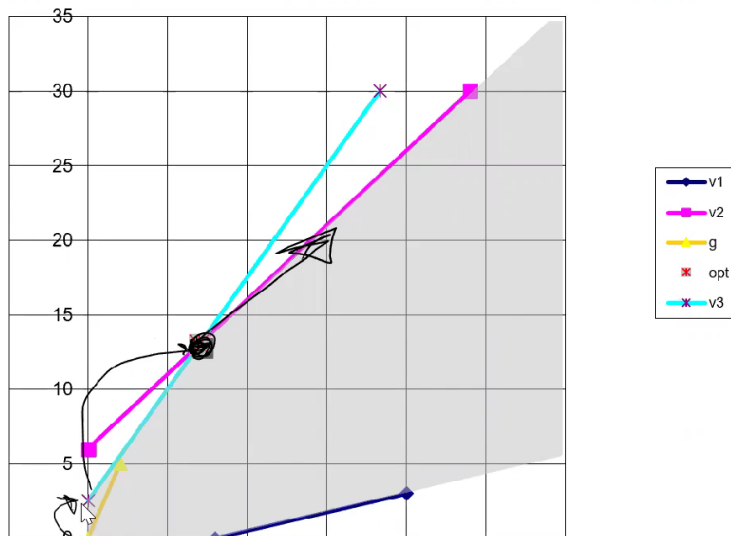
	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	$-19/2$	0	0	0	$5/2$	-1	$25/2$
x_3	-5	0	1	0	2	0	18
x_4	$1/2$	0	0	1	$-1/2$	0	$7/2$
x_2	$-3/2$	1	0	0	$1/2$	0	$5/2$

Ora nel tableau che segue è in forma canonica, ammissibile, non si può dire se ancora ci sta la soluzione ottima, ma si ha una colonna per cui rispetto ad un coefficiente di costo ridotto ci sono solo coefficienti negativi e qui mi fermo, dato che il problema è *illimitato*. Attenzione a non scrivere che il problema è impossibile, semplicemente si è riconosciuta la condizione di illimitatezza.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	0	19	-7	-1	79
x_3	0	0	1	10	-3	0	53
x_1	1	0	0	2	-1	0	7
x_2	0	1	0	3	-1	0	13

problema illimitato

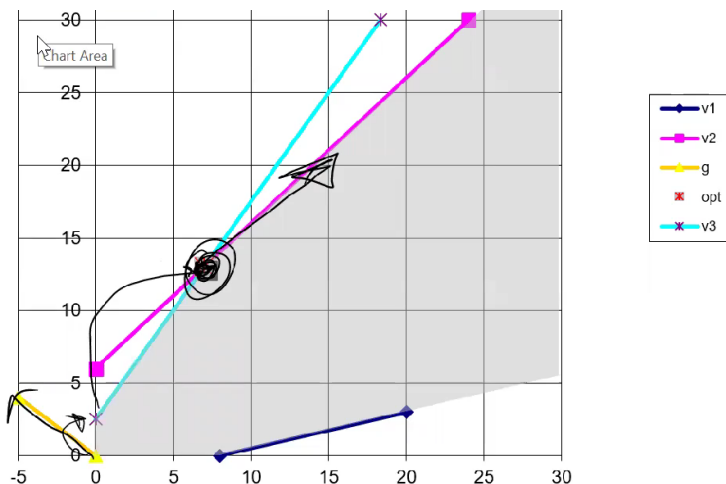
Il problema è come segue nella sua regione ammissibile (quello in grigio). Questa graficamente è illimitata. Per quanto mi allontani dall'origine e dalla funzione obiettivo, rimango sempre una soluzione ammissibile, avendo una direzione in cui aumento ma rimane tutto ammissibile.



Cambiando funzione obiettivo, per cui stesso problema di prima con stessa regione ammissibile ma diversa f.o., come segue qui:

$$\begin{aligned}
 \max \quad & -5x_1 + 4x_2 \\
 x_1 - 4x_2 & \leq 8 \\
 -x_1 + x_2 & \leq 6 \\
 -3x_1 + 2x_2 & \leq 5 \\
 x_1, x_2 & \geq 0
 \end{aligned}$$

Se cambio il verso dell'ottimizzazione, si ha che il punto indicato rimane un vertice ottimo.



I passaggi sono uguali per le basi, ma cambia solo la riga della funzione obiettivo:

Un problema non più illimitato (II)

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$		x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	5	-4	0	0	0	-1	0	$-z$	-1	0	0	0	2	-1	5
x_3	1	-4	1	0	0	0	8	x_3	-5	0	1	0	2	0	18
x_4	-1	1	0	1	0	0	6	x_4	1/2	0	0	1	-1/2	0	7/2
x_5	-3	2	0	0	1	0	5	x_2	-3/2	1	0	0	1/2	0	5/2

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$	
$-z$	0	0	0	2	1	-1	12	ottimo finito
x_3	0	0	1	10	-3	0	53	
x_1	1	0	0	2	-1	0	7	
x_2	0	1	0	3	-1	0	13	

Ci possono essere dei casi che mettono “in crisi” il simplesso, per esempio le cosiddette *soluzioni di base degeneri*, partendo dal seguente problema PL:

$$\begin{aligned}
 \max \quad & 2x_1 + x_2 \\
 & x_1 - x_2 \leq 4 \\
 & 3x_1 - x_2 \leq 12 \\
 & x_1 + x_2 \leq 12 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Le soluzioni di base degeneri significa mettere a zero $n - m$ variabili, con m variabili > 0 . Se anche una sola delle soluzioni di base assume valore 0, allora si dice *degenerare*.

La prima cosa che andiamo a fare è mettere tutto in forma standard, come segue:

- Forma standard:

$$\begin{aligned}
 \min \quad & -2x_1 - x_2 \\
 & x_1 - x_2 + x_3 = 4 \\
 & 3x_1 - x_2 + x_4 = 12 \\
 & x_1 + x_2 + x_5 = 12 \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0
 \end{aligned}$$

Poi, il tableau con la scelta delle variabili di base:

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	-2	-1	0	0	0	-1	0
x_3	1	-1	1	0	0	0	4
x_4	3	-1	0	1	0	0	12
x_5	1	1	0	0	1	0	12

Base 1

x_1	x_2	x_3	x_4	x_5
0	0	4	12	12

Al momento, la situazione è quindi la seguente:

$$x_B = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad x_F = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad B = [A_3 \ A_4 \ A_5] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

con soluzione (le variabili in base sono riquadrate):

$$x = x_1 \ x_2 \ \begin{bmatrix} x_3 & x_4 & x_5 \end{bmatrix} = 0 \ 0 \ \begin{bmatrix} 4 & 12 & 12 \end{bmatrix}$$

Se facciamo entrare in base x_1 , la regola de quoziente minimo individua due righe in corrispondenza del minimo rapporto pari a 4, in particolare x_3 oppure x_4 . Quindi, dobbiamo sceglierne una tra queste due.

argmin $\{ \frac{4}{1}, \frac{12}{3}, \frac{12}{1} \}$
 arg $\{4\} = x_3 \text{ o } x_4$
 Base 1

Non posso fare uscire entrambe le variabili, ma devo sostituirne almeno una, tale che una assuma valore 0 e l'altra assume sempre valore 0 ma resta in base (ma risulterà così *degenere*).

Man mano che x_1 cresce, alcune variabili diminuiscono con valori e velocità diverse, arrivando entrambe a valore zero, come si vede qui

esce ARG $\{4\} = x_3$ oppure x_4

$$x_4 = 12 - 3x_1$$

$$x_3 = 4 - x_1$$

$$x_1 = 4$$

$$x_4 = 0$$

$$x_3 = 0$$

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	-3	2	0	0	-1	8
x_1	1	-1	1	0	0	0	4
x_4	0	2	-3	1	0	0	0
x_5	0	2	-1	0	1	0	8

x_1 entra quindi in base e risulta così il nuovo tableau. Similmente, x_3 (che assume valore zero) esce dalla base, mentre x_4 (che pure assume valore zero) rimane in base. Quindi, come si vede, sia x_3 che x_4 vanno a zero.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	-3	2	0	0	-1	8
x_1	1	-1	1	0	0	0	4
x_4	0	2	-3	1	0	0	0
x_5	0	2	-1	0	1	0	8

Base 2

	x_1	x_2	x_3	x_4	x_5
	4	0	0	0	8

La situazione è quindi la seguente:

$$x_B = \begin{bmatrix} x_1 \\ x_4 \\ x_5 \end{bmatrix} \quad x_F = \begin{bmatrix} x_3 \\ x_2 \end{bmatrix}$$

$$B = [A_1 \ A_4 \ A_5] = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

con soluzione (le variabili in base sono riquadrate):

$$x = \boxed{x_1} \ x_2 \ x_3 \ \boxed{x_4} \ \boxed{x_5} = \boxed{4} \ 0 \ 0 \ \boxed{0} \ \boxed{8}$$

Ora, però, abbiamo la ricerca del rapporto minimo; in questo caso specifico, avremo che il rapporto minimo considera la scelta del numero 0. A questo punto scegliamo x_2 come variabile entrante e, di conseguenza, la riga 2 (variabile x_4) come variabile uscente: corrisponde al minimo rapporto che è 0 (si ricorda che non si devono considerare i rapporti con denominatore minore o uguale a 0). Ciò vuol dire che x_2 entrerà in base al valore 0 e dovrà rimanere a 0 (*iterazione degenere*).

Passaggio per basi degeneri (III) *arg min* $\left\{ \frac{4}{7}, \frac{0}{2}, \frac{8}{2} \right\}$
arg (Base 2) $\{0\} = x_4$

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	-3	2	0	0	-1	8
x_1	1	-1	1	0	0	0	4
x_4	0	2	-3	1	0	0	0
x_5	0	2	-1	0	1	0	8

x_1	x_2	x_3	x_4	x_5
-------	-------	-------	-------	-------

4	0	0	0	8
---	---	---	---	---

x_2 entra quindi al valore zero e poi faccio altre operazioni di pivot, lasciando l'ultima colonna invariata, dato che sto facendo entrare una variabile di base invariata e otteniamo la stessa soluzione (con uno stesso punto nello spazio) rappresentandolo con basi diverse.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	-5/2	3/2	0	-1	8
x_1	1	0	-1/2	1/2	0	0	4
x_2	0	1	-3/2	1/2	0	0	0
x_5	0	0	2	-1	1	0	8

Base 3

x_1	x_2	x_3	x_4	x_5
-------	-------	-------	-------	-------

4	0	0	0	8
---	---	---	---	---

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	0	1/4	5/4	-1	18
x_1	1	0	0	1/4	1/4	0	6
x_2	0	1	0	-1/4	3/4	0	6
x_3	0	0	1	-1/2	1/2	0	4

Base 4

x_1	x_2	x_3	x_4	x_5
-------	-------	-------	-------	-------

6	6	4	0	0
---	---	---	---	---

che corrisponde alla soluzione ottima:

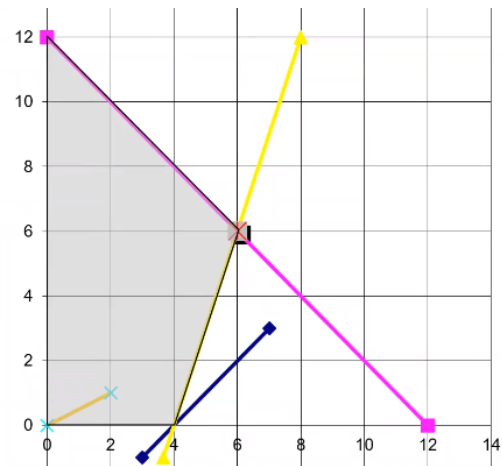
$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad x_F = \begin{bmatrix} x_5 \\ x_4 \end{bmatrix}$$

$$B = [A_1 \ A_2 \ A_3] = \begin{bmatrix} 1 & -1 & 1 \\ 3 & -1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

con soluzione (le variabili in base sono riquadrate):

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} x_4 & x_5 \end{bmatrix} = \begin{bmatrix} 6 & 6 & 4 & 0 & 0 \end{bmatrix}$$

Il passaggio per soluzioni di base degeneri ha un'interessante interpretazione grafica. Vedendo la regione ammissibile, stiamo considerando la rappresentazione dello stesso vertice ottimo usando due basi diverse (infatti, nel primo e nel secondo passo del simplesso, il vertice rimane sempre $(4,0)$); quindi, qui il vertice adiacente è sé stesso.



Ora, altro esempio, caso di *infinite soluzioni ottime*, da risolvere sempre con il metodo del simplesso:

$$\begin{aligned} \max \quad & 3x_1 + 2x_2 \\ & x_2 \leq 7 \\ & 2x_1 + 1/2x_2 \leq 10 \\ & 3/2x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Mettiamo in forma standard e notiamo subito la base ammissibile data dalle variabili di slack x_3, x_4, x_5 :

• Forma standard:

$$\begin{aligned} \min \quad & -3x_1 - 2x_2 \\ & x_2 + x_3 = 7 \\ & 2x_1 + 1/2x_2 + x_4 = 10 \\ & 3/2x_1 + x_2 + x_5 = 10 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Entra quindi in base x_1 e poi esce x_4 :

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	-3	-2	0	0	0	-1	0
x_3	0	1	1	0	0	0	7
x_4	2	1/2	0	1	0	0	10
x_5	3/2	1	0	0	1	0	10

Successivamente entra in base x_2 ed esce dalla base x_5 :

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	-5/4	0	3/2	0	-1	15
x_3	0	1	1	0	0	0	7
x_1	1	1/4	0	1/2	0	0	5
x_5	0	5/8	0	-3/4	1	0	5/2

Alla fine, si ottiene come tableau il successivo:

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	0	0	2	-1	20
x_3	0	0	1	6/5	-8/5	0	3
x_1	1	0	0	4/5	-2/5	0	4
x_2	0	1	0	-6/5	8/5	0	4

FINE!

ma...

che corrisponde alla soluzione ottima (costi ridotti ≥ 0)

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 3 \\ 0 \\ 0 \end{bmatrix} \quad z_{MIN} = -20 \quad (z_{MAX} = 20)$$

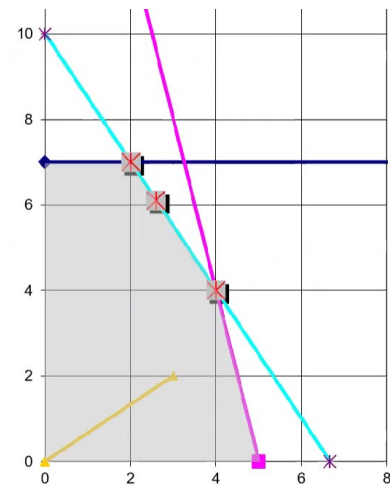
La particolarità della soluzione deriva dall'aver il costo ridotto di una variabile fuori base pari a $\bar{c}_{x_4} = 0$. Ciò vuol dire che, se facessi entrare in base la variabile x_4 la funzione obiettivo rimarrebbe invariata. Anche se l'algoritmo del simplesso ha terminato l'esecuzione (è raggiunta una condizione di terminazione), proviamo a effettuare un'operazione di pivot per far entrare in base x_4 .

La regola del quoziente minimo indica come variabile uscente la variabile x_3 e le operazioni di pivot portano al tableau:

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$	
$-z$	0	0	0	0	2	-1	20	
x_4	0	0	5/6	1	-4/3	0	5/2	altra
x_1	1	0	-2/3	0	-2/3	0	2	soluzione
x_2	0	1	1	0	0	0	7	ottima!

Ho quindi due soluzioni di base che hanno lo stesso valore della funzione obiettivo, avendo che la direzione esistono due vertici ottimi distinti, corrispondenti alle due soluzioni ottime di base trovate. Se consideriamo la direzione del gradiente, possiamo facilmente osservare che tutti i punti del segmento che congiunge i due vertici ottimi sono soluzioni ammissibili e ottime. Abbiamo quindi infinite soluzioni ottime, anche se non di base (perché non sono vertici).

Quindi: se dovessi avere variabili fuori base con costo ridotto pari a zero, potrei trovarmi in una situazione in cui ottengo lo stesso valore della funzione obiettivo anche per diverse variabili.



Altro caso critico, soluzioni ottime con costi ridotti negativi.

- Risolvere con il metodo del simplesso (a partire dalla base x_1, x_2, x_4):

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1 + x_2 \geq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Mettendo in forma standard, ma la soluzione formata dalle variabili di slack è soluzione di base (perché formata da colonne linearmente indipendenti), ma non è ammissibile.

Ecco perché la base viene data dal prof.

Soluzioni ottime con costi ridotti negativi (I)

- Risolvere con il metodo del simplesso (a partire dalla base x_1, x_2, x_4):

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1 + x_2 \geq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- Forma standard:

$$\begin{aligned} \min \quad & -x_1 - 2x_2 \\ & x_1 + x_3 = 1 \\ & x_2 + x_4 = 1 \\ & x_1 + x_2 - x_5 = 2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

$x_5 = -2$

Siccome non siamo in forma canonica, mettiamo delle sbarrette a lato del tableau e facciamo il pivot sugli elementi cerchiati.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
//	-1	-2	0	0	0	-1	0
//	1	0	1	0	0	0	1
//	0	1	0	1	0	0	1
//	1	1	0	0	-1	0	2

Errori:

- far comparire la matrice identità moltiplicando sul (-1)
- la moltiplicazione per (-1) mette (-2) sull'ultima colonna e fa dire che il problema sia impossibile, ma non è così

In corrispondenza della soluzione di base, la funzione obiettivo vale -3 , ma non si sa se sia ottima, avendo almeno un coefficiente di costo ridotto negativo.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	-1	0	-2	-1	3
x_1	1	0	1	0	0	0	1
x_2	0	1	-1	0	-1	0	1
x_4	0	0	1	1	1	0	0

Andando a fare pivoting, faccio un'iterazione degenera (non cambiando vertice) e mi rimane il vecchio valore della funzione obiettivo.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	0	1	-1	-1	3
x_1	1	0	0	-1	-1	0	1
x_2	0	1	0	1	0	0	1
x_3	0	0	1	1	1	0	0

$z = -3$
non si

Facendo un'altra iterazione degenera, si ottiene una base con tutti costi ridotti ≥ 0 e mi fermo.

	x_1	x_2	x_3	x_4	x_5	z	$\bar{b}$
$-z$	0	0	-1	0	-2	-1	3
x_1	1	0	1	0	0	0	1
x_2	0	1	-1	0	-1	0	1
x_4	0	0	1	1	1	0	0

Partendo da soluzioni di base degeneri, ci sono dei problemi:

- 1) Il simplesso potrebbe stare fermo in un punto (senza migliorare il valore della f.o.)
- 2) Ci sono più soluzioni ottime, magari con costi ridotti negativi (dubbi sulla convergenza del simplesso)

Per valutare la convergenza e la complessità del metodo del simplesso, consideriamo il valore di θ , che è il rapporto minimo. Si struttura una serie di passi:

- Partiamo dal valore della funzione obiettivo, in cui si cambia base e si trova θ come valore del rapporto minimo:

- Ad ogni iterazione: $z_{new} = z_{old} + c_h \theta$

$$\theta = \min_{i=1 \dots m} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$$

- Il simplesso, se $\theta > 0$ fosse maggiore di zero, alla peggio visita tutti i vertici, ma sono sicuro che converga

- Se $\theta > 0$ ad ogni iterazione:

cambio base e vertice, $z_{new} < z_{old}$

$$\Rightarrow \text{convergenza in } O\left(\binom{n}{m}\right)$$

- Se invece $\theta = 0$, abbiamo basi degeneri, cioè, potrei tornare sempre a visitare lo stesso vertice sulla stessa base, non avendo mai garanzia di convergenza.

- Se $\theta = 0$ (passaggio tra basi degeneri!):

cambio base ma non vertice, $z_{new} = z_{old}$

potrei tornare alla stessa base

$\Rightarrow$ **nessuna garanzia di convergenza!**

Ad esempio, si potrebbe considerare una politica di *cycle detection*: qualora si incontrasse una base già visitata, si applicano delle regole alternative per la scelta delle variabili per il cambio base, sperando di non tornare sulle stesse basi.

Una possibile regola anti-ciclo sistematica, che citiamo ed usiamo per la sua semplicità, è la regola anticiclo di Bland:

- Fissare un ordine tra le variabili (indice) a priori, per esempio seguendo la numerazione
- Tra tutte le variabili candidate al cambio base, scegliere sempre la prima (quella con indice minimo)

Pertanto:

- in caso di più variabili attualmente fuori base con costo ridotto negativo, entra in base la variabile x_h con indice h minimo
 $x_h: h = \min\{j: \bar{c}_j < 0\}$
- in caso di più variabili attualmente in base che corrispondono al minimo quoziente θ , si sceglie la variabile x_t con indice minimo:

$$x_t: t = \min\{B_i: \bar{b}_i/\bar{a}_{ih} = \theta\}$$

Ad esempio, consideriamo il seguente tableau:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\bar{b}$
$-z$	5	-1	0	-10	0	0	0	0	-10
x_5	1	4	0	1	1	0	0	0	8
x_3	-1	3	1	0	0	0	0	0	6
x_6	0	-2	0	3	0	1	0	0	1
x_8	3	2	0	4	0	0	0	1	5
x_7	3	1	0	-2	0	0	1	0	2

- 1) soluzione base? Devo capire se ci sta una matrice identità, cosa che capita; infatti, ci sono 5 colonne di matrice identità (base ammissibile inoltre, con numeri non negativi)
- 2) è ottima? Non si sa (almeno un costo ridotto negativo)
- 3) possibili cambi base? $\rightarrow$ entra x_2 , esce $\arg \min\{8/4, 6/3, 1/(-2), 5/2, 2/1\} = \arg\{2\} = x_3, x_5, x_7$ (uno tra questi)
- 4) con regola di Bland? $\rightarrow$ Scelgo x_3 perché viene prima per ordine rispetto a x_5, x_7
- 5) perché otterremmo soluzione degenerare? $\rightarrow$ Se applico Bland, ho più valori che corrispondono a più rapporti, ma gli altri rapporti minimi evidenziano variabili che vanno a 0 rimanendo in base (a questa si risponde dando argomentazioni teoriche)

Costi ridotti non negativi è condizione sufficiente di ottimalità; siamo sicuri di raggiungerla?

Sì: se esiste una soluzione ottima, allora esiste una base ottima con costi ridotti tutti non negativi. Quindi:

Teorema: convergenza del simplesso con regola anticiclo di Bland

Utilizzando la regola di Bland, il metodo del simplesso converge sempre in al più $\binom{n}{m}$ iterazioni

Il simplesso è mediamente più efficiente, con complessità media lineare ma anche sublineare in alcuni casi.

Laboratorio 3 - Risoluzione di vari modelli con AMPL

Andiamo a risolvere il problema dei telecomandi con AMPL, usando il modello generale. Consideriamo, però, che ogni prodotto ha uno specifico limite superiore.

Ricordiamo il problema:

1. Un gioco di assemblaggio

Per l'assemblaggio di telecomandi, si hanno a disposizione 10 moduli display, 18 moduli di logica di controllo, 12 trasmettitori, 21 tastierini, 9 moduli di navigazione e 10 led. I telecomandi sono di due tipi. Il tipo A richiede un display, un modulo di navigazione, 2 tastierini, 2 moduli di logica, un trasmettitore e un led. Il tipo B richiede 2 display, 3 tastierini, 2 moduli di logica e 3 trasmettitori. Considerando che il tipo A permette un guadagno netto di 3 euro e il tipo B di 8 euro, determinare la produzione che massimizza il guadagno.

$$\begin{aligned}
 \max \quad & 3x_A + 8x_B \quad (\text{guadagno complessivo}) \\
 \text{s.t.} \quad & x_A + 2x_B \leq 10 \quad (\text{display}) \\
 & x_A \leq 9 \quad (\text{navigazione}) \\
 & 2x_A + 3x_B \leq 21 \quad (\text{tastierini}) \\
 & 2x_A + 2x_B \leq 18 \quad (\text{logica}) \\
 & x_A + 3x_B \leq 12 \quad (\text{trasmissione}) \\
 & x_A \leq 10 \quad (\text{led}) \\
 & x_A, x_B \in \mathbb{Z}_+ \quad (\text{dominio})
 \end{aligned}$$

Il modello generale è presente in un file chiamato *mixOpt.mod*, in cui andiamo a modellare esattamente il discorso citato, quindi il limite superiore diverso per ogni prodotto indicizzato.

Il file *mixOpt_telecomandi.mod* segue nella sua forma codice.

```

set I; #prodotti
set J; #risorse

param maxNumProd;
param P{I};
param Q{J};
param A{I,J};

var x {i in I} >= 0 <= maxNumProd[i] integer;

maximize profitto: sum{i in I} P[i]*x[i];
s.t. dispon{j in J}: sum{i in I} A[i, j]*x[i]

```

Sinteticamente:

- dati i tipi A e B di telecomandi e tutte le categorie di risorse, definisco parametri di prodotti e risorse (con i loro insiemi di riferimento che, riferiti a *mixOpt.mod* si dovranno chiamare *I* e *J*, in quanto si chiamavano così in quel file e simile ad altri parametri)
- per i prodotti, dati i due tipi di telefoni e il loro costo, inseriamo due valori qualsiasi per impostare il calcolo della funzione obiettivo, come si vede ad es. $P = 3$ per *telA*, $P = 6$ per *telB*
- per le risorse, definiamo tutte le quantità dei singoli tipi di telecomandi
- infine, definiamo per tutti i telecomandi di A e di B tutti i sottotipi *j*

```
#insiemi
set I := telA telB;
set J := modisplay navigazione tastierini logica tx led;

#parametri
param P := telA 3 telB 6;
param maxNumProd := telA 5 telB 8;

param          Q :=
modisplay      10
navigazione    9
tastierini     21
logica         18
tx             12
led            10
;

param A:      modisplay navigazione tastierini logica tx led :=
telA          1          1          2          2          1  1
telB          2          0          3          2          3  0
;
```

Eseguiamo il tutto nel file *mixOpt_telec.run*:

```
reset;
model mixOpt.mod;
data mixOpt_telecomandi.dat;
option solver cplex;
solve;
display profitto, x;
```

Per dichiarare sinteticamente i parametri ci sono varie strade, facendo in modo di indicizzare i singoli parametri e comprendere anche graficamente la disposizione e la rispettiva assegnazione. Ciò può essere in forma a più indici, tabellare (disposta con ordine anche tabulando correttamente i parametri) o anche in forma trasposta (tale da mettere sulle colonne il primo indice e sulle righe il secondo indice).

■ Dichiarazioni sintetiche: più parametri con stessi indici

```
param : P MaxNumProd := #notare i due punti dopo "param"
      telA 3 5
      telB 6 999;
```

■ Dichiarazioni sintetiche: tabelle

```
param A : ettari semi tuberi concime := #due punti dopo A
lattuga  1      7      0      10
patata   1      0      3      20;
```

■ Dichiarazioni sintetiche: tabelle trasposte

```
param A (tr) : lattuga patata :=
ettari      1      1
semi        7      0
tuberi      0      3
Concime     10     20;
```

Dato che AMPL vuole un valore per tutti i parametri, possiamo assegnare ad un parametro *param* il valore *default*, in modo tale che capisca sempre quanto vale in ogni momento e, se non viene dichiarato un valore, viene assegnato il valore che segue *default*. Ad esempio:

```
param maxNumProd{I} default 99999;
```

Ora consideriamo il problema della dieta, che riportiamo qui (assieme al modello generale):

Un dietologo deve preparare una dieta che garantisca un apporto giornaliero di proteine, ferro e calcio di almeno 20 mg, 30 mg e 10 mg, rispettivamente. Il dietologo è orientato su cibi a base di verdura (5 mg/kg di proteine, 6 mg/Kg di ferro e 5 mg/Kg di calcio, al costo di 4 €/Kg), carne (15 mg/kg di proteine, 10 mg/Kg di ferro e 3 mg/Kg di calcio, al costo di 10 €/Kg) e frutta (4 mg/kg di proteine, 5 mg/Kg di ferro e 12 mg/Kg di calcio, al costo di 7 €/Kg). Determinare la dieta di costo minimo.

Risolvere il problema con AMPL (file .mod e .dat separati)

Il file *dieta.mod* contiene come codice:

```
set I; #risorse
set J; #domande

param C{I}; #costo risorse
param D{J}; #domande
param A{I,J}; #capacità risorsa "i" per
soddisfare la domanda "j"

var x{I} >= 0 integer;

minimize costo: sum{i in I} C[i]*x[i];

s.t. soddisfazione{j in J}: sum{i in I}
A[i,j] * x[i] >= D[j];
```

Il file *dieta.dat* contiene come codice:

```
#insiemi
set I := 1ver 2car 3fru; #insieme delle risorse
set J := pro fer cal; #insieme delle domande

#parametri

#costi indicizzati dall'indice "i"
param C := 1ver 4      2car 10      3fru 7;

#domande indicizzate dall'indice "j"
param D := pro 20      fer 30      cal 10;

#tabella con colonne indicizzate da "i"
#e righe indicizzate da "j"
#e quindi si scrive trasposta con (tr)

param A (tr) :      1ver      2car      3fru :=
pro      5      15      4
fer      6      10      5
cal      5      3      12
;
```

Modello generale: dieta

I insieme delle risorse disponibili;

J insieme delle domande da coprire;

C_i costo (unitario) per l'utilizzo della risorsa $i \in I$;

D_j ammontare della domanda di $j \in J$;

A_{ij} capacità (unitaria) della risorsa i di soddisfare la domanda j .

$$\begin{aligned} \min \quad & \sum_{i \in I} C_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \geq D_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0,1\}] \quad \forall i \in I \end{aligned}$$

Modello PL

■ Siano x_1, x_2 e x_3 le quantità di cibi a base di verdura, carne e frutta, rispettivamente

$\min \quad 4x_1 + 10x_2 + 7x_3 \quad (\text{costo giornaliero dieta})$

s.t.

$5x_1 + 15x_2 + 4x_3 \geq 20 \quad (\text{proteine})$

$6x_1 + 10x_2 + 5x_3 \geq 30 \quad (\text{ferro})$

$5x_1 + 3x_2 + 12x_3 \geq 10 \quad (\text{calcio})$

$x_i \in \mathbb{R}_+, \forall i \in \{1, 2, 3\}$

In merito al file *diet.run* contiene:

```
reset;
model dieta.mod;
data dieta.dat;
option solver cplex;
solve;
display costo, x;
```

Dando come risultato:

```
ampl: include diet.run;
CPLEX 20.1.0.0: optimal integer solution; objective 20
1 MIP simplex iterations
0 branch-and-bound nodes
costo = 20
```

Esercizio 5.

Il dietologo vuole inserire alimenti a base di pesce azzurro (10 mg/kg di proteine, 15 mg/kg di ferro e 2 mg/kg di calcio, al costo di 3 euro/kg) nella dieta.

Modificare opportunamente i file relativi al problema.

Quindi, listiamo tutti i file, prendendo il file *diet2.mod*:

```
set I; #risorse
set J; #domande

param C{I}; #costo risorse
param D{J}; #domande
param A{I,J};
param MinRisorsa{I} default 0; #parametro indicizzato nell'insieme I
#messo a default a 0 (perché una, la risorsa minima, ha valore 3
#mentre tutte le altre a 0)

var x{i in I} integer >= MinRisorsa[i];

minimize costo: sum{i in I} C[i]*x[i];
s.t. soddisfazione{j in J}: sum{i in I} A[i,j] * x[i] >= D[j];
```

Segue il file *diet2.dat* (contiene un parametro lower bound/limite inferiore sulle cose da comprare, chiamato *MinRisorsa*, inserito nel file *.dat*, parametro che dipende dalla singola risorsa ed è indicizzato dall'indice *i*); dato che solo per il pesce azzurro ho un acquisto minimo di 3 elementi, allora nel file *.dat* che vado a definire metto *default=0*.

```
#insiemi
set I := 1ver 2car 3fru 4azz; #insieme delle risorse
set J := pro fer cal; #insieme delle domande

#parametri

#costi indicizzati dall'indice "i"
param C := 1ver 4          2car 10          3fru 7          4azz 3;

#domande indicizzate dall'indice "j"
param D := pro 20          fer 30          cal 10;

param MinRisorsa := #solo per il pesce azzurro ho un .bound di almeno 3
4azz 3;
```

Scritto da Gabriel

```
#tabella con colonne indicizzate da "i"
#e righe indicizzate da "j"
#e quindi si scrive trasposta con (tr)
```

```
param A (tr) :          1ver          2car          3fru          4azz :=
pro                5          15          4          10
fer                6          10          5          15
cal                5          3          12          2
;
```

E il file *diet2.run*:

```
reset;
model diet2.mod;
data diet2.dat;

option solver cplex;
solve;

display costo, x;
```

Ad esecuzione, poi:

```
ampl: include diet2.run;
CPLEX 20.1.0.0: optimal integer solution; objective 13
0 MIP simplex iterations
0 branch-and-bound nodes
costo = 13

x [*] :=
1ver 1
2car 0
3fru 0
4azz 3
;
```

Segue il problema dell'indagine di mercato (in cui, ancora una volta, si adotta il modello generale della dieta, riportato da me qui per comodità visiva)

Esercizio 6. Indagine di mercato

Un'azienda pubblicitaria deve svolgere un'indagine di mercato per lanciare un nuovo prodotto. Si deve contattare telefonicamente un campione significativo di persone: almeno 150 donne sposate, almeno 110 donne non sposate, almeno 120 uomini sposati e almeno 100 uomini non sposati. Le telefonate possono essere effettuate al mattino (al costo operativo di 1.1 euro) o alla sera (al costo di 1.6 euro). Le percentuali di persone mediamente raggiunte sono riportate in tabella.

	Mattino	Sera
Donne sposate	30%	30%
Donne non sposate	10%	20%
Uomini sposati	10%	30%
Uomini non sposati	10%	15%
Nessuno	40%	5%

Si noti come le telefonate serali sono più costose, ma permettono di raggiungere un maggior numero di persone: solo il 5% va a vuoto. Si vuole minimizzare il costo complessivo delle telefonate da effettuare (mattina/sera) in modo da raggiungere un campione significativo di persone

Risolvere il problema con AMPL (usare soluzione Es. 4)

Modello PLI

- Siano x_1 e x_2 il numero di telefonate da fare al mattino e alla sera, rispettivamente

$$\min \quad 1.1 x_1 + 1.6 x_2 \text{ (costo totale telefonate)}$$

s.t.

$$\begin{aligned} 0.3x_1 + 0.3x_2 &\geq 150 && \text{(donne sposate)} \\ 0.1x_1 + 0.2x_2 &\geq 110 && \text{(donne non sposate)} \\ 0.1x_1 + 0.3x_2 &\geq 120 && \text{(uomini sposati)} \\ 0.1x_1 + 0.15x_2 &\geq 100 && \text{(uomini non sposati)} \end{aligned}$$

$$x_i \in \mathbb{Z}_+, \forall i \in \{1, 2\} \quad [\text{Risorse}] \text{ diet.indagine.dat}$$

Nel file *indagine.dat* si dettaglia la seguente struttura per i singoli dati:

```
#insiemi
set I := 1mattino 2sera; #risorse: telefonate al mattino e alla sera
set J := ds dn us uc; #domande: uomini/donne sposati-e/non sposati-e

#parametri

param C := 1mattino 1.1 2sera 1.6; #costo mattino-sera
param D := ds 150 dn 110 us 120 uc 100; #domande uomini/donne sp/nsp

param A (tr):          1mattino          2sera := #tab. come da slide
ds                      0.3              0.3
dn                      0.1              0.2
us                      0.1              0.3
uc                      0.1              0.15
```

Segue nel dettaglio il file *indagine.run*, che semplicemente usa il modello generale della dieta, come detto, e visualizza le variabili e il costo generale:

```
reset;
model dieta.mod;
data indagine.dat;

option solver cplex;
solve;
display costo, x;
```

Esempio: Localizzazione di servizi

Una città è divisa in sei quartieri, dove si vogliono attivare dei centri unificati di prenotazione (CUP) per servizi sanitari. In ciascun quartiere è stata individuata una possibile località di apertura. Le distanze medie in minuti da ciascun quartiere a ciascuna delle possibili località è indicata in tabella. Si desidera che nessun utente abbia un tempo medio di spostamento superiore a 15 minuti per arrivare al CUP più vicino e si vuole minimizzare il numero di CUP attivati.

	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5

Modello generale: dieta

I insieme delle risorse disponibili;

J insieme delle domande da coprire;

C_i costo (unitario) per l'utilizzo della risorsa $i \in I$;

D_j ammontare della domanda di $j \in J$;

A_{ij} capacità (unitaria) della risorsa i di soddisfare la domanda j .

$$\begin{aligned} \min \quad & \sum_{i \in I} C_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \geq D_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I \end{aligned}$$

Di nuovo, viene consigliato di usare il modello della dieta, seguendo queste regole:

Sia $x_i = 1$, se viene aperto il CUP nel quartiere i , 0 altrimenti

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

s.t.

$$x_1 + x_2 \geq 1 \text{ (esigenze q.re 1)}$$

$$x_1 + x_2 + x_6 \geq 1 \text{ (esigenze q.re 2)}$$

$$x_3 + x_4 \geq 1 \text{ (esigenze q.re 3)}$$

$$x_3 + x_4 + x_5 \geq 1 \text{ (esigenze q.re 4)}$$

$$x_4 + x_5 + x_6 \geq 1 \text{ (esigenze q.re 5)}$$

$$x_2 + x_5 + x_6 \geq 1 \text{ (esigenze q.re 6)}$$

$$x_i \in \{0,1\}$$

- Come modello di copertura («dieta»): richiede data preprocessing ([Risorse]: diet.mod - diet.CUP.dat)
- Modello specifico a partire dai dati «grezzi»:

I : insieme dei quartieri
 c_{ij} : distanza da $i \in I$ a $j \in I$
 T : distanza soglia

$$\min \sum_{i \in I} x_i$$

$$\text{s.t.} \quad \sum_{\substack{j \in I: \\ c_{ij} \leq T}} x_j \geq 1, \forall i \in I$$

$$x_i \in \{0,1\}, \forall i \in I$$

Abbiamo per il file *diet.CUP.dat* (ricordiamo che mettiamo ad 1 nella matrice tutte le risorse con tempo ≤ 15 minuti di spostamento, nel caso decidessimo di andare in quella località per quel quartiere).

Tutto il resto dà un costo unitario alla località associato alla presenza di una domanda per spostarsi nello specifico quartiere.

```
#insiemi
set I := loc1 loc2 loc3 loc4 loc5 loc6;
set J := qre1 qre2 qre3 qre4 qre5 qre6;

#parametri
param C := loc1 1 loc2 1 loc3 1 loc4 1 loc5 1 loc6 1;

param D := qre1 1 qre2 1 qre3 1 qre4 1 qre5 1 qre6 1;

param A (tr):
loc1    loc2    loc3    loc4    loc5    loc6 :=
qre1    1      1      0      0      0      0
qre2    1      1      0      0      0      1
qre3    0      0      1      1      0      0
qre4    0      0      1      1      0      0
qre5    0      0      0      1      1      0
qre6    0      1      0      0      1      1
;
```

Il file *diet.cup.run* è come segue:

```
reset;
model dieta.mod;
data diet.CUP.dat;
option solver xpress;
solve;
display costo, x;
```

Tutto quanto eseguito dà questo risultato:

```
ampl: include diet.CUP.run;
XPRESS 8.13.1(39.01.02): Global search complete
Best integer solution found 2
1 branch and bound node
No basis.
costo = 2

x [*] :=
loc1 0
loc2 1
```

```
loc3 0
loc4 1
loc5 0
loc6 0
;
```

Laboratorio 4 - Ulteriori modelli di PL ed esempi di esercizi AMPL da esame

Ripartiamo esplicitamente dal modello che segue:

- Come modello di copertura («dieta»): richiede data preprocessing ([Risorse]: `diet.mod - diet.CUP.dat`)
- Modello specifico a partire dai dati «grezzi»:

I: insieme dei quartieri
c_{ij}: distanza da i ∈ I a j ∈ I
T: distanza soglia

$$\min \sum_{i \in I} x_i$$

$$\text{nt. } \sum_{\substack{j \in I: \\ c_{ij} \leq T}} x_j \geq 1, \forall i \in I$$

$$x_i \in \{0,1\}, \forall i \in I$$

Noi vorremmo minimizzare il costo realizzando con AMPL la minimizzazione delle distanze. Andiamo ad implementare il modello ricalcando la sintassi nel file `cuptempi.mod`.

```
set I; #quartieri
param C{I,I}; #costi
param T; #soglia
var x{I} binary;
minimize costo: sum {i in I}
x{i};
subject to soddisfa{i in I}:
    sum{j in I : C[i, j] <= T}
x[j] >= 1;
```

Noi vogliamo una somma in cui compaiono elementi indicizzati nell'insieme I implementando il vincolo, dicendo "facciamo una somma alcuni elementi dell'insieme".

Questo capita dentro il vincolo *soddisfa*, in cui la distanza $C[i, j]$ deve stare dentro la somma.

Andiamo a creare i dati dentro `cuptempi.dat`:

```
set I := q1 q2 q3 q4 q5 q6; #quartieri
param C :
q1      q1      q2      q3      q4      q5      q6      :=
q1      5       10      20      30      30      20
q2      10      5       25      35      20      10
q3      20      25      5       15      30      20
q4      30      35      15      5       15      25
q5      30      20      30      15      5       14
q6      20      10      20      25      14      5
;
```

```
param T := 15; #soglia
```

Qui poi il relativo `cuptempi.run`:

```
set I := q1 q2 q3 q4 q5 q6; #quartieri
param C :
q1      q1      q2      q3      q4      q5      q6      :=
q1      5       10      20      30      30      20
q2      10      5       25      35      20      10
q3      20      25      5       15      30      20
q4      30      35      15      5       15      25
q5      30      20      30      15      5       14
q6      20      10      20      25      14      5
;
```

```
param T := 15; #soglia
```

Facciamo un riassunto delle *espressioni indicizzanti*, sapendo che utilizziamo *insiemi*.

Sintassi {...} per definire **indici** di variabili, parametri, vincoli, sommatorie, comandi etc. Utilizzano **insiemi**

... precedentemente dichiarati: {A} {I}

... multidimensionali: {A,B} {I,J} {I,I} {A,B,I}

... calcolati:

{A cross B} (= {A,B}) {I, J, A diff B}

... con nomi degli indici espliciti (usati in espressioni)

{a in A} {a in I, b in J} {i in I, j in I}

... con possibili restrizioni («tale che...»)

{i in I, j in I, k in K :

i != j and c[i,j] <= soglia[k]}

Implementiamo il modello del trasporto dei frigoriferi, basato sul modello dei trasporti.

Una ditta di produzione di elettrodomestici produce dei frigoriferi in tre stabilimenti e li smista in quattro magazzini intermedi di vendita. La produzione settimanale nei tre stabilimenti A, B e C è rispettivamente di 50, 70 e 20 unità. La quantità richiesta dai 4 magazzini è rispettivamente di 10, 60, 30 e 40 unità. I costi per il trasporto di un frigorifero tra gli stabilimenti e i magazzini 1, 2, 3 e 4 sono i seguenti:

- dallo stabilimento A: 6, 8, 3, 4 euro;
- dallo stabilimento B: 2, 3, 1, 3 euro;
- dallo stabilimento C: 2, 4, 6, 5 euro.

Utilizzare AMPL per determinare il piano di trasporti di costo minimo, considerando che non sono ammesse rimanenze alla fine della settimana e che lo stesso modello dovrà essere utilizzato per diverse settimane.

Modello generale: trasporti

I insieme dei centri di offerta; O_i ammontare dell'offerta in $i \in I$;

J insieme dei centri di domanda; D_j ammontare della domanda in $j \in J$.

C_{ij} costo (unitario) per il trasporto da $i \in I$ a $j \in J$;

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in J} x_{ij} \leq O_i \quad \forall i \in I \\ & \sum_{i \in I} x_{ij} \geq D_j \quad \forall j \in J \\ & x_{ij} \in \mathbb{R}_+ [\mathbb{Z}_+ | \{0, 1\}] \quad \forall i \in I, j \in J \end{aligned}$$

- Sia x_{ij} il numero di frigoriferi prodotti nello stabilimento i e smistati nel magazzino j

$$\begin{aligned} \min \quad & 6x_{A1} + 8x_{A2} + 3x_{A3} + 4x_{A4} + \\ & + 2x_{B1} + 3x_{B2} + 1x_{B3} + 3x_{B4} + \\ & + 2x_{C1} + 4x_{C2} + 6x_{C3} + 5x_{C4} \end{aligned}$$

s.t.

$$\begin{aligned} x_{A1} + x_{A2} + x_{A3} + x_{A4} &\leq 50 && (\text{capacità produttiva stabilimento A}) \\ x_{B1} + x_{B2} + x_{B3} + x_{B4} &\leq 70 && (\text{capacità produttiva stabilimento B}) \\ x_{C1} + x_{C2} + x_{C3} + x_{C4} &\leq 20 && (\text{capacità produttiva stabilimento C}) \end{aligned}$$

$$\begin{aligned} x_{A1} + x_{B1} + x_{C1} &\geq 10 && (\text{domanda magazzino 1}) \\ x_{A2} + x_{B2} + x_{C2} &\geq 60 && (\text{domanda magazzino 2}) \\ x_{A3} + x_{B3} + x_{C3} &\geq 30 && (\text{domanda magazzino 3}) \\ x_{A4} + x_{B4} + x_{C4} &\geq 40 && (\text{domanda magazzino 4}) \end{aligned}$$

$$x_{ij} \in \mathbb{Z}_+ \quad \forall i \in \{A, B, C\}, j \in \{1, 2, 3, 4\}$$

Il modello generale dei trasporti viene implementato come segue su *trasporto.mod*:

```
set I; #insieme offerta
set J; #insieme domanda

param O{I}; #offerta
param D{J}; #domanda
param C{I, J}; #costo trasporto

var x{I, J} >=0 integer;

minimize costo_totale:
    sum{i in I, j in J} C[i,j] * x[i, j];

s.t. origine{i in I}: sum{j in J} x[i, j] <= O[i];

s.t. destinazione{j in J}: sum{i in I} x[i, j] >= D[j];
```

I dati del modello dei frigoriferi si trovano nel file come segue *traspFrigo.dat*:

```
set I := A B C;
set J := 1 2 3 4;

param O := A 50 B 70 C 20;

param D := 1 10 2 60 3 30 4 40;

param C :
A      1      2      3      4      :=
A      6      8      3      4
B      2      3      1      3
C      2      4      6      5
;
```

Poi, il successivo file per farlo eseguire, *traspFrigo.run*:

```
reset;
model trasporto.mod;
data traspFrigo.dat;
option solver cplex;
solve;
display costo_totale, x;
```

Possiamo dare vari comandi, ad esempio per visualizzare le x significative:

```
ampl: display {i in I, j in J : x[i, j] > 0} x[i, j];
x[i, j] :=
A 3 10
A 4 40
B 2 50
B 3 20
C 1 10
C 2 10
;
```

Oppure, sfruttando le proprietà indicizzanti di display visualizzando i comandi come separati (perché non ci sono le parentesi):

```
ampl: display {i in I} sum {j in J} x[i, j] , O;
: sum{j in J} x[i, j] O :=
A      50      50
B      70      70
C      20      20
;
```

Possiamo vedere quanto vale l'espressione *origine* come segue:

```
ampl: display origine.body;
origine.body [*] :=
A 50
B 70
C 20
;
```

Inoltre, possiamo visualizzare la differenza tra le singole variabili:

```
ampl: display {i in I} origine[i].body - O[i];
(origine[i].body) - O[i] [*] :=
A 0
B 0
C 0
;
```

Similmente, possiamo avere i classici costrutti iterativi *for*, i costrutti condizionale *if-else*, ecc.

Ora ci concentriamo su un esercizio da fare da parte nostra → Distribuzione PC

Un'azienda assembla dei PC in tre diversi stabilimenti con diverso costo unitario di produzione. I PC sono venduti a cinque clienti bancari e si sopportano dei costi di trasporto (inclusi gli oneri di importazione) per spedire un PC da ciascuno stabilimento a ciascun cliente. Sono definite le richieste di PC di ogni cliente e la produzione di ciascuno stabilimento è limitata. Non sono ammessi eccessi di produzione. I dati sono riassunti nella tabella seguente.

Scrivere in AMPL un modello del problema e fornire la soluzione, in termini di costo complessivo di trasporto e di quantità trasportate tra stabilimenti e sedi bancarie.

Produzione			Costi di trasporto				
Unità	costo unit.	Capacità	Banca Intesa	Uni Credit	Anton Veneta	Credit Suisse	Banca Cina
Italia	220	10000	5,5	7,5	6,9	8,0	10,3
Cina	180	20000	15,0	14,3	13,0	16,4	5,0
Francia	200	10000	6,0	7,8	6,3	6,8	11,0
Domanda			7100	3400	9700	5200	3050

Insiemi: S (stabilimenti) e B (banche)

Parametri: w_i (costi prod.), c_{ij} (costi trasp.), a_i (capacità prod.), b_j (richieste)

$$\begin{aligned}
 \min \quad & \sum_{i \in S, j \in B} (w_i + c_{ij}) x_{ij} \\
 \text{s.t.} \quad & \sum_{j \in B} x_{ij} \leq a_i, \quad \forall i \in S \\
 & \sum_{i \in S} x_{ij} = b_j, \quad \forall j \in B \quad (\text{no eccessi di produzione} \Rightarrow \text{uguaglianza}) \\
 & x_{ij} \in \mathbb{Z}_+ \quad \forall i \in S, j \in B
 \end{aligned}$$

Il modello viene considerato nel file *pc.mod*:

```
set I; #stabilimenti - S
set J; #banche - B

param A{I}; #capacità produttive
param B{J}; #richieste
param W{I}; #costi produttivi
param C{I, J}; #costi
var x{I, J} integer >= 0;

minimize costo: sum{i in I, j in J}
(W[i] + C[i,j])*x[i,j];
```

Scritto da Gabriel


```
s.t. produzione{i in I}: sum{j in J} x[i,j] <= A[i];
s.t. domanda{j in J}: sum{i in I} x[i,j] = B[j];
```

I dati invece seguono nel file *pc.dat*:

```
set I := it ch fr;
set J := bi uc av cs bc;

param A := it 10000 ch 20000 fr 10000;
param B := bi 7100 uc 3400 av 9700 cs 5200 bc 3050;

param W := it 220 ch 180 fr 200;

param C:
it      bi      uc      av      cs      bc :=
ch      5.5      7.5      6.9      8.0      10.3
fr      15      14.3     13.0     16.4      5.0
fr      6.0      7.8      6.3      6.8      11.0
;
```

Il file *pc.run* segue qui:

```
reset;
model pc.mod;
data pc.dat;
option solver cplex;
solve;
display costo, x;
```

Esercizi vari Metodo del Simplexso

Domanda spesso posta nel gruppo

- Perché il prof non si ferma subito con il simplexso?
 - Io credo sia per un fatto di voler "rendere evidente" la cosa, quindi il fatto che individuando che è illimitato in un passaggio, continuerà a rimanere illimitato nel corso delle iterazioni. Dovrebbe essere ugualmente corretto fermarsi subito, ma ho visto che anche in altre sue correzioni lui ha un problema illimitato, ma avanza comunque e mostra con più passaggi che rimane come tale.

$$\begin{aligned} \min \quad & -x_1 - 3x_2 + x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 \leq 3 \\ & x_1 + x_2 + 3x_3 \leq 6 \\ & 2x_1 + x_2 + 3x_3 \leq 8 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Riscriviamo in forma standard:

$$\begin{aligned} \min \quad & -x_1 - 3x_2 + x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + x_4 = 3 \\ & x_1 + x_2 + 3x_3 + x_5 = 6 \\ & 2x_1 + x_2 + 3x_3 + x_6 = 8 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

Iterazione (1) Base iniziale $\rightarrow B = [x_4, x_5, x_6]$ ammissibile

	x_1	x_2	x_3	x_4	x_5	x_6	z	$\bar{b}$
$-z$	-1	-3	1	0	0	0	-1	0
x_4	2	1	0	1	0	0	0	3
x_5	1	1	3	0	1	0	0	6
x_6	2	1	3	0	0	1	0	8

- 1) F.C? $\rightarrow$ Sì
- 2) Ammissibile $\rightarrow$ Sì
- 3) Ottima $\rightarrow$ Non so
- 4) Illimitata $\rightarrow$ Non so
- 5) Entra $\rightarrow$ Tra x_1, x_2 scelgo x_1 (Bland)
- 6) Esce $\rightarrow \arg\min\{3/2, 6/1, 8/2\} = \arg\{3/2\} = x_4$

Riportiamo il tableau in forma canonica eseguendo le operazioni di pivoting:

Iterazione (2) $\rightarrow B = [x_1, x_5, x_6]$

Operazioni: $R_1 \leftarrow R_1/2, R_2 \leftarrow R_2 - R_1/2, R_3 \leftarrow R_3 - R_1, R_4 \leftarrow R_4 + R_1/2$.

	x_1	x_2	x_3	x_4	x_5	x_6	z	$\bar{b}$
$-z$	0	-5/2	1	1/2	0	0	-1	3/2
x_1	1	1/2	0	1/2	0	0	0	3/2
x_5	0	1/2	3	-1/2	1	0	0	9/2
x_6	0	0	3	-1	0	1	0	5

- 1) F.C? $\rightarrow$ Sì
- 2) Ammissibile? $\rightarrow$ Sì
- 3) Ottima? $\rightarrow$ Non so
- 4) Illimitata $\rightarrow$ Non so
- 5) Entra? x_2 l'unica variabile a costo ridotto negativo
- 6) Esce? $\arg\min\{3/2/1/2, 9/2/1/2, 5/0\} = \arg\{3\} = x_1$

Iterazione (3) $\rightarrow B = [x_2, x_5, x_6]$

	x_1	x_2	x_3	x_4	x_5	x_6	z	$\bar{b}$
$-z$	5	0	1	3	0	0	-1	9
x_2	2	1	0	1	0	0	0	3
x_5	-1	0	3	-1	1	0	0	3
x_6	0	0	3	-1	0	1	0	5

- 1) F.C? $\rightarrow$ Sì
- 2) Ammissibile? $\rightarrow$ Sì
- 3) Ottima $\rightarrow$ Sì (tutti i costi ridotti sono positivi)

$$z_{\min} = -\frac{3}{2} - \frac{5}{2}\theta = -\frac{3}{2} - \frac{5}{2}3 = -9$$

Alla fine:

$$\begin{aligned} z^* &= -9 \quad x_1^* = 0 \quad x_2^* = 3 \quad x_3^* = 0 \\ x_5^* &= 0 \text{ (1° vincolo saturo)} \\ x_5^* &= 3 \text{ (2° vincolo lasco)} \\ x_3^* &= 5 \text{ (3° vincolo lasco)} \end{aligned}$$

Riferimento: Esercizio 8 delle dispense

$$\begin{array}{ll}
 \max & 2x_1 + 3x_2 - 12x_3 \\
 \text{s.t.} & x_1 - x_2 + 4x_3 \geq -2 \\
 & 2x_1 + 2x_2 - 6x_3 \leq 4 \\
 & x_1 - x_2 \leq 2 \\
 & x_1, x_2 \geq 0 \\
 & x_3 \leq 0
 \end{array}$$

Simpleso e Bland (variabili ordinate come le colonne).

Riscriviamo in forma standard e passiamo prima all'uguaglianza introducendo le variabili di slack.

$$\begin{array}{ll}
 \min & -2x_2 - 3x_3 + 12x_4 \\
 \text{s.t.} & x_1 - x_2 + 4x_3 - x_4 = -2 \\
 & 2x_1 + 2x_2 - 6x_3 + x_5 = 4 \\
 & x_1 - x_2 + x_6 = 2 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \quad x_3 \leq 0
 \end{array}$$

Variabile negativa da considerare:

$$\begin{array}{ll}
 \min & -2x_2 - 3x_3 + 12x_4 \\
 & x_3 = -\hat{x}_3 \quad \hat{x}_3 = -x_3 \geq 0 \\
 \text{s.t.} & x_1 - x_2 - 4\hat{x}_3 - x_4 = -2 \\
 & 2x_1 + 2x_2 + 6x_3 + x_5 = 4 \\
 & x_1 - x_2 + x_6 = 2
 \end{array}$$

Dobbiamo avere i termini noti non negativi e si inverte segno alla prima uguaglianza:

$$\begin{array}{ll}
 \min & -2x_2 - 3x_3 + 12x_4 \\
 & x_3 = -\hat{x}_3 \quad \hat{x}_3 = -x_3 \geq 0 \\
 \text{s.t.} & -x_1 + x_2 + 4\hat{x}_3 + x_4 = -2 \\
 & 2x_1 + 2x_2 + 6x_3 + x_5 = 4 \\
 & x_1 - x_2 + x_6 = 2 \\
 & x_1, x_2, \hat{x}_3, x_4, x_5, x_6 \geq 0
 \end{array}$$

Ora siamo in forma standard e grazie alle variabili di slack abbiamo una base ammissibile.

Iterazione 1 $\rightarrow B = [x_4, x_5, x_6]$

	x_1	x_2	$\hat{x}_3$	x_4	x_5	x_6	z	$\bar{b}$
$-z$	-2	-3	-12	0	0	0	-1	0
x_4	-1	1	4	1	0	0	0	2
x_5	2	2	6	0	1	0	0	4
x_6	1	-1	0	0	0	1	0	2

1) F.C.? $\rightarrow$ Sì

2) Ammissibile $\rightarrow$ Sì (se si parte da base ammissibile, si deve sempre rimanere in base ammissibile)

3) Ottima? $\rightarrow$ Non so

4) Illimitata? $\rightarrow$ Non so

5) Entra $\rightarrow$ Tra $x_1, x_2, \hat{x}_3$ scelgo x_1 per Bland

6) Esce? $\arg \min \left\{ -\frac{2}{1}, \frac{4}{2}, \frac{2}{1} \right\} = \arg\{2\} =$ Abbiamo due variabili con lo stesso rapporto e, per Bland, tra x_5, x_6 scelgo x_5

Iterazione 2 $\rightarrow B = [x_1, x_4, x_6] \rightarrow$ Si noti che con due variabili che corrispondevano a rapporto minimo, una entra ed una esce dalla base. In questo modo, ci aspettiamo la soluzione degenerare e, come si vede, le colonne della matrice identità sono sparse diversamente.

Operazioni: $R_2 \leftarrow R_2/2, R_1 \leftarrow R_1 + R_2/2, R_3 \leftarrow R_3 - R_2/2, R_0 \leftarrow R_0 + R_2.$

	x_1	x_2	$\hat{x}_3$	x_4	x_5	x_6	z	$\bar{b}$
$-z$	0	-1	-6	0	1	0	-1	4
x_4	0	2	7	1	1/2	0	0	4
x_1	1	1	3	0	1/2	0	0	2
x_6	0	-2	-3	0	-1/2	1	0	0

1) F.C.? $\rightarrow$ Sì

2) Ammissibile? $\rightarrow$ Sì

3) Ottima? $\rightarrow$ Non so

4) Illimitata? $\rightarrow$ Non so

5) Entra? Tra x_2 e $\hat{x}_3$ scelgo x_2 (Bland)

6) Esce? $\arg \min \left\{ \frac{4}{2}, \frac{2}{1}, \frac{0}{-1} \right\} = \arg\{2\}$: tra x_4 e x_1 scelgo x_1

Iterazione 3 $\rightarrow B = [x_4, x_2, x_6] \rightarrow$ Attenzione che 0 nei successivi rapporti si considera dato che, come detto, si ha a che fare con una soluzione degenerare.

Operazioni: $R_1 \leftarrow R_1 - 2R_2$, $R_3 \leftarrow R_3 + 2R_2$, $R_0 \leftarrow R_0 + R_2$.

	x_1	x_2	$\hat{x}_3$	x_4	x_5	x_6	z	$\bar{b}$
$-z$	1	0	-3	0	3/2	0	-1	6
x_4	-2	0	1	1	-1/2	0	0	0
x_2	1	1	3	0	1/2	0	0	2
x_6	2	0	3	0	1/2	1	0	4

- 1) F.C. ? Sì
- 2) Ammissibile? → Sì
- 3) Ottima? → Sì
- 4) Illimitata? → Non so
- 5) Entra? $\hat{x}_3$
- 6) Esce? $\arg \min \left\{ \frac{0}{1}, \frac{2}{3}, \frac{4}{3} \right\} = \arg\{0\} = x_4$

Iterazione 4 → $B = [\hat{x}_3 \ x_2 \ x_6]$ → Il rapporto con 0 non lo considero in quanto negativo e, come tale, esce dalla degenerazione.

	x_1	x_2	$\hat{x}_3$	x_4	x_5	x_6	z	$\bar{b}$
$-z$	-5	0	0	3	0	0	-1	6
$\hat{x}_3$	-2	0	1	1	-1/2	0	0	0
x_2	7	1	0	-3	2	0	0	2
x_6	8	0	0	-3	2	1	0	4

- 1) F.C. ? Sì
- 2) Ammissibile? → Sì
- 3) Ottima? → Sì
- 4) Illimitata? → Non so
- 5) Entra? x_1
- 6) Esce? $\arg \min \left\{ \frac{0}{-2}, \frac{2}{7}, \frac{4}{8} \right\} = \arg\left\{\frac{2}{7}\right\} = x_2$

Iterazione 5 → $B = [\hat{x}_3 \ x_1 \ x_6]$

Operazioni: $R_2 \leftarrow R_2/7$, $R_1 \leftarrow R_1 + 2/7R_2$, $R_3 \leftarrow R_3 - 8/7R_2$, $R_0 \leftarrow R_0 + 5/7R_2$.

	x_1	x_2	$\hat{x}_3$	x_4	x_5	x_6	z	$\bar{b}$
$-z$	0	5/7	0	6/7	10/7	0	-1	52/7
$\hat{x}_3$	0	2/7	1	1/7	1/14	0	0	4/7
x_1	1	1/7	0	-3/7	2/7	0	0	2/7
x_6	0	-8/7	0	3/7	-2/7	1	0	12/7

- 1) F.C.? → Sì
- 2) Ammissibile → Sì
- 3) Ottima → Sì

$$z^* = -\frac{52}{7} \quad z_{\max} = \frac{52}{7} \quad x_1 = \frac{2}{7} \quad x_2 = 0 \quad x_3^* = \frac{4}{7} \quad x_4^* = 0 \text{ (saturo)} = x_5^* = 0 \text{ (saturo)}$$

Riferimento: Esercizio 12 delle dispense di esercizi risolti sul simplesso

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 - x_3 + 3x_4 \\ \text{s.t.} \quad & x_1 - x_2 + 2x_3 - 2x_4 \geq -2 \\ & 2x_1 + 2x_2 + x_3 + 4x_4 \leq 4 \\ & x_1 - x_2 + 2x_3 \leq 2 \\ & x_1, x_2, x_3 \geq 0 \\ & x_4 \leq 0 \end{aligned}$$

Soluzione. Riscriviamo il problema in forma standard. Poiché $x_4 \leq 0$ introduciamo una nuova variabile $\hat{x}_4 = -x_4$ con $\hat{x}_4 \geq 0$.

$$\begin{aligned} \min \quad & -2x_1 - 3x_2 + x_3 + 3\hat{x}_4 \\ \text{s.t.} \quad & -x_1 + x_2 - 2x_3 - 2\hat{x}_4 + x_5 = 2 \\ & 2x_1 + 2x_2 + x_3 - 4\hat{x}_4 + x_6 = 4 \\ & x_1 - x_2 + 2x_3 + x_7 = 2 \\ & x_1, x_2, x_3, \hat{x}_4, x_5, x_6, x_7 \geq 0 \end{aligned}$$

Iterazione 1 → $B = [x_5 \ x_6 \ x_7]$

	x_1	x_2	x_3	$\hat{x}_4$	x_5	x_6	x_7	z	$\bar{b}$
$-z$	-2	-3	1	3	0	0	0	-1	0
x_5	-1	1	-2	-2	1	0	0	0	2
x_6	2	2	1	-4	0	1	0	0	4
x_7	1	-1	2	0	0	0	1	0	2

- 1) F.C.? Sì
- 2) Ammissibile? Sì
- 3) Ottima? Non so
- 4) Illimitata? Non so
- 5) Entra? Tra x_1 e x_2 entra x_1 (Bland)
- 6) Esce? $\arg \min \left\{ -\frac{2}{-1}, \frac{4}{2}, \frac{2}{1} \right\} = \arg\{2\}$: tra x_6 e x_7 scelgo x_6

Iterazione 2 $\rightarrow B = [x_5 \ x_1 \ x_7]$

Operazioni: $R_2 \leftarrow R_2/2$, $R_1 \leftarrow R_1 + R_2/2$, $R_3 \leftarrow R_3 - R_2/2$, $R_0 \leftarrow R_0 + R_2$.

	x_1	x_2	x_3	$\hat{x}_4$	x_5	x_6	x_7	z	$\bar{b}$
$-z$	0	-1	2	-1	0	1	0	-1	4
x_5	0	2	-3/2	-4	1	1/2	0	0	4
x_1	1	1	1/2	-2	0	1/2	0	0	2
x_7	0	-2	3/2	2	0	-1/2	1	0	0

- 1) F.C.? Sì
- 2) Ammissibile? Sì
- 3) Ottima? Non so
- 4) Illimitata? Non so
- 5) Entra? Tra x_2 e $\hat{x}_4$ (Bland) $\rightarrow x_2$
- 6) Esce? $\arg \min \left\{ \frac{4}{2}, \frac{2}{1}, \frac{0}{-2} \right\} = \arg\{2\}$: tra x_5 e x_1 scelgo x_1 (Bland)

Iterazione 3 $\rightarrow B = [x_5 \ x_2 \ x_7]$

Operazioni: $R_1 \leftarrow R_1 - 2R_2$, $R_3 \leftarrow R_3 + 2R_2$, $R_0 \leftarrow R_0 + R_2$. (Si ricordi che non è conveniente eseguire l'operazione $R_1 \leftarrow R_1 + R_3$ in quanto modificherebbe anche la colonna A_7 .)

	x_1	x_2	x_3	$\hat{x}_4$	x_5	x_6	x_7	z	$\bar{b}$
$-z$	1	0	5/2	-3	0	3/2	0	-1	6
x_5	-2	0	-5/2	0	1	-1/2	0	0	0
x_2	1	1	1/2	-2	0	1/2	0	0	2
x_7	2	0	5/2	-2	0	1/2	1	0	4

- 1) F.C? $\rightarrow$ Sì
- 2) Ammissibile? Sì
- 3) Ottima? Non so
- 4) Illimitato? $\rightarrow$ La colonna di $\hat{x}_4$ è tutta negativa $\rightarrow$ Sì
 $\bar{c}_h < 0$ e $\bar{c}_{ih} \leq 0$

08/11/2022: Dualità e Programmazione Lineare: Teoremi, regole di trasformazione, variabili e condizioni

Come sappiamo, un problema PL consiste nel determinare il valore ottimo z^* di una funzione obiettivo su un poliedro P . Normalmente, questo è stato formalizzato sulla base di questi passi:

- Si ipotizza vettore di punti x
- Verifica $x \in P$ e calcola $c^T x$
- z^* è il minimo tra tutti quelli calcolati

$$(PL_1) \quad \begin{aligned} z^* &= \min c^T x \\ \text{s.t.} \quad Ax &= b \\ x &\geq 0 \end{aligned}$$

Il punto di partenza, qui, è il vettore x . Adesso ripensiamo il problema, partendo però da un valore w della funzione obiettivo. In pratica:

- Si ipotizza un valore della funzione obiettivo w
- Verifica $w \leq c^T x$ su P (quindi, che w sia lower bound per z^*), facendo variare x su tutto il poliedro P e facendo valere la condizione di prima
- w^* è il massimo tra tutti quelli ipotizzati e verificati

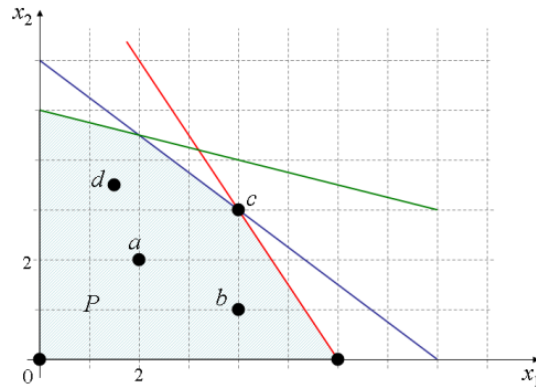
$$(PL_2) \quad \begin{aligned} w^* &= \max w \\ \text{s.t.} \quad w &\leq c^T x \quad \forall x \in P \\ w &\in \mathbb{R} \end{aligned}$$

Si noti che PL_2 è un problema di programmazione lineare con una sola variabile w e un numero infinito di vincoli per tutti i punti di P . Vogliamo quindi trovare il massimo valore della variabile reale w che permetta di soddisfare tutti i vincoli $w \leq c^T x$, al variare di x in P .

Esempio 1 Consideriamo il seguente problema di programmazione lineare:

$$\begin{array}{ll} \min & -x_1 - x_2 \\ \text{s.t.} & 3x_1 + 4x_2 \leq 24 \\ & x_1 + 4x_2 \leq 20 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{array}$$

la cui regione ammissibile corrisponde al poliedro P rappresentato in figura 1:



Per ottenere il problema di programmazione lineare corrispondente, si devono considerare tutti i punti di P , uno alla volta, e scrivere i relativi vincoli, ottenendo:

$$\begin{array}{ll} \max & w \\ \text{s.t.} & w \leq -4 \quad (= c^T x \mid x=a=(2,2)) \\ & w \leq 0 \quad (= c^T x \mid x=0=(0,0)) \\ & w \leq -5 \quad (= c^T x \mid x=b=(4,1)) \\ & w \leq -7 \quad (= c^T x \mid x=c=(3,3)) \\ & w \leq -5 \quad (= c^T x \mid x=d=(2,4)) \\ & \dots \\ & w \text{ libera} \end{array}$$

Se z^* esiste finito, $z^* = w^*$ (sulla base di n variabili, m vincoli, una variabile w). Infatti, se così non fosse, si violerebbe la disuguaglianza $w^* \leq c^T x^* = z^*$

Ciò significa che in corrispondenza del valore della soluzione ottima, avremo il valore massimo della funzione obiettivo.

Riassumendo: dato un problema di programmazione lineare $\min\{c^T x : x \in P\}$, è possibile associargli un problema di programmazione lineare $\max\{w : w \leq c^T x, \forall x \in P\}$. Sotto l'ipotesi che il primo problema ammetta ottimo finito, i valori ottimi delle funzioni obiettivo dei due problemi coincidono.

Vogliamo riscrivere il problema PL_2 in forma maneggevole (meno vincoli, dato che per il momento ne ha un numero infinito).

Assumendo che PL_1 ammetta soluzione di base ottima, avrà ottimo limitato e scriviamo i vincoli del problema per i soli punti corrispondenti ai vertici del poliedro, omettendo i vincoli con termine noto sicuramente $\leq$ al minimo del problema (in quanto il numero di vincoli è limitato dal numero di soluzioni di base).

Vogliamo trovare condizioni tali che ammetta l'introduzione di un limitato numero di variabili, dato che limitiamo il numero di vincoli (pur considerando che questo può essere molto elevato):

Scritto da Gabriel

$$(PL_2) \quad \begin{array}{ll} w^* = \max & w \\ \text{s.t.} & w \leq c^T x \quad \forall x \in \text{vertici}(P) \\ & w \in \mathbb{R} \end{array} \quad \boxed{w \leq c^T x, \forall x \in P} \iff \boxed{\text{condizioni}}$$

A livello esteso, i costi ridotti sono come segue (sapendo che sono in forma canonica funziona):

$$\bullet \quad Ax = b \Rightarrow Bx_B + Fx_F = b \Rightarrow x_B = B^{-1}b - B^{-1}Fx_F$$

$$\bullet \quad z = c^T x = c_B^T x_B + c_F^T x_F = c_B^T (B^{-1}b - B^{-1}Fx_F) + c_F^T x_F \\ \Rightarrow z = \underbrace{c_B^T B^{-1}b}_{\bar{z}_B} + \underbrace{(c_F^T - c_B^T B^{-1}F)}_{\bar{c}_F^T} x_F$$

$$\bullet \quad \bar{c}^T = [\bar{c}_B^T \mid \bar{c}_F^T] = [0^T \mid c_F^T - c_B^T B^{-1}F] = \\ = [c_B^T - c_B^T B^{-1}B \mid c_F^T - c_B^T B^{-1}F] = \\ = [c_B^T \mid c_F^T] - [c_B^T B^{-1}B \mid c_B^T B^{-1}F] = \\ = c^T - c_B^T B^{-1}[B \mid F] = \\ = c^T - c_B^T B^{-1}A$$

Possiamo prendere il problema di PL e mettere il sistema in forma canonica $\Rightarrow x_B = B^{-1}b - B^{-1}Fx_F$

Il valore della f.o. vale $c_B^T (B^{-1}b - B^{-1}Fx_F) + c_F^T x_F$ (non facendo comparire le variabili in base, con costo ridotto < 0). Esisterà almeno una soluzione ottima di base a cui si converge.

I costi ridotti sono dati da quelli delle v. in base e quello delle v. fuori base $\Rightarrow \bar{c}^T = c^T - c_B^T B^{-1}A$

Infine, il valore della f.o. è $z^* = c_B^T B^{-1}b$

Sapendo questa equivalenza, avremo un insieme di condizioni necessarie:

- Il poliedro esiste ed esiste una base ottima con ottimo limitato
 $P \neq \emptyset \wedge z^* > -\infty \Rightarrow \exists c^T \geq c_B^T B^{-1}A$
- Esiste una soluzione ottima di base
 $w \leq c^T x, \forall x \in P, P \neq \emptyset, z > -\infty$
- Esiste una base ottima $\Rightarrow \exists u \in \mathbb{R}^m : c_B^T B^{-1} = u^T$

Riassumendo, avremo che dalle condizioni a sinistra, ricavo quanto sta a destra.

$$\boxed{\begin{cases} w \leq c^T x, \forall x \in P \\ P \neq \emptyset \\ z^* > -\infty \end{cases}} \implies \boxed{\exists u \in \mathbb{R}^m : \begin{cases} u^T A \leq c^T \\ w \leq u^T b \end{cases}}$$

Le condizioni sono anche sufficienti: $\forall x \in P = \{x : Ax = b, x \geq 0\}$

- $\exists u \in \mathbb{R}^m : u^T A \leq c^T \Rightarrow u^T Ax \leq c^T x \Rightarrow u^T b \leq c^T x$ (cioè, la base è lower bound della soluzione ottima)
- $\exists u \in \mathbb{R}^m : w \leq u^T b \Rightarrow w \leq u^T b \leq c^T x$ (cioè, il valore della f.o. e la base sono lower bound della sol. ottima)

Dato un problema di programmazione lineare $z^* = \min\{c^T x : x \in P\}$ con $P = \{x : Ax = b, x \geq 0\}$, $P \neq \emptyset$ e z^* limitato, valgono le seguenti condizioni necessarie e sufficienti:

$$w \leq c^T x, \forall x \in P \iff \exists u \in \mathbb{R}^m : \begin{cases} u^T A \leq c^T \\ w \leq u^T b \end{cases}$$

Sapendo che:

$$\begin{aligned} w &= \max w \\ \text{s.t. } w &\leq c^T x \quad \forall x \in R \\ w &\in R \end{aligned}$$

Il problema (PL_2) può pertanto essere scritto in modo equivalente come

$$(PL_2) \quad \begin{aligned} w^* &= \max w \\ \text{s.t. } w &\leq u^T b && (\text{un vincolo}) \\ u^T A &\leq c^T && (n \text{ vincoli}) \\ u &\in \mathbb{R}^m && (m \text{ variabili}) \end{aligned}$$

Allora (immagine di dx)

Quindi, essendo all'ottimo $w = u^T b$, possiamo eliminare il primo vincolo e la stessa variabile w , ottenendo la formulazione equivalente:

$$(PL_2) \quad \begin{aligned} w^* &= \max u^T b \\ \text{s.t. } u^T A &\leq c^T && (n \text{ vincoli}) \\ u &\in \mathbb{R}^m && (m \text{ variabili}) \end{aligned}$$

Riassumendo: Ad un problema di programmazione lineare (PL_1) è possibile associare un altro problema di programmazione lineare (PL_2) nella forma

$$\begin{array}{ll} (PL_1) & (PL_2) \\ z^* = \min c^T x & w^* = \max u^T b \\ \text{s.t. } Ax = b & \text{s.t. } u^T A \leq c^T \\ x \geq 0 & u \in \mathbb{R}^m \end{array}$$

Il problema PL_1 è composto da:

$3n$ variabili x

m vincoli

Il problema PL_2 è composto da:

n vincoli

m variabili u

Il problema PL_1 è il *problema primale*, mentre PL_2 è il *problema duale*, dunque essi sono la *coppia primale/duale*. I problemi sono diversi, ma sono imparentati dal fatto di avere lo stesso valore della f.o. Ad ogni vincolo primale è associata una variabile duale e i vincoli duali possono essere scritti sapendo che ad ogni variabile primale è associato un vincolo duale. Quanto detto si formalizza come segue:

- vincolo primale $\leftrightarrow$ variabile duale

$$u^T b = u_1 b_1 + u_2 b_2 + \dots + u_m b_m$$

- variabile primale $\leftrightarrow$ vincolo duale

$$u^T A \leq c^T \equiv \begin{cases} u^T A_1 \leq c_1 \\ u^T A_2 \leq c_2 \\ \dots \\ u^T A_j \leq c_j \\ \dots \\ u^T A_n \leq c_n \end{cases} \equiv u^T A_j \leq c_j, \quad \forall j = 1 \dots n$$

Diamo un'idea di calcolo, passo per passo, considerando il seguente esempio:

Esempio

$$\begin{array}{llll} \min & 2x_1 & -3x_2 & -x_3 \\ \text{s.t.} & 2x_1 & +4x_2 & +3x_3 = 22 \\ & x_1 & -7x_2 & = 15 \\ & x_1 & , & x_2 , & x_3 \geq 0 \end{array}$$

$$\max u^T b = [u_1 \ u_2] \begin{bmatrix} 22 \\ 15 \end{bmatrix}$$

$$u^T A \leq c^T$$

$$\max 22u_1 + 15u_2$$

$$u^T A = [u_1 \ u_2] \begin{bmatrix} 2 & 4 & 3 \\ 1 & -7 & 0 \end{bmatrix} =$$

$$[2u_1 + 4u_2 \mid 3u_1 - 7u_2 \mid 0] \leq [2 \mid 1 \mid -3]$$

$$\leq [2 \mid 1 \mid -3] \quad 1 \mid -1$$

Note:

- le variabili u_i sono i termini noti 22 e 15 e, essendo duale, diventa max
- la scrittura dei vincoli procede in colonna: per esempio, nel primo caso, avremo che u_1 e u_2 altro non sono che la prima colonna sotto $2x_1$; da cui il ≤ 2 , che infatti è il coefficiente del numero detto
- similmente, anche gli altri due vincoli ragionano in colonna nel modo appena descritto

Possiamo dimostrare anche la *sufficienza* delle condizioni $u^T A \leq c^T$ e $w \leq u^T b$ considerando una generalizzazione per rappresentare problemi primali che sono in qualsiasi forma.

- Per la **sufficienza**, definire $u^T A \sim c^T$ e u per **mantenere**

$$c^T x \geq u^T Ax \geq u^T b \geq w, \forall x \in F$$

siccome $x \geq 0$ serve $u^T A \leq c^T$ siccome $Ax = b$ vale con u qualsiasi

Cerchiamo ora di capire sotto quali condizioni, più in generale, la catena di minorazioni può essere estesa a variabili primali non necessariamente ≥ 0 e a vincoli primali non necessariamente di uguaglianza.

Scriviamo le componenti della prima minorazione $c^T x \geq u^T Ax$ (cercando di mantenere i $\geq$); stiamo stabilendo il verso delle variabili duali.

- Consideriamo $c^T x \geq u^T Ax$:

$$u^T Ax = [u^T A_1 \mid u^T A_2 \mid \dots \mid u^T A_j \mid \dots \mid u^T A_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix} =$$

$$= u^T A_1 x_1 + u^T A_2 x_2 + \dots + u^T A_j x_j + \dots + u^T A_n x_n$$

$$c^T x = c_1 x_1 + c_2 x_2 + \dots + c_j x_j + \dots + c_n x_n$$

$u^T Ax$ è uno scalare composto da una somma di fattori $u^T A_j x_j$ e, per riuscire a mantenere il rapporto di $\geq$, allora $c^T x = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$. Se ciascun fattore è $\geq$ all'omologo, così sarà anche la loro somma.

Affinché sia valido $c^T x \geq u^T A x$, è sufficiente che la disuguaglianza sia soddisfatta da ciascuno degli addendi, cioè

$$c_j x_j \geq u^T A_j x_j, \forall j = 1 \dots n$$

Quindi si considera una variabile primale alla volta e:

- se $x_j \geq 0$, la condizione sufficiente è che $u^T A_j \leq c_j$ (variabile primale $\geq 0 \Rightarrow$ vincolo duale di $\leq$);
- se $x_j \leq 0$, la condizione sufficiente è che $u^T A_j \geq c_j$ (variabile primale $\leq 0 \Rightarrow$ vincolo duale di $\geq$);
- se x_j è una variabile libera, la condizione $c_j x_j \geq u^T A_j x_j$ potrebbe cambiare verso a seconda del segno di x_j , pertanto abbiamo più restrizioni e la condizione sufficiente è che $u^T A_j = c_j$ (variabile primale libera $\Rightarrow$ vincolo duale di uguaglianza).

variabile primale ≥ 0 / ≤ 0 / libera $\rightsquigarrow$ vincolo duale $\leq$ / $\geq$ / $=$

Ora dobbiamo mantenere la seconda minorazione, garantendo che $u^T A x \geq u^T b$. Stiamo ora stabilendo il *segno delle variabili duali*

- Manteniamo adesso la seconda disuguaglianza della catena

$$\underbrace{c^T x}_{x \geq 0} \geq \underbrace{u^T A x}_{Ax=b} \geq u^T b \geq w, \forall x \in P$$

- Consideriamo $u^T A x \geq u^T b$:

$$\begin{aligned} u^T A x &= [u_1 | u_2 | \dots | u_i | \dots | u_n] \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_i^T x \\ \vdots \\ a_m^T x \end{bmatrix} = \\ &= u_1 a_1^T x + u_2 a_2^T x + \dots + u_i a_i^T x + \dots + u_m a_m^T x \\ u^T b &= u_1 b_1 + u_2 b_2 + \dots + u_i b_i + \dots + u_m b_m \end{aligned}$$

Si considera lo stesso valore di prima, ma confrontato con somme diverse. Per ricavarlo, ciascun addendo deve essere uguale al corrispondente termine nella seconda somma. Se questo accade, si preserva la disuguaglianza.

Affinché sia valido $u^T A x \geq u^T b$ è sufficiente che la disuguaglianza sia soddisfatta da ciascuno degli addendi, cioè

$$u_i a_i^T x \geq u_i b_i, \forall i = 1 \dots m$$

Quindi si considera un vincolo primale alla volta e:

- se $a_i^T x \geq b_i$, la condizione sufficiente è che $u_i \geq 0$ (vincolo primale di $\geq \Rightarrow$ variabile duale ≥ 0);
- se $a_i^T x \leq b_i$, la condizione sufficiente è che $u_i \leq 0$ (vincolo primale di $\leq \Rightarrow$ variabile duale ≤ 0);
- se $a_i^T x = b_i$, la condizione $u_i a_i^T x \geq u_i b_i$ è soddisfatta all'uguaglianza per ogni $u_i \in \mathbb{R}$ (vincolo primale di uguaglianza $\Rightarrow$ variabile duale libera).

Riassumendo tutto questo, abbiamo ricavato un set di condizioni sufficienti affinché $w \leq c^T x$ sia rispettata $\forall x \in P$. Possiamo generalizzare queste condizioni per ottenere il problema duale corrispondente, considerando *singolarmente* ciascun vincolo.

Si può dimostrare che queste condizioni sono anche necessarie e generalizzare le regole per ottenere il problema duale corrispondente ad un problema di programmazione lineare in forma generica.

(Importante)

Queste regole valgono *partendo da un problema primale in forma di minimo*.

Quando ci sono problemi di max, si parta *dalla colonna di destra e si legga la tabella da destra a sinistra*.

In un problema primale di massimo:

- Una variabile primale ≥ 0 genera vincolo duale di $\leq$
- Un vincolo primale di $\leq$ corrisponde ad una variabile duale ≥ 0

Primale ($\min c^T x$)	Duale ($\max u^T b$)
$a_i^T x \geq b_i$	$u_i \geq 0$
$a_i^T x \leq b_i$	$u_i \leq 0$
$a_i^T x = b_i$	u_i libera
$x_j \geq 0$	$u^T A_j \leq c_j$
$x_j \leq 0$	$u^T A_j \geq c_j$
x_j libera	$u^T A_j = c_j$

Esempio di scrittura del problema duale:

Esempio 1

min	$10x_1 + 20x_2 + 0 \cdot x_3$	
s.t.	$2x_1 - x_2 \geq 1$	u_1
	$x_2 + x_3 \leq 2$	u_2
	$x_1 - 2x_3 = 3$	u_3
	$3x_2 - x_3 \geq 4$	u_4
	$x_1 \geq 0$	
	$x_2 \leq 0$	
	x_3 libera	

$$\begin{aligned}
 \max \quad & 1 \cdot u_1 + 2 \cdot u_2 + 3 \cdot u_3 + 4 \cdot u_4 \\
 \text{s.t.} \quad & 2 \cdot u_1 + 0 \cdot u_2 + 1 \cdot u_3 + 0 \cdot u_4 \leq 10 \\
 & -u_1 + u_2 + 3u_3 \geq 20 \\
 & +u_2 - 2u_3 - u_4 = 0 \\
 & u_1 \geq 0 \quad u_2 \leq 0 \quad u_3 \text{ libera} \quad -u_4 \geq 0
 \end{aligned}$$

Note:

- I termini u_i sono l'insieme dei termini noti (quindi, 4 termini noti con rispettivi coefficienti)
- Come detto sopra, si ragiona a colonne, quindi:
 - 1) abbiamo x_1 nelle posizioni di u_1 e $u_3 \leq 10$ (valore della prima colonna). Il segno è $\leq$ (in quanto passiamo da min a max e il $\geq$ diventa $\leq$)
 - 2) abbiamo x_2 rispetto a $u_1, u_2, u_4 \leq 20$ (valore della seconda colonna). Il segno è $\geq$ (in quanto passiamo da min a max e il $\leq$ diventa $\geq$)
 - 3) abbiamo x_3 libera (quindi, avremo $=$ come segno) e si ragiona in colonna come prima.
- Si fissano i domini sulla base di tutto quanto appena visto

Esercizio 1 Si scriva il problema duale del seguente problema:

$$\begin{aligned}
 \max \quad & x_1 + 2x_2 + 3x_3 + 4x_4 \\
 \text{s.t.} \quad & 2x_1 + x_3 \leq 10 \\
 & -x_1 + x_2 + 3x_4 \geq 20 \\
 & x_2 - 2x_3 - x_4 = 0 \\
 & x_1 \geq 0 \\
 & x_2 \leq 0 \\
 & x_3 \text{ libera} \\
 & x_4 \geq 0
 \end{aligned}$$

Suggerimento: si può procedere in due modi *equivalenti*: applicare la tabella data da destra a sinistra (ottenendo il duale in forma di minimo) oppure trasformare la funzione obiettivo in forma di minimo, applicare le regole da sinistra a destra (ottenendo il duale in forma di massimo), ricondurre la funzione obiettivo del duale in forma di minimo.

Viene da noi affrontato nelle slide:

Esempio 2

max	x_1	$+2x_2$	$+3x_3$	$+4x_4$	
s.t.	$2x_1$	$+x_2$	$+3x_4$		≤ 10
	$-x_1$	$+x_2$	$-2x_3$	$+3x_4$	≥ 20
	x_1	x_2	x_3	x_4	$= 0$
					≥ 0
					libera
					≥ 0

Primale (min $c^T x$)	Duale (max u^T)
$a_i^T x \geq b_i$	$u_i \geq 0$
$a_i^T x \leq b_i$	$u_i \leq 0$
$a_i^T x = b_i$	u_i libera
$x_j \geq 0$	$u^T A_j \leq c_j$
$x_j \leq 0$	$u^T A_j \geq c_j$
x_j libera	$u^T A_j = c_j$

Handwritten dual problem:

$$\min 10y_1 + 20y_2$$

$$\text{s.t. } 2y_1 - y_2 \geq 1$$

$$y_2 + y_3 \leq 2$$

$$y_1 - 2y_3 = 3$$

$$3y_2 - y_3 \geq 4$$

$$y_1 \geq 0, y_2 \leq 0, y_3 \text{ libera}$$

Note:

- I termini u_i sono l'insieme dei termini noti (quindi, 2 termini noti con rispettivi coefficienti)
- Come detto sopra, si ragiona a colonne, quindi:
 - 1) abbiamo x_1 nelle posizioni di u_1 e $u_2 \geq 1$ (valore della prima colonna). Il segno è $\geq$ (in quanto passiamo da max a min e il $\leq$ diventa $\geq$)
- Avendo già descritto approfonditamente il funzionamento, simile rimane per i casi (2), (3), (4)
- Si fissano i domini sulla base di tutto quanto appena visto

A seguito di questi due esempi/esercizi:

- se prendiamo il problema primale e ne scriviamo il duale, otterremo un altro problema duale
- se poi prendiamo il problema duale risultato e ne scriviamo il duale a sua volta, otterremo il problema primale di partenza

A conferma di quanto appena detto, introduciamo una serie di teoremi sulla dualità. Quanto appena descritto, è la *trasformazione duale doppia*.

Teoremi della dualità

$$x^T y = (y^T x)^T$$

Teorema: trasformazione duale doppia

Il duale del duale è il primale

Dimostrazione:

$$\left\{ \begin{array}{l} \min c^T x \\ \text{s.t. } Ax \geq b \\ x \geq 0 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \max u^T b \\ \text{s.t. } u^T A \leq c^T \\ u \geq 0 \end{array} \right\} \equiv \left\{ \begin{array}{l} \max b^T u \\ \text{s.t. } A^T u \leq c \\ u \geq 0 \end{array} \right\} \rightarrow$$

$$\left\{ \begin{array}{l} \min c^T y \\ \text{s.t. } y^T A^T \geq b^T \\ y \geq 0 \end{array} \right\} \equiv \left\{ \begin{array}{l} \min c^T y \\ \text{s.t. } Ay \geq b \\ y \geq 0 \end{array} \right\} \equiv \left\{ \begin{array}{l} \min c^T x \\ \text{s.t. } Ax \geq b \\ x \geq 0 \end{array} \right\} \quad \blacksquare$$

Data una coppia di problemi, se il primo ammette soluzione ottima finita, anche l'altro ammette soluzione ottima finita e la soluzione è la stessa (in quanto, in base alle ipotesi discusse sopra, avremo che $z^* = w^*$, quindi i valori delle f.o. coincidono). Tale idea è nota come *teorema della dualità forte* (questo sotto, per i problemi in forma generica)

Esso è normalmente usato alla fine degli esercizi di dualità in esame (come verifica; se le due f.o. hanno valore uguale sostituendo i loro valori, abbiamo conferma della presenza della soluzione ottima finita).

Teorema 2 (Dualità forte per problemi in forma generica): *Sia data una coppia di problemi primale-duale indicati con (PL) e (DL) rispettivamente, e sia la regione ammissibile di (PL) un poliedro non vuoto. Allora (PL) ammette soluzione ottima finita se e solo se (DL) ammette soluzione ottima finita e i valori delle funzioni obiettivo coincidono.*

Data coppia (PL) e (DL)

(PL) ammette **ottimo finito** $z^* \iff$ (DL) ammette **ottimo finito** w^*
e, inoltre, $z^* = w^*$

Dimostrazione:

$\Rightarrow$ per costruzione, $z^* = w^*$

$\Leftarrow$ per costruzione del duale applicata al duale, visto che il duale del duale è il primale ■

La dualità forte si applica *solo* se il problema primale ha soluzione ottima *finita*. Vogliamo però delle proprietà applicabili alle coppie primale-duale sapendo che uno dei due problemi è illimitato oppure impossibile.

Mantenendo la condizione di ammissibilità e rilassando la condizione di limitatezza della soluzione ottima del problema primale, si ha che:

“Il valore della f.o. del duale è sempre $\leq$ rispetto al valore della f.o. del primale”.

Tale affermazione ha riscontro nel *teorema della dualità debole*.

Teorema 4 (Dualità debole): *Siano P e D i poliedri delle regioni ammissibili dei problemi primale e duale rispettivamente: $P = \{x \geq 0 : Ax \geq b\}$ e $D = \{u \geq 0 : u^T A \leq c^T\}$. Sia inoltre $P \neq \emptyset$ e $D \neq \emptyset$. Allora, per ogni coppia di soluzioni ammissibili per il primale e per il duale $x \in P$ e $u \in D$, vale la relazione:*

$$u^T b \leq c^T x$$

Ricordando che i **risultati possono essere generalizzati**, fissiamo le idee:

$$\begin{array}{ll} (PL) & \begin{array}{l} z^* = \min c^T x \\ \text{s.t. } Ax \geq b \\ x \geq 0 \end{array} \end{array} \quad \begin{array}{ll} (DL) & \begin{array}{l} w^* = \max u^T b \\ \text{s.t. } u^T A \leq c^T \\ u \geq 0 \end{array} \end{array}$$

Siano $P = \{x \geq 0 : Ax \geq b\}$ e $D = \{u \geq 0 : u^T A \leq c^T\}$

Se $P \neq \emptyset$ e $D \neq \emptyset$, allora $\forall x \in P, u \in D$ si ha $u^T b \leq c^T x$

Dimostrazione:

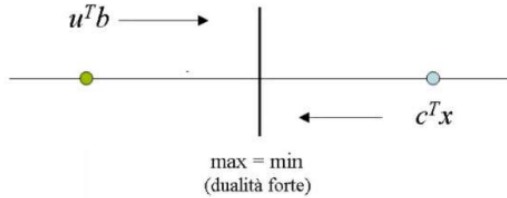
$$\begin{array}{ccccc} u^T b & \leq & u^T Ax & \leq & c^T x \\ \begin{array}{l} u^T \geq 0 \\ b \leq Ax \end{array} & & & \begin{array}{l} x \geq 0 \\ u^T A \leq c^T \end{array} & \end{array}$$

[Nota: per costruzione del duale, maggiorazioni **sempre** rispettate!] ■

Come conseguenza, i valori delle soluzioni primale e duale si limitano a vicenda, ponendosi come bound l'uno dell'altro. In particolare:

- i valori ammissibili del duale sono un *lower bound* per il primale
- i valori ammissibili del primale sono un *upper bound* per il duale

Come si vede, poi, ammettono lo stesso valore in base alle considerazioni sopra.



Idea concreta rispetto a quanto appena descritto:

Ammettiamo di avere un problema primale difficile e otteniamo un problema duale difficile anch'esso; supponiamo di avere una soluzione ammissibile e di conoscere almeno il valore della f.o (mettiamo 220). Quando abbiamo questo valore, sappiamo per certo che il valore della soluzione ammissibile non sarà più piccolo di 220.

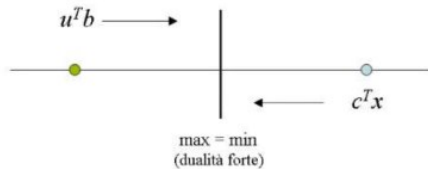
Corollario: condizione di ottimalità

Date soluzioni x ammissibile primale e u ammissibile duale
 $c^T x = u^T b \iff x$ e u ottime primale e duale risp.

Dimostrazione:

$\Rightarrow$ soluzioni migliori violerebbero il teorema della dualità debole

$\Leftarrow$ per il Teorema della dualità forte



Dualità e Programmazione Lineare: Condizioni di ottimalità/complementarietà ed esercizi

Altro corollario: caso illimitato. Quando si ha una soluzione illimitata, dualmente non esiste un problema che genera una soluzione illimitata e, dunque, l'altro è inammissibile.

Corollario 2 Sia data una coppia di problemi primale-duale (PL)-(DL). Allora:

(i) Se (PL) è illimitato, allora (DL) è inammissibile.

(ii) Se (DL) è illimitato, allora (PL) è inammissibile.

Dato che entrambi sono problemi di PL, sono possibili una serie di risultati (considerando però che valgono tutte le implicazioni precedenti, in particolare il teorema della dualità forte)

		(DL)		
		Finito	Illimitato	Impossibile
(PL)	Finito	SI e ($z^* = \omega^*$)	NO	NO
	Illimitato	NO	NO	SI (corollario)
	Impossibile	NO	SI (corollario)	SI (esempi)

Si ribadisce che le due implicazioni (i) e (ii) valgono solo nella direzione data. Esistono infatti casi di coppie di problemi primale-duale in cui entrambi i problemi sono inammissibili, come ad esempio:

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & x_1 + x_2 \geq 1 \\ & -x_1 - x_2 \geq 1 \\ & x_1, x_2 \text{ libere} \end{array} \quad \begin{array}{ll} \max & u_1 + u_2 \\ \text{s.t.} & u_1 - u_2 = 1 \\ & u_1 - u_2 = 0 \\ & u_1, u_2 \geq 0 \end{array}$$

(Non richiesto all'esame) → Interpretazione economica di coppie di problemi primale-duale (pagg. 15/16 dispense)

I teoremi della dualità possono essere utilizzati per definire delle condizioni di ottimalità per una coppia primale-duale che possano essere impiegate operativamente, per determinare l'ottimalità di soluzioni disponibili o per derivare degli algoritmi risolutivi per problemi di ottimizzazione.

Data una coppia di problemi primale/duale e due vettori $\bar{x}, \bar{u} \in R^n$ in qualsiasi forma, anche se $\bar{x}$ non è soluzione di base. Ciò serve a trovare l'ottimo di un problema nella forma $\{\min/\max c^T x: Ax = b, x \in R\}$

$$\begin{array}{ll} (PP) & \min c^T x \\ & \text{s.t. } Ax \geq b \\ & x \geq 0 \end{array} \quad \begin{array}{ll} (PD) & \max u^T b \\ & \text{s.t. } u^T A \leq c^T \\ & u \geq 0 \end{array}$$

I teoremi della dualità forniscono delle *condizioni di ottimalità*, al fine di determinare l'ottimalità di soluzioni o risolvere problemi di ottimizzazione.

$$\begin{array}{ll} \bar{x} \text{ e } \bar{u} \text{ ottime} & \iff \begin{array}{l} \bar{x} \text{ è ammissibile primale: } A\bar{x} \geq b \wedge \bar{x} \geq 0 \\ \bar{u} \text{ è ammissibile duale: } \bar{u}^T A \leq c^T \wedge \bar{u} \geq 0 \\ \text{vale la dualità forte: } c^T \bar{x} = \bar{u}^T b \end{array} \\ \text{primale e duale (risp.)} & \end{array}$$

Le condizioni possono essere estese a coppie di problemi in qualsiasi forma, non fanno nessuna assunzione sul tipo di soluzioni ammissibili e possono essere applicate, ad esempio, anche a soluzioni NON di base. In alcuni casi, queste condizioni possono essere sfruttate per calcolare la soluzione ottima di un problema di programmazione lineare, come nel seguente esempio:

Esempio 3 Trovare la soluzione ottima del seguente problema di programmazione lineare:

$$\begin{array}{ll} \max & z = -3x_1 - x_3 \\ \text{s.t.} & x_1 + 2x_2 + x_3 = 7 \\ & 2x_1 + x_2 + x_3 = 20 \\ & x_1, x_2, x_3 \text{ libere} \end{array}$$

Scriviamo il duale del problema:

$$\begin{array}{ll} \min & 7u_1 + 20u_2 \\ \text{s.t.} & u_1 + 2u_2 = -3 \\ & 2u_1 + u_2 = 0 \\ & u_1 + u_2 = -1 \\ & u_1 \text{ libera, } u_2 \text{ libera} \end{array}$$

La presenza di soli vincoli di uguaglianza e di sole variabili libere suggerisce l'applicazione diretta delle condizioni di ottimalità primale duale, impostando un sistema di equazioni lineari contenente i due vincoli del primale (di uguaglianza), i tre vincoli del duale (variabili primali libere $\Rightarrow$ vincoli del duale di uguaglianza) e il vincolo di uguaglianza tra la f.o. primale e f.o. duale.

Si ottiene pertanto il seguente sistema con 6 equazioni in 5 incognite:

$$\begin{cases} x_1 + 2x_2 + x_3 = 7 \\ 2x_1 + x_2 + x_3 = 20 \\ u_1 + 2u_2 = -3 \\ 2u_1 + u_2 = 0 \\ u_1 + u_2 = -1 \\ -3x_1 - x_3 = 7u_1 + 20u_2 \end{cases}$$

Si può pertanto applicare un qualsiasi metodo per la soluzione di sistemi di equazioni lineari. Si ottengono così, per le variabili primali e duali, dei valori corrispondenti a soluzioni ammissibili che verificano la dualità forte, e che sono, quindi, ottime.

Applicando un qualsiasi metodo per la soluzione, troveremo infinite soluzioni come segue. x_3 è grado di libertà e, fissandolo ad un qualsiasi valore arbitrario, tutte le soluzioni sono ottime ed ammissibili:

$$x_1 = 11 - \frac{1}{3}x_3, x_2 = -2 - \frac{1}{3}x_3, u_1 = 1 \text{ e } u_2 = -2$$

(Si consideri, a livello generale, che il numero di gradi di libertà di un sistema rappresenta la sua varianza, ovvero il numero di variabili indipendenti necessarie per determinare completamente lo stato del sistema. In questo contesto indica semplicemente "le variabili che fissate mi danno certi valori e risolvono il sistema sulla base delle condizioni presenti".

Sostituiamo l'uguaglianza delle funzioni obiettivo con altre equazioni con le stesse implicazioni.

Nel caso di vincoli di non uguaglianza/variabili non libere, valgono le condizioni di ortogonalità/condizioni di complementarità primale duale (CCPD)/scarti complementari:

Teorema 5 Condizioni di ortogonalità (o di complementarità) primale-duale. *Data la coppia di problemi primale-duale*

$$\begin{aligned} & \min\{c^T x : x \geq 0, Ax \geq b\} \\ & \max\{u^T b : u \geq 0, u^T A \leq c^T\} \end{aligned}$$

$$x \text{ e } u \text{ ottime primale e duale (risp.)} \iff \left. \begin{aligned} & Ax \geq b \wedge x \geq 0 && (\text{ammissibilità primale}) \\ & u^T A \leq c^T \wedge u \geq 0 && (\text{ammissibilità duale}) \\ & u^T (Ax - b) = 0 \\ & (c^T - u^T A)x = 0 \end{aligned} \right\} \quad (\text{ortogonalità})$$

(Si chiama condizione di ortogonalità in quanto il prodotto dei vettori è 0).

I valori delle f.o. devono essere uguali e, partendo da un problema primale e duale:

- il vettore delle variabili primali è ortogonale al vettore delle variabili duali
- il vettore delle variabili duali è ortogonale al vettore delle variabili primali

In quanto x, u ottime $\Rightarrow x, u$ ammissibili primale e duale e possiamo scrivere:

$$\begin{array}{ccc} u^T b & \leq & u^T Ax & \leq & c^T x \\ u^T \geq 0 & & x \geq 0 & & \\ b \leq Ax & & u^T A \leq c^T & & \end{array}$$

In quanto ottime $\Rightarrow u^T b = c^T x$ Per la dualità forte $\Rightarrow u^T b = u^T Ax = c^T x$

Le due disuguaglianze sopra riportate devono pertanto essere rispettate necessariamente all'uguaglianza:

$$\begin{aligned} u^T Ax = u^T b & \Rightarrow u^T (Ax - b) = 0 \\ c^T x = u^T Ax & \Rightarrow (c^T - u^T A)x = 0 \end{aligned}$$

In merito alla sufficienza, dalle condizioni di ortogonalità si ricava $u^T b = c^T x$:

$$\begin{aligned} u^T (Ax - b) = 0 & \Rightarrow u^T b = u^T Ax \\ (c^T - u^T A)x = 0 & \Rightarrow c^T x = u^T Ax \end{aligned}$$

Abbiamo quindi due soluzioni x e u ammissibili primale e duale, rispettivamente, i cui valori delle funzioni obiettivo coincidono. Le due soluzioni sono ottime, date tutte le condizioni.

Sviluppando i prodotti tra matrici, le condizioni di ortogonalità di due soluzioni ammissibili e ottime x e u possono essere scritte come segue.

$$\begin{aligned} u^T(Ax - b) &= \sum_{i=1}^m u_i(a_i^T x - b_i) = 0 \in \mathbb{R} \\ (c^T - u^T A)x &= \sum_{j=1}^n (c_j - u^T A_j)x_j = 0 \in \mathbb{R} \end{aligned}$$

Vedendo tutti gli addendi, ciascuno è ≥ 0 e la loro somma è $= 0$.
Dunque, ogni addendo è $= 0$, sulla base di m equazioni (e vale l'ortogonalità)

Ricordando che, per l'ammissibilità dei problemi primale e duale TUTTI i fattori sono ≥ 0 , si ha che, all'ottimo:

$$\begin{aligned} u_i(a_i^T x - b_i) &= 0, \quad \forall i = 1 \dots m \\ (c_j - u^T A_j)x_j &= 0, \quad \forall j = 1 \dots n \end{aligned}$$

Le condizioni di ortogonalità (o complementarietà) sono rispettate per ciascun vincolo/variabile primale/duale.

Quanto dicono queste condizioni è che: "possedendo una soluzione ottima primale/duale, sicuramente il vincolo primale/duale è soddisfatto all'uguaglianza".

L'insieme di condizioni estese di complementarietà primale-duale è quella dell'immagine seguente (estensibile a coppie primale-duale in qualsiasi forma). Le condizioni possono essere sfruttate per verificare se una soluzione data sia ottima senza usare algoritmi specifici (es. metodo del simplesso).

Teorema (condizioni "estese")

Data **qualsiasi** coppia di problemi primale duale e $x \in \mathbb{R}^n$ $u \in \mathbb{R}^m$

$$\begin{array}{lcl} x \text{ e } u & \iff & x \text{ ammissibile primale} \\ \text{ottime} & & u \text{ ammissibile duale} \\ & & u_i(a_i^T x - b_i) = 0, \quad \forall i = 1 \dots m \\ & & (c_j - u^T A_j)x_j = 0, \quad \forall j = 1 \dots n \end{array}$$

- Esempio: (PP) $\min\{c^T x : x \geq 0, Ax \geq b\}$
(PD) $\max\{u^T b : x \geq 0, u^T A \leq c^T\}$

1) variabile primale $\neq 0$	$x_j > 0 \Rightarrow u^T A_j = c_j$	vincolo duale <i>saturo</i>
2) vincolo duale <i>lasco</i>	$u^T A_j < c_j \Rightarrow x_j = 0$	variabile primale nulla
3) variabile duale $\neq 0$	$u_i > 0 \Rightarrow a_i^T x = b_i$	vincolo primale <i>saturo</i>
4) vincolo primale <i>lasco</i>	$a_i^T x > b_i \Rightarrow u_i = 0$	variabile duale nulla

solo nel verso $\Rightarrow$!

Se il problema primale è in forma standard, le condizioni 3 e 4 non sono da considerarsi, perché già contenute nell'ammissibilità primale (dato che $u^T(Ax - b) = 0$ è soddisfatta $\forall u \in \mathbb{R}^n$, poiché abbiamo $Ax = b$ per le soluzioni ammissibili primali).

In questo modo, sfruttiamo queste condizioni per verificare l'ottimalità di soluzioni senza usare algoritmi appositi come il simplesso.

Esercizio 1. Dato il problema

$$\begin{array}{ll} \min & 2x_1 - x_2 \\ \text{s.t.} & x_1 + 2x_2 \geq 7 \\ & 2x_1 - x_2 \geq -6 \\ & -3x_1 + 2x_2 \geq 8 \\ & x_1, x_2 \geq 0 \end{array}$$

applicare le condizioni di complementarità primale-duale per verificare se la soluzione $[x_1, x_2] = [0, 6]$ è ottima.

Per verificare se la soluzione è ottima dobbiamo verificare che sia una soluzione primale ammissibile e che sia possibile trovare una soluzione del problema duale che sia ammissibile e in scarti complementari con la soluzione primale data.

1. Verifica dell'ammissibilità primale della soluzione data:

$$\begin{aligned} x_1 + 2x_2 &= 0 + 2 \cdot 6 = 12 \geq 7 \text{ (OK)} \\ 2x_1 - x_2 &= 2 \cdot 0 - 6 = -6 \geq -6 \text{ (OK)} \\ -3x_1 + 2x_2 &= -3 \cdot 0 + 2 \cdot 6 = 12 \geq 8 \text{ (OK)} \\ x_1 = 0 \geq 0, x_2 = 6 \geq 0 &\text{ (domini OK)} \end{aligned}$$

In questo caso, sostituisco i valori di $x_1 = 0, x_2 = 6$, eseguo le somme algebriche e verifica sia $\geq$ al termine noto

2. Passaggio al duale:

$$\begin{array}{ll} \max & 7u_1 - 6u_2 + 8u_3 \\ \text{s.t.} & u_1 + 2u_2 - 3u_3 \leq 2 \\ & 2u_1 - u_2 + 2u_3 \leq -1 \\ & u_1, u_2, u_3 \geq 0 \end{array}$$

Quanto si vede a lato è quanto visto nella lezione precedente: si prendono i termini noti (u_i), poi si risolve per colonne e si pongono tutti i vincoli del caso (in questo problema, tutti ≥ 0)

1) Applico CCPD (Condizioni Complementarietà Primale-Duale)

Quello che si deve fare è prendere gli u_i e moltiplicarli alle variabili x_i poste all'uguaglianza, sia per il problema primale che per il duale.

Attenzione che si considerano i vincoli se il risultato tra parentesi se u_i oppure x_i è già moltiplicato per qualcosa > 0 .

I primi tre sono quelli primali:

- $u_1(x_1 + 2x_2 - 7) = 0 \Rightarrow 0 + 12 - 5 \Rightarrow u_1(5) = 0$ (u_1 deve essere uguale a 0, prima condizione)
- $u_2(2x_1 - x_2 + 6) = u_2(0 - 6 + 6) \Rightarrow u_2(0)$ (Il risultato tra parentesi vale già 0, quindi non posso dedurre condizioni di complementarità [cioè, che u_2 valga un numero che mi azzeri il risultato])
- $u_3(-3x_1 + 2x_2 - 8) = u_3(0 + 12 - 8) \Rightarrow u_3(4) = 0 \Rightarrow u_3 = 0$ (u_3 deve essere uguale a 0, seconda condizione)

I secondi due sono quelli duali all'uguaglianza posti a 0 e moltiplicati per la corrispondente variabile primale:

- $(u_1 + 2u_2 - 3u_3 - 2)x_1 = 0 \Rightarrow (u_1 + 2u_2 - 3u_3 - 2)0 = 0$ (non deduco condizioni di complementarità per i motivi detti sopra)
- $(2u_1 - u_2 + 2u_3 + 1)x_2 = 0 \Rightarrow (2u_1 - u_2 + 2u_3 + 1)6 \Rightarrow 2u_1 - u_2 + 2u_3 = -1$ (terza condizione)

2) Sistema di tutte le disuguaglianze che derivano da CCPD e dalle condizioni di ammissibilità duale

$$\begin{cases} u_1 = 0 & (\text{ccpd}) \\ u_3 = 0 & (\text{ccpd}) \\ 2u_1 - u_2 + 2u_3 = -1 & (\text{ccpd}) \end{cases} \Rightarrow \begin{cases} u_1 = 0 \\ u_2 = 1 \\ u_3 = 0 \end{cases}$$

In pratica, prendiamo i valori di u_i trovati in precedenza e la condizione duale trovata. Poi, per rendere valida l'uguaglianza, sostituiamo i valori che abbiamo (quindi, $u_1, u_3 = 0, u_2 = 1$)

- 3) Verifica ammissibilità duale (quindi, se vale rispetto al dominio del duale e se vale numericamente rispetto alle disuguaglianze del problema duale).

Nello specifico, significa che:

- I valori di u_1, u_2, u_3 sono dentro ai loro domini
- Quando sostituisco u_1, u_2, u_3 rispetto i valori che hanno dentro i vincoli duali

la soluzione duale trovata:

- soddisfa i vincoli di dominio ($u_1, u_2, u_3 \geq 0$);
- soddisfa i due vincoli duali ($u_1 + 2u_2 - 3u_3 = 2 \leq 2$, $2u_1 - u_2 + 2u_3 = -1 \leq -1$);

Correttamente, u è ammissibile duale.

4) Conclusioni

Abbiamo a disposizione una soluzione primale x e una soluzione duale u tali che:

- x è ammissibile primale (come da verifica);
- u è ammissibile duale (come da verifica);
- x e u sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale.

Scarti complementari → Si è dimostrata la condizione di ortogonalità precedentemente detta.

Esercizio 5. Dato il problema

$$\begin{array}{llllll}
 \max & 2x_1 & + & x_2 & & \\
 \text{s.t.} & x_1 & + & x_2 & & \leq 4 \quad u_1 \\
 & 2x_1 & - & x_2 & + & x_3 \geq 8 \quad u_2 \\
 & x_1 & + & 2x_2 & - & x_3 = -1 \quad u_3 \\
 & x_1 & + & x_2 & + & 2x_3 = 4 \quad u_4 \\
 & x_1 & , & & & x_3 \text{ libere} \\
 & & & x_2 & & \leq 0
 \end{array}$$

applicare le condizioni di complementarità primale-duale per verificare se la soluzione $[x_1, x_2, x_3] = [9, -5, 0]$ è ottima.

Per verificare se la soluzione proposta è ottima dobbiamo verificare che sia una soluzione primale ammissibile e che sia possibile trovare una soluzione del problema duale che sia ammissibile e in scarti complementari con la soluzione primale data.

Il problema dirà "Enunciare le CCPD e usarle per dimostrare che $x_1 = 9, x_2 = -5, x_3 = 0$ è ottima". (Si può enunciare anche solo coi prodotti scalari, ma non solo con la condizione di uguaglianza della f.o.)

1) Verifica ammissibilità primale

$$\begin{aligned}
 x_1 + x_2 &= 9 - 5 = 4 \leq 4 \quad (\text{OK}) \\
 2x_1 - x_2 + x_3 &= 2 \cdot 9 + 5 + 0 = 23 \geq 8 \quad (\text{OK}) \\
 x_1 + 2x_2 - x_3 &= 9 + 2 \cdot (-5) - 0 = -1 \quad (\text{OK}) \\
 x_1 + x_2 + 2x_3 &= 9 - 5 + 2 \cdot 0 = 4 \quad (\text{OK}) \\
 x_2 &= -5 \leq 0 \quad (\text{domini OK})
 \end{aligned}$$

Come prima, vado a sostituire i valori di x_1, x_2, x_3 nelle disuguaglianze e verifico se sono rispettate tutte le disuguaglianze (ammissibilità).

2) Passaggio al duale

$$\begin{array}{ll}
\min & 4u_1 + 8u_2 - u_3 + 4u_4 \\
\text{s.t.} & u_1 + 2u_2 + u_3 + u_4 = 2 \\
& u_1 - u_2 + 2u_3 + u_4 \leq 1 \\
& u_2 - u_3 + 2u_4 = 0 \\
& u_1 \geq 0 \\
& u_2 \leq 0 \\
& u_3, u_4 \text{ libere}
\end{array}$$

Ora, passiamo alla funzione di minimo considerando i 4 termini noti come u_i , prendendo x_i in colonna rispetto a ciascuno, ponendo il dominio duale (considerando il passaggio min-max, $\leq$ diventa $\geq$ e viceversa).

3) Applicazione delle condizioni di complementarità primale-duale (CCPD)

Esaminiamo i vincoli di ammissibilità primale:

- $u_1(x_1 + x_2 - 4) = 0 \rightarrow u_1(0) = 0 \rightarrow //$;
- $u_2(2x_1 - x_2 + x_3 - 8) = 0 \rightarrow u_2(15) = 0 \rightarrow u_2 = 0$ (prima condizione);
- il terzo vincolo primale è di uguaglianza, quindi non ci sono da imporre condizioni di complementarità con la variabile duale u_3 , visto che la condizione $u_3(x_1 + 2x_2 - x_3 + 1) = 0$ vale per ammissibilità primale;
- il quarto vincolo primale è di uguaglianza, quindi non ci sono da imporre condizioni di complementarità con la variabile duale u_4 , visto che la condizione $u_4(x_1 + x_2 + 2x_3 - 4) = 0$ vale per ammissibilità primale.

Nei casi visti sopra, a parte u_1 su cui, per i ragionamenti di sopra, non si hanno informazioni, per gli altri abbiamo che, nel caso dei vincoli di uguaglianza, non si possono imporre condizioni di complementarità (perché avendo già l'uguaglianza, è inutile andare a mettere $=0$)

Esaminiamo i vincoli di ammissibilità duale:

- $u_1 + 2u_2 + u_3 + u_4 = 2 \rightarrow$ Vero per ammissibilità duale (perché è già in forma di uguaglianza, infatti $(u_1 + 2u_2 + u_3 + u_4 - 2)x_1 = 0$)
- $(u_1 - u_2 + 2u_3 + u_4 - 1)x_2 = 0 \Rightarrow (u_1 - u_2 + 2u_3 + u_4 - 1)(-5) \Rightarrow u_1 - u_2 + 2u_3 + u_4 = 1$ (seconda condizione)
- $u_2 - u_3 + 2u_4 = 0 \rightarrow$ Vero per ammissibilità duale (perché è già in forma di uguaglianza, infatti $(u_2 - u_3 + 2u_4 - 0)x_3 = 0$)

4) Sistema delle equazioni da CCPD e da ammissibilità duale

Andiamo ora a sostituire tutto sulla base di tutte le osservazioni precedenti:

$$\begin{array}{l}
\left\{ \begin{array}{ll} u_2 = 0 & (\text{ccpd}) \\ u_1 - u_2 + 2u_3 + u_4 = 1 & (\text{ccpd}) \\ u_1 + 2u_2 + u_3 + u_4 = 2 & (\text{ammiss. duale}) \\ u_2 - u_3 + 2u_4 = 0 & (\text{ammiss. duale}) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} u_2 = 0 \\ u_1 = 1 - 5u_4 \\ 1 - 5u_4 + 2u_4 + u_4 = 2 \\ u_3 = 2u_4 \end{array} \right. \Rightarrow \\
\Rightarrow \left\{ \begin{array}{l} u_1 = \frac{7}{2} \\ u_2 = 0 \\ u_3 = -1 \\ u_4 = -\frac{1}{2} \end{array} \right.
\end{array}$$

L'unica soluzione che soddisfa tutti i nostri vincoli (soddisfa alcune equazioni di ammissibilità duale e soddisfa gli scarti complementari)

5) Verifica ammissibilità duale

La soluzione duale trovata:

- soddisfa i tre vincoli duali ($u_1 + 2u_2 + u_3 + u_4 = 2$, $u_1 - u_2 + 2u_3 + u_4 \leq 1$, $u_2 - u_3 + 2u_4 = 0$);
- soddisfa i vincoli di dominio ($u_1 \geq 0, u_2 \leq 0, u_3, u_4$ libere).

Per capire se soddisfa i vincoli, letteralmente, andiamo a sostituire tutti gli u_i e verifichiamo se i valori di u_i rispettano i domini iniziali. Se ciò accade, passiamo tranquillamente alle conclusioni.

6. Conclusioni:

Abbiamo a disposizione una soluzione primale x e una soluzione duale u tali che:

- x è ammissibile primale (come da verifica);
- u è ammissibile duale (come da costruzione e da verifica);
- x e u sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale.

Dunque, i valori delle funzioni obiettivo sono uguali, pertanto sono entrambe ammissibili e in scarti complementari.

Laboratorio 5 – Scenari modello distribuzione PC, esempi scripting, esempio modello primale/duale

Ripartendo dall'esercizio del modello dei PC, consideriamo una serie di scenari:

Esercizio: distribuzione PC (scenari)

Per bilanciare la produzione, l'azienda richiede che nello stabilimento italiano si assemblino almeno il 25% dei PC.

1. Inoltre, l'azienda richiede che nello stabilimento italiano si assemblino **almeno il 30% dei PC (ipotesi 1)** **o il 40% dei PC (ipotesi 2)** prodotti in ciascuno degli altri stabilimenti
2. Produrre un elenco che permetta di individuare i casi in cui una banca riceve forniture da un solo paese.
3. Visualizzare l'utilizzo delle capacità produttive per paese.
4. Conviene, nell'ipotesi 2, potenziare di 5000 unità la produzione in Cina, al costo di 4.000 euro?
5. Tornare alla situazione senza bilanciamenti e studiare gli effetti della diminuzione (a intervalli di 6 euro) del costo di produzione in Italia (diminuzione massima di 40 euro), indicando in quali casi in Italia la produzione complessiva supera quella della Francia.

Insiemi: S (stabilimenti) e B (banche)

Parametri: w_i (costi prod.), c_{ij} (costi trasp.), a_i (capacità prod.), b_j (richieste), α (bilanciamento generale), β (bilanciamento singolo)

$$\begin{aligned}
 &\min \sum_{i \in S, j \in B} (w_i + c_{ij}) x_{ij} \\
 &\text{s.t.} \sum_{j \in B} x_{ij} \leq a_i \quad \forall i \in S \\
 &\quad \sum_{i \in S} x_{ij} = b_j \quad \forall j \in B \quad (\text{no eccessi di produzione} \Rightarrow \text{uguaglianza}) \\
 &\quad \sum_{j \in B} x_{\text{Italia } j} \geq \alpha \sum_{i \in S, j \in B} x_{ij} \\
 &\quad \sum_{j \in B} x_{\text{Italia } j} \geq \beta \sum_{j \in B} x_{ij} \quad \forall i \in S \setminus \{\text{Italia}\} \\
 &\quad x_{ij} \in \mathbb{Z}_+ \quad \forall i \in S, j \in B
 \end{aligned}$$

[Risorse] PC.mod
- PC.dat -
PC.run
(PC_plus.run)

Il file *pc.mod* viene modificato come segue:

```

set I; #stabilimenti - S
set J; #banche - B

param A{I}; #capacità produttive
param B{J}; #richieste
param W{I}; #costi produttivi
param C{I, J}; #costi
var x{I, J} integer >= 0;
param stab_bil symbolic in S; #Per mettere 'it' all'interno delle
variabili
param bil1;
param bil2;

minimize costo: sum{i in I, j in J}
(W[i] + C[i,j])*x[i,j];
s.t. produzione{i in I}: sum{j in J} x[i,j] <= A[i];
s.t. domanda{j in J}: sum{i in I} x[i,j] = B[j];

```

```
#Introduciamo i due vincoli di bilanciamento aggiuntivi
#Inseriamo 'it' perché è parte di un insieme, ma è una stringa
#Per fare in modo che AMPL non dipenda dal tipo stringa,
#inseriamo un parametro simbolico "stab_bil", tale che
#non venga visto solo come numero
s.t. bilan1: sum{j in J} x[stab_bil, j] >= 0.25 *
          sum{i in I, j in J} x[i, j];

#Attenzione che, a livello di insieme, deve essere escluso 'it'
s.t. bilan2{i in I: i != 'it'}: sum{j in J} x['it', j] >= 0.30 *
          sum{j in J} x[i, j];
```

La modifica del file *pc.dat* avviene come segue, introducendo anche i parametri:

```
set I := it ch fr;
set J := bi uc av cs bc;

param A := it 10000 ch 20000 fr 10000;
param B := bi 7100 uc 3400 av 9700 cs 5200 bc 3050;

param W := it 220 ch 180 fr 200;

#param :      w      a :=
#it      220    10000
#ch      100    20000
#fr      200    10000

param C:      bi      uc      av      cs      bc :=
it            5.5      7.5      6.9      8.0      10.3
ch            15       14.3     13.0     16.4     5.0
fr            6.0      7.8      6.3      6.8     11.0
;

param stab_bil := 'it';
param bil1 := 0.25;
param bil2 := 0.30;
```

Aggiorniamo includendo dei controlli di scripting all'interno del file *PC.run*, al fine di aggiungere una modifica alla variabile di bilanciamento e l'elenco delle forniture esclusive. Lo facciamo introducendo le direttive di scripting accennate nella lezione precedente.

```
Reset;
model pc.mod;
data pc.dat;
option solver cplex;
solve;
display costo, x;

display {crème in I, j in J: x[crème,j]>0} x[crème,j];
display {crème in I} (produzione[i].body, A[i]);

printf "\n\n***** IPOTESI *****\n";
let bil2 := 0.40;
solve;

display {i in I} (produzione[i].body, A[i]);
#Produrre un elenco che permetta di individuare i casi in cui una
#banca riceve forniture da un solo paese.
```

```

Printf "\n\n***** ELENCO FORNITURE ESCLUSIVE *****\n";
for {j in J, crème in I: x[crème,j]>0} {
    printf "%12s → %12s: %8.2f / %8.2f %c\n", crème, j, x[crème,j], B[j],
    if(x[crème,j]==B[j]) then '*' else '';
}

```

A titolo di completezza, inserisco anche gli altri punti, presenti già nel file zip soluzione del prof, giusto per vedere anche gli altri punti e ragionarvi nel caso lo si ritenga opportuno.

```

#Visualizzare l'utilizzo delle capacità produttive per paese
printf "\n\n* * * P U N T O   3 * * *\n";
printf "\nSITUAZIONE FORNITORI\n";
for {crème in S} {
    printf "%12s: %8.2f / %8.2f : %6.2f%\n", crème,
        produzione[i].body, capacita[i],
        produzione[i].body/capacita[i]*100;
}

```

#Conviene, nell'ipotesi 2, potenziare di 5000 unità la produzione in
#Cina, al costo di 4.000 euro?

```

printf "\n\n* * * P U N T O   4 * * *\n";
let bilan2 := 0.40; #considero lo scenario 2
printf "\n\nSCENARIO CON CAPACITA' PRODUTTIVA CINESE AUMENTATA DI 5000
UNITA'\n";
option solver_msg 0;
solve;
#deve essere dichiarato fuori dall'if: param old_fo;
let old_fo := costo_totale;
let capacita['Cina'] := capacita['Cina'] + scen4_aumento_produzione;
solve;
printf "\ncosto passa da %12.2f a %12.2f: \
    risparmio %12.2f - %12.2f = %12.2f\n",
    old_fo, costo_totale, old_fo - costo_totale,
    scen4_aumento_costo, old_fo - costo_totale - scen4_aumento_costo;
if old_fo-(costo_totale + scen4_aumento_costo) < 0 then { printf "NON "; }
printf "CONVIENE!\n\n";

```

```

printf "\nSITUAZIONE FORNITORI\n";
for {crème in S} {
    printf "%12s: %8.2f / %8.2f : %6.2f%\n", crème,
        produzione[i].body, capacita[i],
        produzione[i].body/capacita[i]*100;
}

```

#Tornare alla situazione senza bilanciamenti e studiare gli effetti della
#diminuzione (a intervalli di 6 euro) del costo di produzione in Italia
#(diminuzione massima di 40 euro), indicando in quali casi in Italia la
#produzione complessiva supera quella della Francia

```

printf "\n\n* * * P U N T O   5 * * *\n";

printf "\n\nCALCOLO EFFETTI COSTI ITALIA\n";
let bilan1 := 0.00;
let bilan2 := 0.00;
let diminuzione_max := 40;
let diminuzione_step := 6;
for { diminuzione in 0..diminuzione_max by diminuzione_step }
{
    let costo_produzione['Italia'] := costo_produzione['Italia'] -
    diminuzione;
    printf "\nCosto produzione Italia %8.2f (- %8.2f)\t",

```

```

costo_produzione['Italia'], diminuzione;
solve;
printf "\tcosto: %12.2f, Italia: %8.2f, Francia: %8.2f",
      costo_totale,
      produzione['Italia'].body,
      produzione['Francia'].body;
if ( produzione['Italia'].body > produzione['Francia'].body ) then
  printf "\t\tSUPERATO!";
else
  printf "\t\tNON SUPERATO";
let costo_produzione['Italia'] := costo_produzione['Italia'] +
diminuzione;
}

} # not infeasible

```

Ora consideriamo un esempio di problema primale/duale e verificare quanto fatto nella lezione di ieri (attenzione che non sono 1000 Kg di gianduia, ma di cacao, corretto brutalmente da me):

Dualità in AMPL: un esempio (testo)

- Modellare il seguente problema, trovare la soluzione ottima e analizzarla alla luce della teoria della dualità

Un'industria produce due tipi di creme: fondente e gianduia. Per avere un kg di ciascuna crema sono necessari, tra gli altri, due ingredienti grezzi (zucchero e cacao) e la lavorazione su una macchina, come riportato in tabella:

	Fondente	Gianduia
Zucchero (kg)	3	2
Cacao (kg)	4	1
Lavorazione (ore)	2	1

Settimanalmente, si hanno a disposizione al più 1200 Kg di zucchero e al più 1000 Kg di ^{cacao} ~~gianduia~~, mentre la disponibilità massima settimanale di ore lavorative della macchina è pari a 700. Un kg di fondente è venduto a 24 Euro e un kg di gianduia è venduto a 14 Euro. Si vuole pianificare la produzione settimanale in modo da massimizzare il ricavo complessivo.

Dualità in AMPL: un esempio (modelli)

PROBLEMA PRIMALE	PROBLEMA DUALE
max $24x_1 + 14x_2$	min $1200u_1 + 1000u_2 + 700u_3$
$3x_1 + 2x_2 \leq 1200$	$3u_1 + 4u_2 + 2u_3 \geq 24$
$4x_1 + x_2 \leq 1000$	$2u_1 + u_2 + u_3 \geq 14$
$2x_1 + x_2 \leq 700$	$u_1 \geq 0, u_2 \geq 0, u_3 \geq 0$
$x_1 \geq 0, x_2 \geq 0$	

Implementare il problema primale e il problema duale dell'esempio e:

- verificare il valore ottimo delle variabili primali e duali
- verificare il teorema della dualità forte
- verificare le condizioni di complementarità primale-duale
- vedere come cambiano i valori ottimi delle funzioni obiettivo primale e duale variando i termini noti dei vincoli primali, uno per volta
- dire se esiste una relazione tra il valore ottimo delle variabili duali e le variazioni osservate nel valore ottimo della funzione obiettivo

Associamo la modellazione e i corrispondenti problemi primale/duale. Partiamo dal problema primale. Creiamo il file *mixOpt2.mod* come modello generale di implementazione, utile ovviamente per risolvere anche questo problema (nel caso del problema delle creme è comunque adatto), comunque molto simile al modello di mix ottimo di produzione:

```

set PRODOTTI; #I
set RISORSE; #J
param prezzo{PRODOTTI}; #p{I}
param consumo_risorse{RISORSE, PRODOTTI}; #A{J,I}
param disp{RISORSE}; #d{J}
var x{PRODOTTI}; #x{I}
maximize ricavo: sum{j in PRODOTTI} prezzo[j] * x[j];
s.t. v_disp{i in RISORSE}:
    sum{j in PRODOTTI} consumo_risorse[i,j] * x[j]
    <= disp[i];

```

Listiamo il file *mixOpt_creme.dat* per il modello in uso e un file *mixOpt_duale.run*:

```

set PRODOTTI := fondente gianduia;
set RISORSE := zucchero cacao lavoro;

param prezzo := fondente 24 gianduia 14;
param consumo_risorse :      fondente      gianduia :=
zucchero                3                2

```



```
cacao          4          1
lavoro         2          1
;
param disp := zucchero 1200 cacao 1000 lavoro 700;
```

Segue il file .run:

```
reset;
option solver cplex;
model mixopt2.mod;
data mixopt_creme.dat;
solve;
display ricavo, x;
```

Scriviamo ora il problema duale nel file *duale.mod*:

```
set PRODOTTI;
set RISORSE;
param prezzo{PRODOTTI};
param consumo_risorse{RISORSE, PRODOTTI};
param disp{RISORSE};
var u{RISORSE} >= 0; #per il duale, si deve inserire  $u_i$ 
minimize costo: sum{i in RISORSE} disp[i]*u[i]; #costo relativo ad  $u_i$ 
s.t. convenienza{j in PRODOTTI}:
    sum{i in RISORSE} consumo_risorse[i,j]*u[i] >= prezzo[j];
#dualmente, nel primale si considera la disponibilità ( $\leq$ ), qui il prezzo ( $\geq$ )
```

Modifichiamo di conseguenza il file .run precedente aggiungendo quest'ultima implementazione e visualizzando come sono fatti i singoli vincoli a livello numerico/quantitativo.

```
Reset;
model mixOpt2.mod;
data mixopt_creme.dat;
solve;
display ricavo, x;
display v_disp.dual, v_disp.body; #valore della variabile duale associata ad un vincolo
reset;
model duale.mod;
data mixOpt_creme.dat;
solve;
display costo, u;
display convenienza.body, prezzo;
```

Possiamo visualizzare il valore della variabile duale associata ad un vincolo, aggiungendo *.dual*, sul file .run precedente:

```
display v_disp.dual, v_disp; #valore della variabile duale associata ad un vincolo
```

Proviamo a variare la disponibilità di una risorsa nel duale direttamente da linea di comando, aumentando di 1 la disponibilità di zucchero:

```
ampl: let disp['zucchero'] := disp['zucchero']+1;
```

Risolvendo il problema, la funzione obiettivo aumenta di 6.4; per ogni unità di zucchero che aggiungo, guadagno in più esattamente il valore della variabile duale.

In particolare, se nel calcolo della soluzione ottima, non si ha una variazione del vettore b particolarmente elevata, le variabili duali sono ancora ottime e si ha un incremento della f.o. commisurato alle variabili presenti.

u_i è la variazione in unità della f.o. e indica di quanto aumenta il ricavo se aumentiamo di un'unità la risorsa i , cioè *quanto siamo disposti a spendere per ottenere un'unità aggiuntiva di risorsa i* : u_i viene definito *prezzo ombra* della risorsa i .

Se aumento o diminuisco il valore di una certa risorsa, a livello economico si verifica, date le precedenti considerazioni, questa per quanto piccola si riflette sulla f.o. in quantità proporzionale alla variabile duale.

Problemi di ottimizzazione su reti di flusso e algoritmi per il problema dei cammini minimi

Premessa: Algoritmo di Dijkstra (richiamato dal prof per questa parte, da me esteso per avere una totale chiarezza dei concetti). Esempio d'uso e riferimento di questa parte:

<https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/>

L'algoritmo è basato sui *grafi*, quindi strutture dati che rappresentano connessioni tra coppie di elementi, in cui ciascuno è identificato come nodo e le connessioni tra questi sono gli archi. Sono facilmente trasponibili a scenari reali, per esempio per rappresentare attività/persone e strade/percorsi che le collegano. Tra i grafi, ci sono:

- quelli indiretti (per ogni coppia di nodi connessi, i nodi vanno da un nodo ad un altro in entrambe le direzioni)
- quelli diretti (per ogni coppia di nodi connessi, è possibile andare da un nodo ad un altro in una specifica direzione)
- quelli pesati (in cui ogni arco ha un peso/costo che rappresenta una distanza/tempo)

L'algoritmo di Dijkstra permette di trovare il cammino più breve da un nodo (chiamato nodo sorgente) a tutti gli altri. In particolare:

- inizia al nodo scelto come sorgente e analizza il grafo per trovare il cammino più corto tra quel nodo e tutti gli altri nel grafo
- tiene traccia della distanza attualmente più breve da ogni nodo a quello sorgente e aggiorna questi valori quando trova un cammino più breve (inizialmente fissato ad ∞ , poi aggiornato)
- una volta trovato il cammino più corto tra nodo sorgente ed un altro nodo, il nodo è marcato come visitato e viene aggiunto al percorso
- il processo continua finché tutti i nodi del grafo non siano stati aggiunti al percorso

Problema: Non funziona in caso di costi negativi, dato che lavora solo per grafi con pesi positivi (questo perché, quando si marca un nodo visitato, si perde il conto del peso totale e si ha difficoltà a gestirlo).

Molti problemi di ottimizzazione combinatoria possono essere modellati ricorrendo ai grafi.

Consideriamo i *problemi di flusso di costo minimo*, con un esempio:

Esempio 1 (Un problema di distribuzione di energia) Una società di produzione di energia elettrica dispone di diverse centrali di produzione e distribuzione, collegate tra loro con cavi reofori. Ogni centrale i può:

- produrre p_i kW di energia elettrica ($p_i = 0$ se la centrale non produce energia);
- distribuire energia elettrica su una sottorete di utenti la cui domanda complessiva è di d_i kW ($d_i = 0$ se la centrale non serve utenti);
- smistare l'energia da e verso altre centrali.

I cavi che collegano una centrale i ad una centrale j hanno una capacità massima di u_{ij} kW e costano c_{ij} euro per ogni kW trasportato. La società vuole determinare il piano di distribuzione dell'energia elettrica di costo minimo, sotto l'ipotesi che l'energia complessivamente prodotta sia pari alla domanda totale delle sottoreti di utenti.

Lo modelliamo matematicamente tramite un grafo. Siano:

- $G = (N, A)$ un grafo i cui nodi corrispondono alle centrali e gli archi corrispondono ai collegamenti tra le centrali
- $\forall v \in N: b_v = d_v - p_v$ (richiesta del nodo v ; differenza tra la domanda che la centrale deve soddisfare e l'offerta di energia che è in grado di generare)
- nodi domanda (richiesta $b_i > 0$, la centrale deve far arrivare energia da altre centrali)
- nodi offerta (richiesta $b_i < 0$, la produzione in eccesso deve essere convogliata verso altre centrali)
- nodi transito (richiesta $b_i = 0$, se la domanda e l'offerta di energia si equivalgono)

Le variabili decisionali sono definite sugli archi del grafo relative alla quantità x_{ij} sull'arco $(i, j) \in A$.

Si usa il seguente modello (che potrebbe essere risolvibile col simplesso essendo x_{ij} continue):

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = b_v \quad \forall v \in N \\ & x_{ij} \leq u_{ij} \quad \forall (i,j) \in A \\ & x_{ij} \in \mathbb{R}_+ \end{aligned}$$

Descrivendo i pezzi di questo modello:

- La f.o. minimizza il costo complessivo e i vincoli x_{ij} definiscono il *flusso ammissibile* sulla rete.
- Il primo insieme di vincoli è il *vincolo di bilanciamento dei nodi* e assicura che la differenza tra flusso entrante ed uscente da un nodo v sia pari alla richiesta del nodo stesso.
- Il secondo insieme di vincoli rappresenta la *capacità degli archi*, quindi la quantità che può fluire in ogni arco

Il modello presentato può essere generalizzato come *problema del flusso su reti di costo minimo* (MCF – min cost flow), per modellare vari problemi di questa natura.

Disponiamo di un modello PL e rappresentiamo il modello di flusso di costo minimo come segue (grafo e vincoli, con relativa matrice E dei vincoli).

Se trascuriamo i vincoli di capacità degli archi, il modello di flusso di costo minimo presenta i seguenti vincoli, scritti per esteso:

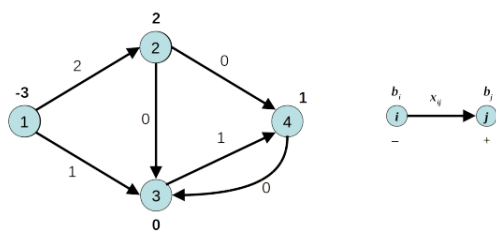


Figure 1: Esempio di flusso su rete.

$$\begin{aligned}
 -x_{12} - x_{13} &= -3 && \text{bilanciamento del nodo 1} \\
 +x_{12} - x_{23} - x_{24} &= 2 && \text{bilanciamento del nodo 2} \\
 +x_{13} + x_{23} - x_{34} + x_{43} &= 0 && \text{bilanciamento del nodo 3} \\
 +x_{24} + x_{34} - x_{43} &= 1 && \text{bilanciamento del nodo 4}
 \end{aligned}$$

x_{12}	x_{13}	x_{23}	x_{24}	x_{34}	x_{43}
-1	-1	0	0	0	0
+1	0	-1	-1	0	0
0	+1	+1	0	-1	+1
0	0	0	+1	+1	-1

Nella matrice abbiamo una colonna per ogni arco $(i, j) \in A$ e una riga per ogni nodo $v \in N$. Per ogni colonna ci sono esattamente $+1$ e -1 , dato che il flusso x_{ij} contribuisce al flusso di due soli nodi, sapendo che viene aggiunto al bilanciamento del flusso di nodo entrante e viene sottratto al bilanciamento del flusso di nodo uscente. Se la somma di tutti i b_i è pari a 0, pur non avendo la matrice rango massimo, allora si sa che si ha una *rete di flusso bilanciata*.

Consideriamo invece il caso di variabili x_{ij} *discrete* (non risolubile con il metodo del simplesso, in quanto non compatibile con i problemi di interezza), considerando ad esempio dei beni che circolano sulla rete ma non divisibili (es. container, automobili, frigoriferi, passeggeri, etc.). Come risolverlo?

$$\begin{aligned}
 \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\
 \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = b_v \quad \forall v \in N \\
 & x_{ij} \leq u_{ij} \quad \forall (i, j) \in A \\
 & x_{ij} \in \mathbb{Z}_+
 \end{aligned}$$

Riprendiamo vincoli e relativa matrice:

$$\begin{aligned}
 -x_{12} - x_{13} &= -3 && \text{bilanciamento del nodo 1} \\
 +x_{12} - x_{23} - x_{24} &= 2 && \text{bilanciamento del nodo 2} \\
 +x_{13} + x_{23} - x_{34} + x_{43} &= 0 && \text{bilanciamento del nodo 3} \\
 +x_{24} + x_{34} - x_{43} &= 1 && \text{bilanciamento del nodo 4}
 \end{aligned}$$

x_{12}	x_{13}	x_{23}	x_{24}	x_{34}	x_{43}
-1	-1	0	0	0	0
+1	0	-1	-1	0	0
0	+1	+1	0	-1	+1
0	0	0	+1	+1	-1

Caratteristiche:

- È una matrice di incidenza (matrice che descrive il grafo indicando quali collegamenti arrivano a quali vertici, che inserisce 1 dove l'arco esce, -1 dove l'arco entra, 0 altrimenti)
 - o Questo suggerisce che il flusso x_{ij} su ogni arco contribuisce ad essere aggiunto/sottratto al bilanciamento
- La somma delle righe è il vettore nullo (non è di rango massimo): $\det(E) = 0$
- Si dimostra che $\rho(E) = |N| - 1$ [sfrutta: un $+1$ e un -1 per colonna]
- Ammissibile $\Leftrightarrow \sum_{i \in N} b_i = 0$ (rete bilanciata, il sistema delle equazioni di flusso ha soluzione)

Enunciamo questa proprietà:

Proprietà 1 Sia E la matrice corrispondente ai vincoli di bilanciamento di flusso e D una qualsiasi sottomatrice quadrata (di qualsiasi dimensione). Allora $\det(D) \in \{-1, 0, +1\}$.

Matrici con tale proprietà si dicono *matrici totalmente unimodulari (TU)* (quindi, una matrice per cui ogni matrice quadrata non singolare è unimodulare, quindi determinante $\in \{-1, 0, 1\}$).

Quindi, la matrice relativa ai vincoli di bilanciamento di flusso è *TU*.

Con B sottomatrice quadrata di rango massimo di E (meno una riga). Si ha:

- $(B^{-1})_{i,j} = (-1)^{i+j} \frac{\det(B^{ji})}{\det(B)} \rightarrow$ calcolo di ogni elemento della matrice inversa tramite rimozione della j -esima riga/ i -esima colonna e quindi E è TU (per forza, dato che la base rimarrà lì dentro)
- B^{-1} è intera (componenti in $\{0, +1, -1\}$)

Sapendo che: $x_B = B^{-1}b$, $x_F = 0$, il prodotto scalare sarà intero. Di conseguenza, una soluzione di base del sistema $Ex = b$, rappresentante i vincoli di bilanciamento di flusso è ottenuta come si vede:

$$x = \begin{bmatrix} x_B \\ x_F \end{bmatrix} = \begin{bmatrix} B^{-1}\tilde{b} \\ 0 \end{bmatrix}$$

Quindi, tutte le soluzioni di base avranno coordinate intere, essendo le variabili in base ottenute dal prodotto di una matrice intera per un vettore intero, con le variabili fuori base = 0 (sapendo che per ottenere una base B bisogna sempre escludere una riga di E , pari ad un vincolo ridondante nella rete di flusso).

Dunque, vale il successivo teorema:

Teorema

Siano $A \in \mathbb{R}^{m \times n}$ una matrice TU e $b \in \mathbb{R}^m$ un vettore intero. Allora tutte le soluzioni di base del sistema $Ax = b$ ($x \geq 0$) sono intere.

Risolvendo il problema intero col simplesso (*applicabile solo quando le variabili sono reali*), troviamo una soluzione ottima di base a coordinate intere (ottenuta grazie alle proprietà di TU), ma se ne trova solo una tra quelle possibili.

Se $b_v \in \{0,1\} \forall v \in N$, posso risolvere con il metodo del simplesso.

Più in generale, sia il PLI $\min\{c^T x : Ax = b, x \in \mathbb{Z}_+^n\}$.

Se A è TU e $b \in \mathbb{Z}_+^m$, allora il problema può essere risolto con l'algoritmo del simplesso (fornisce direttamente la soluzione ottima intera). **Altrimenti, il simplesso non basta!**

Si noti che, comunque, anche aggiungendo vincoli di capacità agli archi, la TU della matrice dei vincoli si conserva; per questo motivo, possiamo in modo equivalente utilizzare lo stesso modello con variabili reali (basata sul fatto che i vincoli siano interi).

Il problema del cammino minimo, Algoritmo Label Correcting per problema cammino minimo e Generico: Correttezza e Convergenza, Albero e grafo dei cammini minimi

Uno dei problemi classici sui grafi è il *problema del cammino minimo*, organizzando un flusso sulla rete tale che un'unità corrisponda allo scambio di energia (entrante ed uscente). Cambia solamente il vettore dei termini noti. Dato un grafo, troviamo il costo minimo con queste variabili decisionali:

$$x_{ij} = \begin{cases} 1, & \text{l'arco } (i,j) \text{ è sul cammino minimo;} \\ 0, & \text{altrimenti.} \end{cases}$$

Un possibile modello è il seguente:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = \begin{cases} -1, & v = s; \\ +1, & v = d; \\ 0, & v \in N \setminus \{s, d\}. \end{cases} \\ & x_{ij} \in \{0, 1\} \quad \forall (i,j) \in A \end{aligned}$$

Ricerca operativa semplice (per davvero)

Il problema viene visto come particolare problema di flusso (quindi, come problema PL) e le variabili x_{ij} selezionano un insieme di archi a costo minimo. Essendo queste un flusso unitario, si vuole realizzare un percorso che permetta di modellare il costo minimo per evitare di pagare in f.o.; in questo modo, prendiamo il problema visto come variabili continue (grazie alla proprietà TU della matrice dei vincoli), quindi risolvibile col semplice:

$$x_{ij} \in \mathbb{R}_+ \quad \forall (i, j) \in A \quad x_{ij} \leq 1 \text{ per ottimalità (se } \nexists \text{ cicli di costo } < 0)$$

Grazie alla TU della matrice e per il fatto che la soluzione ottima non avrà mai flussi superiori ad 1, $x_{ij} \in \mathbb{Z}_+$ e si scrive quanto appena detto.

L'obiettivo è sfruttare la teoria della dualità in PL per trovare un algoritmo più efficiente del semplice (che ha complessità esponenziale); passiamo quindi al duale del problema del cammino minimo (per risolverlo in tempo polinomiale).

Riconsideriamo la matrice dei vincoli del problema originale (in cui ogni nodo ha dei vincoli π e ricordando che questa è TU). Il problema duale del cammino minimo è scritto nel riquadro sotto la matrice.

	x_{12}	x_{13}	x_{23}	x_{24}	x_{34}	x_{43}	b_i	
min	c_{12}	c_{13}	c_{23}	c_{24}	c_{34}	c_{43}		
	-1	-1	0	0	0	0	-1	$\pi_1 = \pi_s$
	+1	0	-1	-1	0	0	0	π_2
	0	+1	+1	0	-1	+1	0	π_3
	0	0	0	+1	+1	-1	+1	$\pi_4 = \pi_d$

In particolare, notiamo che i vincoli sono esattamente gli stessi e, quindi, per il teorema della dualità forte hanno la stessa soluzione partendo dalla f.o.

$$\max \quad \pi_d - \pi_s$$

$$\text{s.t.} \quad \pi_j - \pi_i \leq c_{ij} \quad \forall (i, j) \in A$$

$$\pi_v \in \mathbb{R} \quad \forall v \in N$$

Le variabili duali sono libere in quanto associate a vincoli di uguaglianza e ogni vincolo ha solo due variabili (che sono associate agli archi). In particolare, i valori π_i sono detti *etichette/labels* del nodo i (in generale, comunque, si intendono le variabili duali).

Se, in corrispondenza di etichette ammissibili, riusciamo a costruire una soluzione del problema primale che sia ammissibile primale e con ugual valore della f.o. (oppure in scarti complementari con π), allora avremmo ottenuto una soluzione ottima per il problema.

Un algoritmo molto semplice che permette di ottenere delle etichette ammissibili è questo

a destra, detto

label correcting/LC perché procede correggendo iterativamente le etichette che non soddisfano il vincolo duale:

Algoritmo *label correcting* generico

```

 $\pi_s := 0$ ; set  $p(s) = \wedge$ ;
for each  $v \in N - s$  { set  $\pi_v := +\infty$ , set  $p(v) = \wedge$ ; }
while (  $\exists (i, j) \in A : \pi_j > \pi_i + c_{ij}$  ) do {
    set  $\pi_j := \pi_i + c_{ij}$ 
    set  $p(j) := i$ ;
}

```

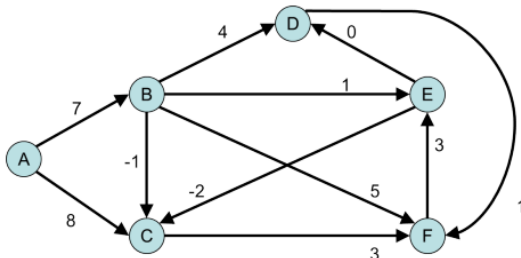
Assegniamo un valore arbitrario a π_s , che sarà uguale a 0 (cioè, eliminiamo una variabile, rendendolo ridondante) e il predecessore $p(s)$ e $p(v)$ fissati a valori nulli. Questo permette di mettere a 0 un'etichetta (magari quella del nodo sorgente), le altre a $+\infty$ e di capire il percorso dei nodi raggiunti, selezionando un insieme di archi. Fissare un $\pi_i = 0$ corrisponde ad eliminare il vincolo di bilanciamento e non farla comparire più nel problema duale.

Il ciclo si può descrivere con:

“Finché esiste un arco tale che il corrispondente vincolo duale non è soddisfatto, correggi l’etichetta facendo in modo il vincolo sia rispettato all’uguaglianza e aggiorna il nodo corrispondente”.

Consideriamo un esempio e le sue iterazioni (letteralmente, itero e trovo tutti i percorsi possibili con relativo costo e termino quando tutti gli archi soddisfano i vincoli duali):

Esempio 2 Si applichi l’algoritmo label correcting generico al grafo in Figura 2, per calcolare un insieme di etichette ammissibili duali con nodo origine $s = A$ e nodo destinazione $d = F$.



Inizializzazioni: $\pi_A = 0$; $\pi_B = \pi_C = \pi_D = \pi_E = \pi_F = +\infty$;

Iterazione 1: arco (A, B) : $\pi_B = 7$; $p(B) = A$.

Iterazione 2: arco (B, D) : $\pi_D = 11$; $p(D) = B$.

Iterazione 3: arco (A, C) : $\pi_C = 8$; $p(C) = A$.

Iterazione 4: arco (B, E) : $\pi_E = 8$; $p(E) = B$.

Iterazione 5: arco (C, F) : $\pi_F = 11$; $p(F) = C$.

Iterazione 6: arco (E, C) : $\pi_C = 6$; $p(C) = E$.

Iterazione 7: arco (C, F) : $\pi_F = 9$; $p(F) = C$.

Iterazione 8: arco (E, D) : $\pi_D = 8$; $p(D) = E$.

Costo:

π_F : $7 + 1 - 2 + 3 = 9$.

Cammino:

$F \leftarrow C \leftarrow E \leftarrow B \leftarrow A$

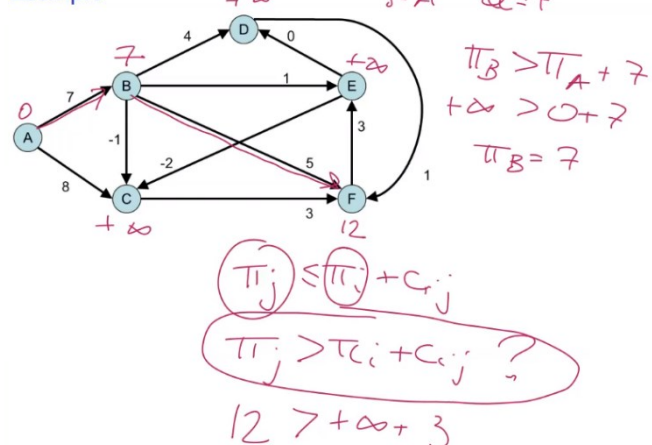
Come si ragiona:

Per ogni iterazione, considero di raggiungere un qualsiasi arco con costo minimo che mi permette di raggiungerlo; di fatto a questo algoritmo basta esaminare tutte le etichette e “sperare” di raggiungerle tutte.

Ciò viene evidenziato a lato.

Il cammino minimo si trova considerando tutti i puntatori dall’ultimo al primo nodo.

Esempio



Vogliamo abbassare il costo delle etichette ad ogni passaggio (queste rappresentano, infatti, i cammini minimi).

La soluzione ammissibile duale è data dall’algoritmo label correcting; ora verifico che siano ottime.

Per farlo, sfrutto le proprietà del problema duale.

La soluzione primale si ottiene sulla base di $x_{ij} = 1$ sul cammino $[A, B, C, E, F]$ se $p(j) = 1, 0$ altrimenti).

Così facendo troviamo un cammino verso A sfruttando i puntatori di tutti i predecessori partendo dal nodo destinazione d . La soluzione duale ha costo 9; per dualità forte, le soluzioni sono ottime, dato che problema primale/duale hanno la stessa soluzione.

$$\sum_{(i,j) \in A} c_{ij} x_{ij} = \pi_F - \pi_A \Rightarrow x \text{ ottima!}$$

Dimostriamo la correttezza dell'algoritmo (usando le CCPD), cioè che trova effettivamente il cammino minimo. Prima diamo questo lemma:

Lemma 1 Alla fine di ogni iterazione, considerati i valori correnti delle etichette π , per ogni nodo $j \in N : \pi_j < +\infty$ si ha:

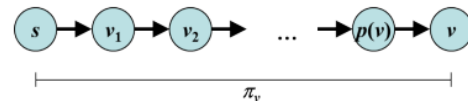
- esiste un cammino P dal nodo origine s a j ;
- $p(j)$ è il predecessore di j in tale cammino P ;
- il costo del cammino P è pari a π_j .

Inoltre, se l'algoritmo termina, per ogni nodo $v \neq s$ per cui esiste un cammino da s a v , si ha $\pi_v < +\infty$ e $p(v) \neq \wedge$.

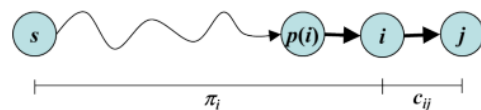
Descrivendo la dimostrazione a parole:

All'inizio delle iterazioni, esistono al più due nodi compreso il nodo origine, con predecessore 0 e costo nullo. Alla fine della prima iterazione, abbiamo un successore di s , avendo due etichette π_0 e π_w suo successore. Tra queste, esiste un cammino, in questo momento con un costo 0 e con il giusto predecessore.

Nel caso induttivo ($k > 1$), se $\pi_v < +\infty$, esiste un cammino tra tutti i nodi con un predecessore per ognuno e un costo π_v :



All'iterazione $k + 1$, e con arco (i, j) , esiste certamente un arco $\pi_j = \pi_i + c_{ij}$, con un costo determinato dal predecessore π_i con un costo π_j . Questo afferma che esista un cammino minimo partendo dal predecessore.



Ciò conclude la dimostrazione; dimostriamo l'opposto.

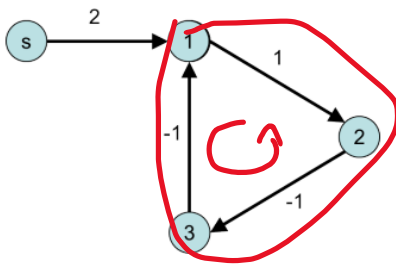
Assumendo che l'algoritmo termini, esiste un nodo v con un cammino P da s verso v (serve per capire se, partendo da un nodo, riesca ad individuare un cammino generico, aggiornando l'etichetta con un valore finito).

Per assurdo, $\pi_v = +\infty$; al termine dell'algoritmo, π ammissibile e, per ogni arco, sono rispettati i vincoli duali e, quindi, esiste un cammino del tipo:

$$\begin{aligned} +\infty = \pi_v &\leq \pi_{v_{n-1}} + c_{v_{n-1}v} \leq \pi_{v_{n-2}} + c_{v_{n-2}v_{n-1}} + c_{v_{n-1}v} \leq \dots \\ &\dots \leq \pi_{v_1} + c_{v_1v_2} + \dots + c_{v_{n-2}v_{n-1}} + c_{v_{n-1}v} \leq \dots \\ &\dots \leq \pi_s + c_{sv_1} + c_{v_1v_2} + \dots + c_{v_{n-2}v_{n-1}} + c_{v_{n-1}v} \end{aligned}$$

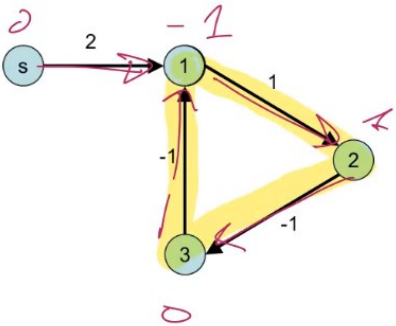
Quindi, otteniamo una somma finita e, sicuramente, avrò un'etichetta finita, ricostruita tramite i predecessori e, quindi, l'algoritmo termina e funziona (dato che $\pi_s = 0$)

Problema: L'algoritmo label-correcting generico, in alcuni casi, *non termina* (anche in caso di *ciclo di lunghezza negativa*). Nell'esempio che segue, si ha proprio un ciclo (evidenziato). Quindi, in presenza di cicli di costo negativo, l'algoritmo non converge.



Inizializzazioni: $\pi_s = 0$; $\pi_1 = \pi_2 = \pi_3 = +\infty$;
 Iterazione 1: arco $(s, 1)$: $\pi_1 = 2$; $p(1) = s$.
 Iterazione 2: arco $(1, 2)$: $\pi_2 = 3$; $p(2) = 1$.
 Iterazione 3: arco $(2, 3)$: $\pi_3 = 2$; $p(3) = 2$.

Iterazione 4: arco $(3, 1)$: $\pi_1 = 1$; $p(1) = 3$.
 Iterazione 5: arco $(1, 2)$: $\pi_2 = 2$; $p(2) = 1$.
 Iterazione 6: arco $(2, 3)$: $\pi_3 = 1$; $p(3) = 2$.
 Iterazione 7: arco $(3, 1)$: $\pi_1 = 0$; $p(1) = 3$.
 Iterazione 8: arco $(1, 2)$: $\pi_2 = 1$; $p(2) = 1$.
 Iterazione 9: arco $(2, 3)$: $\pi_3 = 0$; $p(3) = 2$.
 Iterazione 10: arco $(3, 1)$: $\pi_1 = -1$; $p(1) = 3$.



In pratica, arriva un punto in cui si crea un ciclo dato che il predecessore/successore è costantemente migliore e non esce mai convergendo (banalmente, il cammino minimo avrà un nodo che compare almeno due volte). L'esempio del ciclo è quello a lato.

Uso l'algoritmo LC generico solo quando sono sicuro che non esistono cicli di lunghezza negativa (in tali casi, l'algoritmo converge in un numero finito di iterazioni).

Consideriamo il caso in cui non esistono cicli di costo negativo.

- Ad ogni iterazione, un'etichetta diminuisce strettamente (almeno una, finché l'algoritmo itera)
- Le etichette finite indicano il costo di un cammino ammissibile, non posso diminuire indefinitamente: $\pi_j \geq c(P^*) > -\infty$
- π_j rimane sempre a $+\infty$ se j non è raggiungibile

Quindi, senza cicli negativi (e assumendo c_{ij} razionali; se fossero numeri reali, la convergenza non sarebbe garantita, in quanto continuando a togliere ci avviciniamo al valore senza mai raggiungerlo), si avrà convergenza in un numero finito di iterazioni (prima o poi finisce).

Osservazione $\rightarrow$ Senza cicli negativi, un cammino minimo contiene al più $|N| - 1$ archi (cioè, avrà sempre un cammino un albero aciclico ma non un ciclo [dato che tracciamo il predecessore]).

La complessità computazionale senza cicli negativi ha:

- come numero max di iterazioni $N - 1$ etichette da $+\infty$ a costo minimo
- $+\infty$: possiamo usare un upper bound $\bar{M}$ per il costo massimo
- sia $\underline{M}$ un lower bound per il costo minimo
- ogni iterazione costa $O(|A|)$, dove A è il numero di archi

Senza cicli negativi, l'algoritmo LCG (Label Correcting Generico) converge in $O((|N| - 1)(\bar{M} - \underline{M}|A|))$ (numero di nodi moltiplicato al numero di diminuzioni che posso fare).

È quindi possibile stabilire un limite superiore $\bar{M}$, un limite inferiore $\underline{M}$ per il costo del cammino minimo da dal nodo sorgente s ad un qualsiasi nodo j :

- max costo P ammissibile $\bar{M} = (|N| - 1) \max \left\{ 0, \max_{(i,j) \in A} c_{ij} \right\}$
- min costo P ammissibile $\underline{M} = (|N| - 1) \min \left\{ 0, \min_{(i,j) \in A} c_{ij} \right\}$

L'algoritmo ha complessità *pseudopolinomiale*, cioè un algoritmo la cui complessità temporale nel caso peggiore è polinomiale rispetto al valore numerico dell'input (non al numero di input).

Infatti, la complessità dipende da $\bar{M} - \underline{M}$, a sua volta dipendente dai costi sugli archi, che sono parametri (e non dimensione) del problema. Tale differenza potrebbe essere molto elevata e, di conseguenza, la convergenza potrebbe essere molto lenta.

Riassumendo → ritrovo una soluzione ammissibile sapendo che esiste sempre un cammino tra sorgente e destinazione, ricostruibile con la catena dei predecessori e costruisco una soluzione primale ammissibile (bilanciata) ottima (per la dualità forte, applicabile in quanto usato per PL con variabili continue). Queste osservazioni completano la discussione di correttezza dell'algoritmo, individuando quindi un cammino minimo da un nodo origine s ad un nodo destinazione d .

Correttezza

Proprietà

Senza cicli negativi, l'algoritmo LCG trova il cammino minimo da s a d

- π è soluzione ammissibile duale di costo π_d
- Senza cicli negativi, se esiste cammino da s a d , allora $\pi_d < +\infty$.
- Per il Lemma, esiste $P = [s, \dots, d]$ di costo π_d . Tale cammino si ottiene con la catena dei predecessori.
- Costruisco una soluzione primale ponendo $x_{ij} = 1$ sugli archi di P
- La soluzione è ammissibile primale (flusso bilanciato) e ha costo

$$\sum_{(i,j) \in A} c_{ij} x_{ij} = \sum_{(i,j) \in P} c_{ij} = \pi_d$$

⇒ P è ottimo! (dualità forte¹)

Consideriamo invece il caso in cui, partendo dalla stessa origine, cambia la destinazione (in generale, cambia la funzione obiettivo, ma non cambiano i vincoli; quindi, partendo da un qualsiasi nodo, non cambia il raggiungimento tramite cammino).

Esponiamo quindi la coppia di modelli primale-duale relativa all'albero dei cammini minimi (dove sta max è il problema duale, quello sopra come prima è il classico cammino minimo):

Stessa origine, destinazione diversa

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = \begin{cases} -1, & v = s; \\ +1, & v = d; \\ 0, & v \in N \setminus \{s, d\}. \end{cases} \\ & x_{ij} \in \mathbb{R}_+ \quad \forall (i,j) \in A \\ \max \quad & \pi_d - \pi_s \\ \text{s.t.} \quad & \pi_j - \pi_i \leq c_{ij} \quad \forall (i,j) \in A \\ & \pi_v \in \mathbb{R} \quad \forall v \in N \end{aligned}$$

In particolare, i vincoli dei due problemi duali sono esattamente gli stessi e, di conseguenza, lo stesso algoritmo label correcting fornisce delle etichette ammissibili duali, per tutte le possibili destinazioni del cammino minimo.

In generale, qualsiasi sia il nodo, si costruisce un cammino ottimo grazie al teorema della dualità forte, applicabile a qualsiasi cammino dato che è ricostruibile con i predecessori e con costo uguale a π . Quanto a lato formalizza e completa quanto appena detto.

- π sono ammissibili **per tutte le destinazioni**
- $p(\cdot)$ fornisce cammini da s verso tutti i nodi
- per tutti tali cammini vale la dualità forte

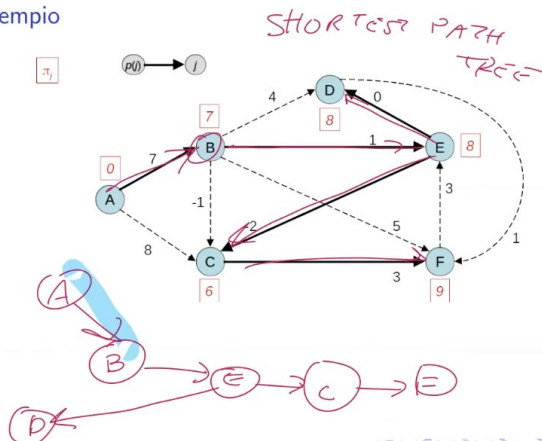
Proprietà

Algoritmo LCG restituisce i cammini minimi **da s verso tutti i nodi**

- $p(\cdot)$ determina struttura gerarchica:
albero dei cammini minimi con radice in s .

Per ogni nodo, come si vede da immagine, si ha un solo genitore e si costruisce, di fatto, una struttura gerarchica (albero orientato). Quanto segue è l'*albero dei cammini minimi* del problema dato, che rappresenta *uno* dei possibili cammini di costo minimo. Nella figura che segue, sono note in rosso anche le etichette ottime. L'albero rappresenta *una sola* delle possibili soluzioni ottime.

Esempio

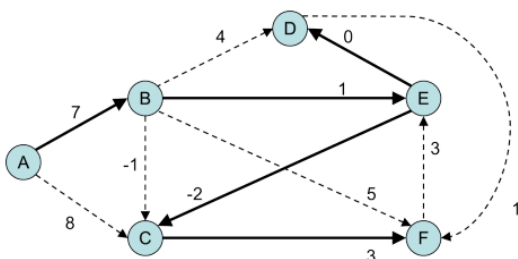


Risolvendo il problema del cammino minimo attraverso la teoria della dualità, è possibile ricavare *tutti* i cammini minimi da un nodo sorgente verso un altro nodo (*grafo dei cammini minimi/shortest path tree*).

Dato un grafo $G = (N, A)$ e le etichette ottime π derivate dall'algoritmo label correcting in relazione ad un nodo origine s , si costruisca il grafo $G_{s,\pi} = (N, A_{s,\pi})$ dove $A_{s,\pi} = \{(i, j) \in A : \pi_j = \pi_i + c_{ij}\}$.

Se io costruisco un cammino minimo usando solo archi del grafo dei cammini minimi, sto costruendo una soluzione ammissibile primale che è in scarti complementari con una soluzione ammissibile duale, quindi ottima. In questo modo, siamo certi di avere un cammino minimo. In particolare, sappiamo essere una soluzione ammissibile, valenti le condizioni CCPD.

Nella figura che segue, il grafo dei cammini minimi è dato dai soli archi non tratteggiati:



$$G_{s,\pi} = (N, A_{s,\pi}) \quad \text{dove} \quad A_{s,\pi} = \{(i, j) \in A : \pi_j = \pi_i + c_{ij}\}.$$

Cammino $[s, \dots, j]$ su $G_{s,\pi} \iff$ cammino minimo da s a j

cammino individua soluzioni ammissibili primale e duale con

$$(\pi_j - \pi_i - c_{ij}) \times x_{ij} = 0, \quad \forall (i, j) \in A$$

Algoritmo di Bellman-Ford: Convergenza, Correttezza, Complessità. Esempi applicativi e Cammini Minimi con massimo numero di archi/hop

L'algoritmo label correcting generico presenta una complessità pseudo-polinomiale. Ciò è dovuto al fatto che non possono essere fatte assunzioni sull'ordine nel quale gli archi vengono controllati e le relative etichette aggiornate. In effetti, dando sistematicità ai controlli e agli aggiornamenti è possibile tenere sotto controllo il numero di iterazioni; tuttavia, non abbiamo garanzia che converga sempre (solo in assenza di cicli negativi, non avendo ordine definito).

Un algoritmo che non va avanti ad iterare per sempre e serve a dare un ordine è l'algoritmo di Bellman-Ford, presentato come segue (complessità $O(N * A)$):

```

 $\pi_s := 0$ ; set  $p(s) = \wedge$ ;
for each  $v \in N - s$  { set  $\pi_v := +\infty$ , set  $p(v) := \wedge$  }
for  $h = 1$  to  $|N|$  {
  set  $\pi' := \pi$ ; set flag_aggiornato := false;
  for all  $(i, j) \in A : \pi_j > \pi'_i + c_{ij}$  do {
    set  $\pi_j := \pi'_i + c_{ij}$ ;
    set  $p(j) := i$ ;
    set flag_aggiornato := true;
  }
  if (not flag_aggiornato) then { STOP:  $\pi$  è ottima }
}
STOP:  $\exists$  ciclo di costo negativo.

```

Viene fatto un numero di iterazioni pari al numero N di nodi ed effettua un controllo sugli archi A /aggiornamento (salvando il precedente); il controllo viene fatto sistematicamente su tutti gli archi.

Ad ogni iterazione si controllano le etichette duali su tutti gli archi. Se usciamo dall'algoritmo, abbiamo trovato le etichette stabili oppure avendo aggiornato almeno un'etichetta (esiste almeno un ciclo negativo).

Come trovare un ciclo negativo

Le condizioni per trovare un ciclo con Bellman-Ford:

- arrivo alla fine delle iterazioni e vedo che le etichette non sono stabili (cioè, è stato fatto almeno un aggiornamento);
- ripercorro la catena dei predecessori e mi accorgo che ritorno su un nodo che avevo già attraversato

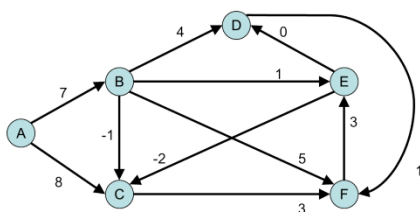
Domanda spesso posta nel gruppo

- Occorre mettere la spiegazione di tutti i passaggi per gli algoritmi dei grafi (come si vede negli appelli di esempio)?
 - Sì; si parla apposta di "giustificare" i passi dell'algoritmo in una tabella. Anche perché, logicamente, si fa capire che quello che si sta facendo non piove dal cielo.

Rispetto all'algoritmo label correcting generico, l'algoritmo di Bellman-Ford:

- presenta una nuova condizione di arresto (che permette all'algoritmo di individuare dei cicli negativi)
- il numero di iterazioni è limitato dal ciclo for esterno. Come vedremo, nonostante tale utile limitazione, se si esce con *not flag_aggiornato*, le etichette finali sono ottime.

Esempio 6 Si consideri il grafo in Figura 9 e si calcoli il costo dei cammini minimi da A verso tutti gli altri nodi applicando l'algoritmo di Bellman-Ford.



Organizziamo le iterazioni in forma tabellare, per seguire il corretto ordine degli archi e aggiornamento delle etichette. Qui riportiamo il valore di π e del nodo predecessore.

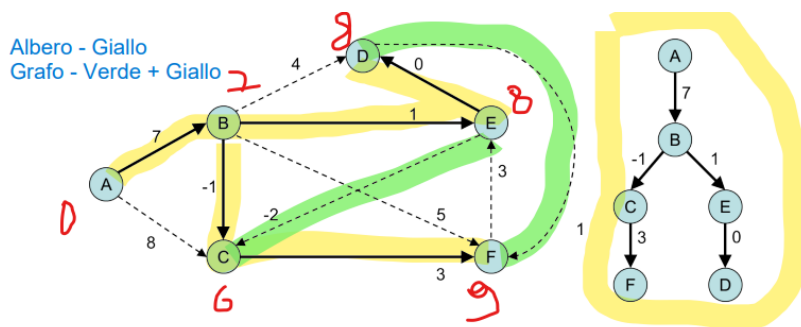
iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	flag
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	true
$h = 1$	0 ($\wedge$)	+7 (A)	+8 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	true
$h = 2$	0 ($\wedge$)	+7 (A)	+6 (B)	+11 (B)	+8 (B)	+12 (B) / +11 (C)	true
$h = 3$	0 ($\wedge$)	+7 (A)	+6 (B)	+8 (E)	+8 (B)	+9 (C)	true
$h = 4$	0 ($\wedge$)	+7 (A)	+6 (B)	+8 (E)	+8 (B)	+9 (C)	false

(Attenzione a controllare sempre la riga dell'iterazione precedente)

Per esempio:

- Iterazione (1)	$h=1$	0 ($\wedge$)	+7 (A)	+8 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	true
- Iterazione (2)	$h=1$	0 ($\wedge$)	+7 (A)	+8 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	true
	$h=2$	0 ($\wedge$)	+7 (A)	+6 (B)	+11 (B)	+8 (B)	+12 (B) / +11 (C)	true

Le etichette ammissibili duali con le quali ho finito mi permettono di ottenere gli alberi contenuti nel grafo dei cammini minimi. Visualizziamo alberi e grafo (quest'ultimo ottenibile con il rispetto dei vincoli duali):



L'albero è dato seguendo a cascata tutti i predecessori (quindi, dall'ultima iterazione guardo tutti i nodi precedenti e collego).

Il grafo invece è dato dai vincoli di uguaglianza ($x_{ij} = 1$); quindi è dato sempre *almeno dagli stessi nodi dell'albero ed eventualmente altri che hanno stesso costo minimo (più percorsi che mi portano ad avere sempre lo stesso costo minimo finale)*

Per trovare gli altri archi del grafo dei cammini minimi, ci segniamo le etichette ottime (quelle in rosso). Partendo da questo, considero per ogni etichetta ottima tutti i costi che sono $\leq$ a quelli che ho già (dunque, non aggiungono costo rispetto a quelle ottime). Vengono quindi aggiunti due collegamenti, quelli che partono da D e da E. *Considero semplicemente etichette che hanno costo $\leq$ rispetto alle etichette ottime.*

In generale, il grafo dei cammini minimi viene dato da tutti e soli i cammini minimi; per questo, spesso e volentieri, può coincidere con l'albero dei cammini minimi.

Inoltre, in presenza di un grafo con numero massimo di archi/hop, questo *non rappresenta* tutti e soli i cammini minimi vincolati (nel senso che, semplicemente, avremo altri cammini minimi aumentando il numero di archi e il grafo li rappresenta tutti; con un limite massimo *non è possibile*).

Per dimostrare convergenza, complessità e correttezza dell'algoritmo ci si basa sulla seguente proprietà:

Proprietà: cammini minimi e max-hop

Lemma

Alla fine dell'iterazione con $h = \bar{h}$, π_j rappresenta il costo di un **cammino minimo da s a j con al più $\bar{h}$ archi**

Per induzione:

- $h = 1$: verificati **tutti** i modi di arrivare da s con 1 solo arco, ✓
- $h = k$
 - π' ottime per $k - 1$ archi (ipotesi induttiva)
 - verificati **tutti** i modi di raggiungere nodi con *un arco in più*, ✓

Quanto enuncia concretamente questa proprietà a lato è che, in un certo momento, consideriamo il cammino minimo con n archi; ad esempio, se consideriamo un cammino minimo $\pi_F = 11$ (nodi A, C, F), avremo che questo è minimo quando si hanno esattamente 2 archi.

Quindi, magari con 3 archi ce ne sta un altro, con 1 un altro ancora, etc.

In questo modo, ho aggiornato tutte le volte in cui ho davvero *un arco un più*, avendo quindi il miglior cammino con k (ottimo nel caso in cui abbia applicato “bene” l’algoritmo fino ad ora).

Prendiamo un grafo in cui abbiamo due casi (capendo perché funziona)

- 1) Senza cicli negativi, cammino minimo ha al più $|N| - 1$ archi $\Rightarrow$ *flag_aggiornato* rimane *false* entro iterazione $h = |N|$ (vuol dire quindi che non si ha un ciclo e uscirei certamente prima di N)
- 2) Con cicli negativi *flag_aggiornato* diventa *true* all’iterazione $h = |N|$ (un ciclo negativo individuato seguendo a ritroso i puntatori a partire da uno dei nodi con etichetta aggiornata); di fatto, di certo esisterà un ciclo e quindi si ferma ad N diventando vero.

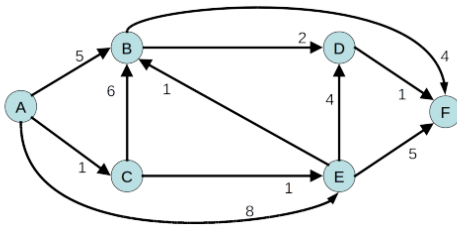
Inoltre, si avrà complessità polinomiale, con aggiornamenti dei nodi in tempo costante e numero di controlli pari al numero di archi;

Proprietà 7 Dato un grafo pesato $G = (N, A)$ e un nodo origine $s \in N$, l’algoritmo di Bellman-Ford fornisce la soluzione ottima del problema del cammino minimo dal nodo origine s verso tutti gli altri nodi o individua un ciclo di costo negativo in $O(|N| \cdot |A|)$.

Quando descritto sopra si può tradurre con:

“A prescindere dal grafo, comunque trova un cammino minimo e, a prescindere, trova sempre un ciclo negativo quando questo sia presente”.

Esempio 7 Si consideri il grafo in Figura 11 e si calcoli il costo dei cammini minimi da A verso tutti gli altri nodi applicando l’algoritmo di Bellman-Ford.



iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)
$h = 1$	0 ($\wedge$)	5 (A)	1 (A)	$+\infty$ ($\wedge$)	8 (A)	$+\infty$ ($\wedge$)
$h = 2$	0 ($\wedge$)	5 (A)	1 (A)	7 (B)	2 (C)	9 (B)
$h = 3$	0 ($\wedge$)	3 (E)	1 (A)	6 (E)	2 (C)	8 (D) 7 (E)
$h = 4$	0 ($\wedge$)	3 (E)	1 (A)	5 (B)	2 (C)	7 (E)
$h = 5$	0 ($\wedge$)	3 (E)	1 (A)	5 (B)	2 (C)	6 (D)
$h = 6$	0 ($\wedge$)	3 (E)	1 (A)	5 (B)	2 (C)	6 (D)

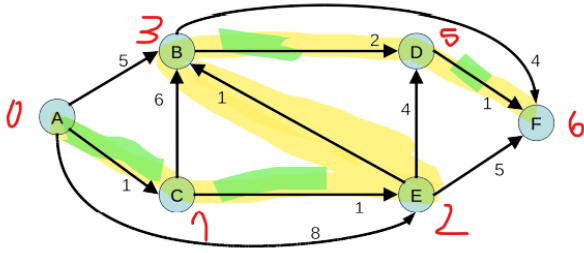
Attenzione che ha senso controllare i nodi di tutte le iterazioni (cioè, tutto in riga) quando effettivamente si ha un aggiornamento di almeno un nodo (conviene segnarselo in una colonna a lato); altrimenti, se non si hanno cambiamenti, implicitamente significa che la situazione non migliora e posso non controllare, scrivendo già su tutta la colonna il costo trovato.

Un esempio di nodi aggiornati che vado a segnare (mi basta quindi controllare i nodi uscenti rispetto ai nodi che si sono aggiornati):

iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	agg.
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	
$h = 1$	0 ($\wedge$)	5 (A)	1 (A)	$+\infty$ ($\wedge$)	8 (A)	$+\infty$ ($\wedge$)	
$h = 2$	0 ($\wedge$)	5 (A)	1 (A)	7 (B)	2 (C)	9 (B)	
$h = 3$	0 ($\wedge$)	3 (E)	1 (A)	6 (E)	2 (C)	8 (D) 7 (E)	B D E
$h = 4$	0 ($\wedge$)	3 (E)	1 (A)	5 (B)	2 (C)	7 (E)	D

Si noti che, essendoci un cammino minimo composto dal massimo numero di archi (5, cammino da A a F), solo all’iterazione $h = 6$ si ha la convergenza delle etichette.

Segniamo l’albero dei cammini minimi (giallo) e grafo dei cammini minimi (verde); nel qual caso, come si vede, coincide.



Si fa notare come, ad ogni iterazione, sia inutile verificare le condizioni per gli archi uscenti da nodi le cui etichette non vengono aggiornate.

Pertanto, è possibile migliorare l'efficienza computazionale media andando a memorizzare, ad ogni iterazione, la lista dei nodi le cui etichette vengono aggiornate (lista *Aggiornati*). All'iterazione successiva basterà controllare gli archi che escono da nodi aggiornati. L'algoritmo è identico al precedente, con la sola aggiunta della lista *Aggiornati*.

Algoritmo di *Bellman-Ford* con lista *Aggiornati*

```

 $\pi_s := 0$ ; set  $p(s) = \wedge$ ; Aggiornati := {s}
for each  $v \in N - s$  { set  $\pi_v := +\infty$ , set  $p(v) := \wedge$  }
for  $h = 1$  to  $|N|$  {
  set  $\pi' := \pi$ ;
  set Aggiornati' := Aggiornati;
  set Aggiornati :=  $\emptyset$ ;
  for all  $(i, j) \in A : i \in \text{Aggiornati}' \wedge \pi_j > \pi'_i + c_{ij}$  do {
    set  $\pi_j := \pi'_i + c_{ij}$ 
    set  $p(j) := i$ ;
    set Aggiornati := Aggiornati  $\cup \{j\}$ ;
  }
  if ( Aggiornati =  $\emptyset$  ) then { STOP:  $\pi$  è ottima }
}
STOP:  $\exists$  ciclo di costo negativo.
    
```

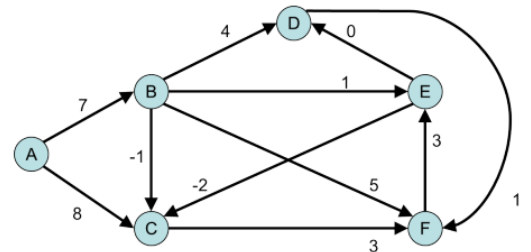
Cammini minimi con al più $\bar{h}$ archi (situazioni utili, nel concreto, es. nelle reti di telecomunicazione)

- Fermarsi all'iterazione $h = \bar{h}$ (esempi, definiti anche con cicli negativi)
- Cammini individuati saltando alla riga precedente
- Alberi e grafi dei cammini minimi non sono definiti (etichette non necessariamente ammissibili)

Esempio 9 Si calcolino i cammini minimi con al più 3 archi per il grafo in Figura 2, a partire dal nodo A.

Applicando l'algoritmo di Bellman-Ford limitato a 3 iterazioni si ottiene:

iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	Aggiornati
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A
$h = 1$	0 ($\wedge$)	+7 (A)	+8 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B, C
$h = 2$	0 ($\wedge$)	+7 (A)	+6 (B)	+11 (B)	+8 (B)	+12 (B) +11 (C)	C, D, E, F
$h = 3$	0 ($\wedge$)	+7 (A)	+6 (B)	+8 (E)	+8 (B)	+9 (C)	D, F

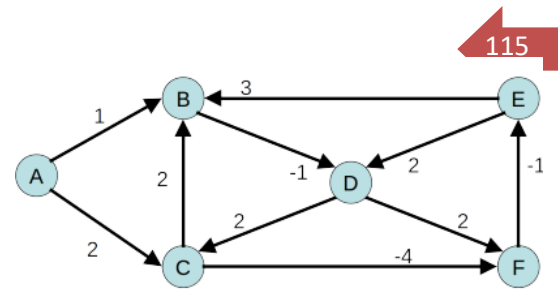


Si noti che, anche in presenza di cicli negativi, i cammini minimi con un massimo numero di archi sono ben definiti, visto che non è possibile percorrere i cicli negativi un numero di volte illimitato.

Esempio 10 Si calcolino i cammini minimi con al più 4 archi a partire dal nodo A per il grafo in Figura 13.

Applicando l'algoritmo di Bellman-Ford limitato a 4 iterazioni si ottiene:

iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	Aggiornati
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A
$h = 1$	0 ($\wedge$)	+1 (A)	+2 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B, C
$h = 2$	0 ($\wedge$)	+1 (A)	+2 (A)	0 (B)	$+\infty$ ($\wedge$)	-2 (C)	D, F
$h = 3$	0 ($\wedge$)	+1 (A)	+2 (A)	0 (B)	-3 (F)	-2 (C)	E
$h = 4$	0 ($\wedge$)	0 (E)	+2 (A)	-1 (E)	-3 (F)	-2 (C)	B, D



Esempio 11 Si risolva il problema dei cammini minimi con origine nel nodo A e al più 3 archi per il grafo in Figura 15.

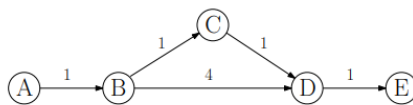
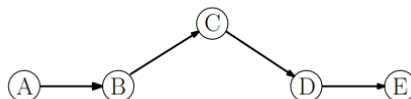


Figure 15: Grafo per l'Esempio 11.

Applicando l'algoritmo di Bellman-Ford limitato a 3 iterazioni si ottiene:

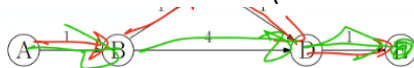
iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	Aggiornati
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A
$h = 1$	0 ($\wedge$)	1 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B
$h = 2$	0 ($\wedge$)	1 (A)	2 (B)	5 (B)	$+\infty$ ($\wedge$)	C, D
$h = 3$	0 ($\wedge$)	1 (A)	2 (B)	3 (C)	6 (D)	D, E

Utilizzando i puntatori ottenuti nell'ultima iterazione, si ottiene l'albero in Figura 16, che *non* è un albero di cammini minimi con al massimo 3 archi, visto che il cammino da A a E ne contiene 4.



Nell'esempio a lato, ci sarebbe un cammino minimo, ma non è valido in quanto considera 4 archi e non 3; in generale, la logica è che, guardando i predecessori, siamo sempre in grado di capire il cammino minimo verso qualsiasi destinazione

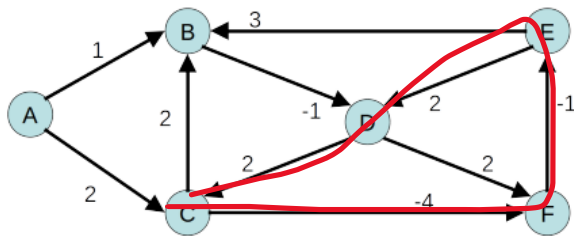
Un cammino minimo è A/B/C/D/E, che però ha 4 archi. Queste sono etichette ammissibili duali, ma non per questo problema. Il cammino minimo considera, come detto, a lato, di partire dall'ultima iterazione, trovare il predecessore e diminuire di 1 (da 3 diminuisco a 2, da 2 diminuisco ad 1).



iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	Aggiornati
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A
$h = 1$	0 ($\wedge$)	1 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B
$h = 2$	0 ($\wedge$)	1 (A)	2 (B)	5 (B)	$+\infty$ ($\wedge$)	C, D
$h = 3$	0 ($\wedge$)	1 (A)	2 (B)	3 (C)	6 (D)	D, E

In generale, nel caso si risolva un problema di cammino minimo con un massimo numero di hop, non si può parlare di albero dei cammini minimi. Per lo stesso motivo, non si può neanche definire un grafo dei cammini minimi in presenza di un massimo numero di hop. Infatti, il grafo dei cammini minimi, anche se costruito sulla base delle etichette ottenute limitando il numero di iterazioni, e anche se queste etichette implicano dei cammini con un numero corretto di archi, NON rappresenta tutti e soli i cammini minimi vincolati: alcuni cammini sul grafo potrebbero superare il limite di archi.

Vediamo un altro esempio:



iterazione	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	Aggiornati
$h = 0$	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A
$h = 1$	0 ($\wedge$)	+1 (A)	+2 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B, C
$h = 2$	0 ($\wedge$)	+1 (A)	+2 (A)	0 (B)	$+\infty$ ($\wedge$)	-2 (C)	D, F
$h = 3$	0 ($\wedge$)	+1 (A)	+2 (A)	0 (B)	-3 (F)	-2 (C)	E
$h = 4$	0 ($\wedge$)	0 (E)	+2 (A)	-1 (E)	-3 (F)	-2 (C)	B, D
$h = 5$	0 ($\wedge$)	0 (E)	+1 (D)	-1 (E)	-3 (F)	-2 (C)	C
$h = 6$	0 ($\wedge$)	0 (E)	+1 (D)	-1 (E)	-3 (F)	-3 (C)	F

L'algoritmo ha terminato con $flag_aggiornato = true$ (perché ad ogni iterazione, la colonna Aggiornati riporta almeno un nodo modificato), pertanto esiste un ciclo negativo.

Tale ciclo è individuato partendo dal nodo F (uno dei nodi) che ha subito un aggiornamento all'ultima iterazione, e procedendo a ritroso attraverso i predecessori; come si vede, un nodo è ripetuto.

$F \leftarrow C \leftarrow D \leftarrow E \leftarrow F$ (di costo $-1 + 2 + 2 - 4 = -1$).

Il ciclo negativo si evidenzia sopra in rosso.

La correttezza degli algoritmi label correcting (tra cui l'algoritmo di Bellman-Ford) si basa sul seguente principio di ottimalità, tramite un algoritmo di programmazione dinamica (se un cammino è minimo, lo sono tutti i sottocammini, poi ricomponibili tramite un algoritmo bottom-up):

Proprietà 8 (Condizioni di ottimalità dei cammini minimi) Dato un grafo $G = (N, A)$ con costi c sugli archi e un nodo origine $s \in N$, e date delle etichette π_v per ogni nodo $v \in N$, queste rappresentano i costi dei cammini minimi da s verso v se e solo se π_v è la lunghezza di un cammino ammissibile da s a v e $\pi_j - \pi_i \leq c_{ij}$, $\forall (i, j) \in A$.

Grazie a questo, sappiamo che i cammini minimi si mantengono ottime nelle sottoistanze del problema, secondo il principio noto in programmazione dinamica come *principio di ottimalità*.

Nel qual caso (citato a puro livello di completezza e chiarezza logica), sappiamo che nella pratica se non abbiamo un cammino di costo minimo, ne esisterà sicuramente uno di costo inferiore:

Proprietà 9 (Principio di sub-ottimalità dei cammini minimi) P è un cammino minimo da s a d se e solo se, per ogni coppia di nodi i e j nel cammino, il sotto-cammino da i a j $P_{ij} \subseteq P$ è un cammino minimo.

Tutorato (1) del prof: Esercizi di programmazione lineare

2. Un'associazione umanitaria internazionale deve spedire i regali di Natale per i bambini di due orfanotrofi in Africa. Quest'anno si regaleranno puzzle, orsacchiotti e trenini, secondo le richieste minime sintetizzate in tabella:

Orfanotrofo	Puzzle	Orsacchiotti	Trenini
A: Tanzania	2500	3000	1400
B: Kenia	2100	2400	1300

I regali saranno smistati a partire da 3 centri di raccolta. I regali sono stati confezionati in pacchi per la spedizione, e ciascun centro di raccolta ha composto dei pacchi diversi. La composizione dei pacchi e il numero di pacchi disponibili sono indicati nella seguente tabella:

Centro	Puzzle per pacco	Orsacchiotti per pacco	Trenini per pacco	Pacchi disponibili
1	10	4	15	220
2	5	12	7	240
3	14	9	16	260

La spedizione avverrà per via aerea: da ciascun centro potrà partire al massimo un aereo per ciascuna destinazione, tenendo conto che il Centro 2 ha al massimo un aereo a disposizione. Ciascun aereo ha un costo fisso (tasse aeroportuali in partenza) e un costo variabile per pacco, secondo i dati (in euro) riportati in tabella

Centro	Costo fisso per aereo	Costo variabile per pacco verso:	
		Tanzania	Kenia
1	500	10	12
2	300	15	14
3	400	5	25

Si vuole determinare un piano di smistamento dei regali di costo minimo, considerando che il governo della Tanzania incentiva l'arrivo di puzzle chiedendo una sovrattassa di 1000 € qualora il numero di puzzle arrivati non superi di 500 unità la richiesta minima.

x_{ij} : pacchi da inviare da $i \in \{1,2,3\}$ dal paese $j \in \{T, K\}$

y_{ij} : variabile binaria che vale 1 se aereo parte da i a j

$$\min 10x_{1T} + 12x_{1K} + 15x_{2T} + 14x_{2K} + 5x_{3T} + 25x_{3K} \\ \text{s. t.}$$

$$10x_{1T} + 5x_{2T} + 14x_{3T} \geq 2500 \text{ // (richieste minime puzzle Tanzania)}$$

$$4x_{1T} + 12x_{2T} + 2x_{3T} \geq 3000 \text{ // (richieste minime orsacchiotti Tanzania)}$$

$$15x_{1T} + 7x_{2T} + 16x_{3T} \geq 1400 \text{ // (richieste minime trenini Tanzania)}$$

$$10x_{1K} + 5x_{2K} + 14x_{3K} \geq 2100 \text{ // (richieste minime puzzle Kenya)}$$

$$4x_{1K} + 12x_{2K} + 2x_{3K} \geq 2400 \text{ // (richieste minime orsacchiotti Kenya)}$$

$$15x_{1K} + 7x_{2K} + 16x_{3K} \geq 1300 \text{ // (richieste minime trenini Kenya)}$$

$$x_{1T} + x_{1K} \leq 220 \text{ // (disponibilità pacchi centro 1 Tanzania/Kenia)}$$

$$x_{2T} + x_{2K} \leq 240 \text{ // (disponibilità pacchi centro 2 Tanzania/Kenia)}$$

$$x_{3T} + x_{3K} \leq 260 \text{ // (disponibilità pacchi centro 3 Tanzania/Kenia)}$$

Introduco la variabile binaria per i costi fissi e modifico la f.o.

Devo fare in modo che non parta l'aereo se non attivo il paese.

$$\min 10x_{1T} + 12x_{1K} + 15x_{2T} + 14x_{2K} + 5x_{3T} + 25x_{3K} + 500(y_{1T} + y_{1K}) + 300(y_{2T} + y_{2K}) \\ + 400(y_{3T} + y_{3K})$$

L'attivazione vale:

$$// y_{ij} \sim x_{ij} \quad x_{ij} \leq M y_{ij}, \forall i \in \{1,2,3\}, \forall j \in \{K, T\}$$

Non serve dimensionare per forza bene M ; basta dire “costante sufficientemente grande”; non sarebbe *del tutto* sbagliato dire $M \rightarrow +\infty$

Per modellare “il centro 2 che ha al massimo un aereo a disposizione”, usiamo le variabili.

$$y_{2T} + y_{2K} \leq 1$$

Il vincolo che “parta al massimo un aereo a destinazione” è già contenuto nell’attivazione (e non sarebbe necessario)

$$y_{ij} \leq 1$$

Io posso *decidere* se prendere la penalità o meno:

$w = 1$ se prendo la penalità di 1000 euro, 0 altrimenti

La variabile ha impatto sulla f.o., infatti aggiungerò $1000w$.

Se io decido di pagare la penalità, allora modifico il vincolo di puzzle della Tanzania:

$$10x_{1T} + 5x_{2T} + 14x_{3T} \geq 2500 + 500(1 - w)$$

Se $w = 0$, pago la penalità e sono libero di inviare 2500 pacchi

Se $w = 1$, non pago la penalità invio almeno 3000 pacchi

Il seguente vincolo sarebbe ridondante, perché modella la stessa cosa:

$$10x_{1T} + 5x_{2T} + 14x_{3T} \geq 3000(1 - w)$$

Domini: $x_{ij} \in \mathbb{Z}_+, y_{ij} \in \mathbb{Z}_+, w \in \{0,1\}$

6. Per l’assortimento di scatole di cioccolatini, sono disponibili praline di forme (cuore, fiore, stella o chicco) e gusti (latte, fondente o caffè) diversi. Le praline sono acquistate dalla sede centrale in confezioni, ciascuna contenente praline della stessa forma e dello stesso gusto. Il numero di praline per confezione dipende dalla forma: 70 cuori, 50 fiori, 100 stelle o 200 chicchi. Il costo per confezione dipende dal gusto: 30 euro per il latte, 50 euro per il fondente e 40 euro per il caffè. Le disponibilità di confezioni per le diverse forme e i diversi gusti sono riassunti nella tabella sotto. Il confezionamento avviene in tre diversi stabilimenti. Ogni stabilimento produce lo stesso numero di scatole dello stesso peso, ma con una composizione diversa: ciascuno stabilimento richiede 900 kg di cioccolato in tutto, e una quantità minima diversa di cioccolato dei diversi gusti, come dalla tabella.

Gusto	Disponibilità				Richiesta minima (kg)		
	Cuore	Fiore	Stella	Chicco	Stab. 1	Stab. 2	Stab. 3
Latte	Sì	Sì	No	No	500	100	100
Fondente	Sì	No	Sì	No	100	500	100
Caffè	No	Sì	Sì	Sì	100	100	500

Ciascuna pralina a forma di cuore, fiore, stella e chicco pesa, rispettivamente, 30, 50, 20 e 10 grammi. Si scriva un modello di programmazione lineare che minimizzi i costi tenendo conto che:

- si vogliono acquistare almeno 10 confezioni di cuori di cioccolato fondente;
- prima della spedizione negli stabilimenti, le praline acquistate sono pretrattate su linee diverse a seconda della forma, indipendentemente dal gusto, e ogni linea ha un costo fisso di setup pari a 200 euro;
- si vogliono acquistare praline di almeno 3 forme, indipendentemente dal gusto;
- si può evitare di rifornire uno stabilimento a scelta, pagando 15 000 euro a un fornitore esterno.

x_{ij} : # confezioni del gusto $i \in \{L, F, C\}, j \in \{1,2,3,4\}$

$x_{F1} \geq 10$ //almeno 10 confezioni di cuori di cioccolato fondente

$$\min 30(x_{L1} + x_{L2}) + 50(x_{F1} + x_{F3}) + 40(x_{C2} + x_{C3} + x_{C4})$$

Queste confezioni devono soddisfare una richiesta minima;
di kg che arriva allo stabilimento $1 \geq 900$

Con le variabili decisionali che ho, non ho la possibilità di modellare il problema correttamente.
 y_{ij}^k : # di confezioni di gusto $i \in \{L, F, C\}$ per la forma $j \in \{1,2,3,4\}$ per lo stabilimento $k \in \{1,2,3\}$

Devo legare le variabili:

$$x_{L1} = y_{L1}^1 + y_{L2}^2 + y_{L3}^3$$

Questo sarebbe evitabile; non mi interessa lo stabilimento, ma semplicemente acquistare le richieste minime.

$$70x_{L1} * 0.03 + 70x_{F1} * 0.03 + 0.05 * 50(x_{L2} + x_{C2}) + 0.02 * 100(x_{F3} + x_{C3}) + 0.01 * 200(x_{C4}) \geq 3 * 900 \text{ (moltiplico la forma per il peso)}$$

(cioccol. Latte ≥ 500); non serve per forza avere una variabile a 3 indici

Tutorato (2) del prof: Esercizi di programmazione lineare, Simpleso, CCPD

$$\begin{aligned} 70x_{L1} * 0.03 + 0.05 * 50x_{L2} &\geq 500 + 100 + 100 \\ 0.03 * 70x_{F1} + 0.02 * 100x_{F3} &\geq 100 + 500 + 100 \\ 0.05 * 50x_{L2} + 0.02 * 100x_{C3} + 0.01 * 200x_{C4} &\geq 100 + 100 + 500 \end{aligned}$$

w_j : variabile logica che vale 1 se acquisto la forma $j \in \{1,2,3,4\}$, 0 altrimenti

Questo comporta una modifica della f.o., dato che aggiungeremo:

$$200(w_1 + w_2 + w_3 + w_4)$$

Per attivare le variabili abbiamo bisogno di un vincolo di big-M (sapendo che vale per il cioccolato e, quando questo vale 0, pretrattiamo rispetto alla singola forma; questo giustifica la somma:

$$\begin{aligned} x_{L1} + x_{F1} &\leq Mw_1 \\ x_{L2} + x_{F2} &\leq Mw_2 \\ x_{F3} + x_{L3} &\leq Mw_3 \\ x_{L4} &\leq Mw_4 \end{aligned}$$

“prima della spedizione negli stabilimenti, le praline acquistate sono pretrattate su linee diverse a seconda della forma, indipendentemente dal gusto”

$$w_1 + w_2 + w_3 + w_4 \geq 3$$

Se non rifornisco uno stabilimento, non sono costretto a dover rispettare le quantità di richiesta minima. Dal punto di vista specifico, ogni stabilimento non ha le stesse richieste; quindi, dovremmo associare più variabili binarie per ognuno.

Per esempio, introduciamo:

$z = 0$ se produco in stabilimento, 1 se non produco in stabilimento

In questo modo, andiamo ad introdurre in funzione obiettivo $-900z$ dato che non rispettiamo genericamente la richiesta minima; tuttavia, occorre indicizzarla.

$z = 1$ se non produco in stabilimento $i \in \{1,2,3\}$, 0 altrimenti

Vincolo di attivazione se scegliessi un certo stabilimento, legando le singole variabili

$$z_1 - z_2 - z_3 = 3 - z$$

Occorre aggiungere il non rispetto delle richieste minime prima:

Scritto da Gabriel

$$\begin{aligned} 70x_{L1} * 0.03 + 0.05 * 50x_{L2} &\geq 500z_1 + 100z_2 + 100z_3 \\ 0.03 * 70x_{F1} + 0.02 * 100x_{F3} &\geq 100z_1 + 500z_2 + 100z_3 \\ 0.05 * 50x_{L2} + 0.02 * 100x_{C3} + 0.01 * 200x_{C4} &\geq 100z_1 + 100z_2 + 500z_3 \end{aligned}$$

La f.o. è correttamente:

$$\min 30(x_{L1} + x_{L2}) + 50(x_{F1} + x_{F3}) + 40(x_{C2} + x_{C3} + x_{C4}) + 200(w_1 + w_2 + w_3 + w_4) + 15000z$$

Aggiungiamo i domini:

$$x_{ij} \in Z_+, z \in \{0,1\}, z_i \in \{0,1\}, w_i \in \{0,1\}$$

2) Risolvere il seguente problema di programmazione lineare partendo dalla base $B = \{x_1, x_2\}$.

$$\begin{aligned} \min \quad & x_1 + x_2 - x_3 \\ \text{s.t.} \quad & x_1 - 2x_2 + x_3 = 2 \\ & x_1 + 2x_2 - x_3 - x_4 = 2 \\ & x_1, x_3, x_4 \geq 0 \\ & x_2 \leq 0 \end{aligned}$$

Soluzione: $x^* = (2, 0, 0, 0)$ $z^* = 2$.

$$\begin{aligned} F.STD \quad & x_2 = -\widehat{x}_2, \quad \widehat{x}_2 \geq 0 \\ \min \quad & x_1 - \widehat{x}_2 - x_3 \\ \text{s.t.} \quad & x_1 + 2\widehat{x}_2 + x_3 = 2 \\ & x_1 - 2\widehat{x}_2 + x_3 = 2 \\ & x_1 - 2\widehat{x}_2 - x_3 - x_4 = 2 \\ & x_1, \widehat{x}_2, x_3, x_4 \geq 0 \end{aligned}$$

	x_1	$\widehat{x}_2$	x_3	x_4	z	$\bar{b}$
$-z$	1	-1	-1	0	-1	0
	1	2	1	0	0	2
	1	-2	-1	-1	0	2

In rosso l'elemento di pivot. Seguono i passaggi:

x_1	$\widehat{x}_2$	x_3	x_4	z	$\bar{b}$
1	-1	-1	0	-1	0
1	2	1	0	0	2
1	-2	-1	-1	0	2

x_1	$\widehat{x}_2$	x_3	x_4	z	$\bar{b}$
0	-3	-2	0	-1	-2
1	2	1	0	0	2
1	-4	-2	-1	0	0

$$R'_0 \Leftarrow R_0 - R_1$$

$$R'_2 \Leftarrow R_2 - R_1$$

x_1	$\widehat{x}_2$	x_3	x_4	z	$\bar{b}$
0	0	-1/2	3/4	-1	-2
1	0	0	-1/2	0	2
1	1	1/2	1/4	0	0

$$\begin{aligned} R'_0 &\Leftarrow R_0 + 3R_2 \\ R'_1 &\Leftarrow R_1 - 2R_2 \\ R'_2 &\Leftarrow R_2/4 \end{aligned}$$

F.C? $x_1, \widehat{x}_2$? Si Ottima Non so

Entra $\rightarrow x_3$

Scritto da Gabriel

Esce $\arg \min \left\{ \frac{2}{0}, 0 \right\} = \widehat{x}_2$

x_1	$\widehat{x}_2$	x_3	x_4	z	$\bar{b}$
0	1	0	1	-1	-2
1	0	0	-1/2	0	2
0	2	1	1/2	0	0

F.C.? Sì Amm.? Sì Ottima Sì

$$x_1^* = 2 \quad x_2^* = x_2 = 0 \quad x_3^* = 0 \quad x_4^* = 0, z_{\min} = 2$$

3) Risolvere il seguente problema di programmazione lineare.

$$\begin{aligned} \max \quad & -x_1 + 2x_2 + x_3 \\ \text{s.t.} \quad & 3x_1 + x_2 - 4x_3 \leq 4 \\ & x_1 - x_2 - x_3 \leq 10 \\ & x_1 - 2x_2 + 6x_3 \leq 9 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Soluzione: problema illimitato.

x_1	x_2	x_3	x_4	x_5	x_6	z	$\bar{b}$
1	-2	-1	0	0	0	-1	0
3	1	-4	1	0	0	0	4
1	-1	-1	0	0	1	0	10
1	-2	6	0	0	1	0	9

F.C.? Sì

Entra -> Tra x_2 e x_3 scelgo x_2 (Bland)

Esce $\rightarrow \arg \min \left\{ \frac{4}{1}, \frac{10}{-1}, \frac{9}{-2} \right\} = x_4$

Passaggi di pivoting $\rightarrow R'_0 = R_0 + 2R_1, R'_1 = R_1, R'_2 = R_2 + R_1, R_3 = R_3 + 2R_1$

x_1	x_2	x_3	x_4	x_5	x_6	z	$\bar{b}$
7	0	-9	2	0	0	-1	8
3	1	-4	1	0	0	0	4
4	0	-5	1	1	0	0	14
7	0	-2	2	0	1	0	17

F.C.? Sì

Ammissibile? Sì

Ottima? Non so

Illimitata? Sì $\rightarrow$ Colore azzurro per indicare la colonna dell'illimitato

6. Si consideri il seguente tableau del simplesso:

	x_1	x_2	x_3	x_4	x_5	x_6	z	b
x_2	0	0	-4	1	-7	0	-1	5
x_6	0	1	2	0	-6	0	0	8
x_1	0	0	2	-1	1	1	0	2
	1	0	3	4	2	0	0	3

Si dica, senza svolgere calcoli e fornendo una giustificazione teorica delle risposte:

- riusciamo a individuare una soluzione di base corrispondente? Quale? Possiamo subito dire se è ottima?
- perché non è consentita l'operazione di pivot sull'elemento evidenziato nel cerchio (1)?
- su quali elementi è possibile effettuare il pivot secondo le regole del simplesso (indipendentemente dalle regole anticiclo)?
- quale sarà il cambio base secondo le regole del simplesso e applicando la regola di Bland? Perché la soluzione di base ottenuta in seguito a questo cambio base è sicuramente degenera?

a) Riusciamo ad individuare una soluzione di base in quanto nelle righe dei vincoli esiste la matrice identità. La base è formata in riga dalle colonne che posseggono i valori della matrice identità, cioè $B = [x_2, x_6, x_1]$. Non possiamo dire se sia ottima, essendoci alcuni costi ridotti negativi.

Ragionamento in più:

(Nel tableau, esiste un costo ridotto negativo e quindi potrei far entrare in base un costo ridotto negativo. Per esempio, con θ che è il minimo dei rapporti: $5 + \bar{c}_h \theta < 0$).

Dato che la soluzione di base corrente non è degenera ed esiste un costo ridotto strettamente negativo, otterrò una f.o. con valore più basso e quindi sicuramente la soluzione non è ottima).

b) La regola del rapporto minimo individua, seguendo la logica $\frac{b}{x_i}$, come elemento minimo x_1 e non x_5

c) I pivot ammissibili sono le coppie di variabili che entrano/escono; entrano x_3 o x_5 .

x_3 in, x_6 out x_3 in, x_1 out x_5 in, x_1 out

d) Entra? x_5 ($3 < 5$) Esce? x_1 ($1 < 6$)

Attenzione: Se ci viene data una base da cui partire nell'esercizio del simplesso, ma risulta non ammissibile, occorre dirlo e motivarlo.

a) Enunciare le condizioni di complementarità primale-duale in generale

x e u ottime $\Leftrightarrow x$ è ammissibile primale, u è ammissibile duale, $x_j(u^T A_j - c_j) = 0 \forall j$, $(a_i^T x - b_i)u_i = 0 \forall i$

b) Applicare tali condizioni per dimostrare che $(x_1, x_2, x_3) = (-1, 0, -2)$ è sol. ottima del problema

$$\begin{aligned}
 &\max 3x_1 - x_2 \\
 &s. t. x_1 + x_2 \geq 1 \\
 &2x_1 + x_2 + x_3 \leq 0 \\
 &2x_1 - x_2 - x_3 = 4 \\
 &2x_1 - x_3 \geq -1 \\
 &x_1 \geq 0, x_2 \text{ libera}, x_3 \leq 0
 \end{aligned}$$

1) Ammissibile primale

$$\begin{aligned}x_1 + x_2 &= 1 \\2x_1 + x_2 + x_3 &= 0 \\2x_1 - x_2 - x_3 &= 4 \\3x_1 - x_3 &= 4 > 1 \\x_1 > 0, x_2 = 0, x_3 < 0\end{aligned}$$

Tutto OK

2) Duale

$$\begin{aligned}\min u_1 + 0u_2 + 4u_3 - u_4 \\s. t. u_1 + 2u_2 + 2u_3 + 2u_4 &\geq 3 \\u_1 + u_2 - u_3 &= -1 \\u_2 - u_3 - u_4 &\leq 0 \\u_1 \leq 0, u_2 \geq 0, u_3 \text{ libera}, u_4 &\leq 0\end{aligned}$$

Verso del vincolo primale letta da dx a sx e corrispondeva variabile ≤ 0 per i segni; analogamente tutto il resto.

3) Applico CCPD

$$\begin{aligned}x_1(u_1 + 2u_2 + 2u_3 + 2u_4 - 3) &= 0 \rightarrow x_1 = 0 \rightarrow \text{CCPD1} \\ \Rightarrow u_1 + 2u_2 + 2u_3 + 2u_4 &= 3\end{aligned}$$

$x_2(u_1 + u_2 - u_3 + 1) = 0$ (vero per ammissibilità duale; cioè, non dipende da $x_2 = 0$, dato che è variabile libera e quindi $0 \cdot$ corrispondente scarto è uguale a 0 per l'ammissibilità duale)

$$x_3(u_2 - u_3 - u_4) = 0 \rightarrow \text{CCPD2}$$

$$u_1(x_1 + x_2 - 1) = 0 \rightarrow u_1 \cdot 0 = 0 \rightarrow //$$

$$u_2(2x_1 + x_2 + x_3 - 0) = 0 \rightarrow u_2 \cdot 0 = 0 \rightarrow //$$

$$u_3(2x_1 - x_2 - x_3 - 4) = 0 \rightarrow \text{Vero per ammissibilità primale}$$

$$u_4(2x_1 - x_3 + 2) = 0 \rightarrow u_4 \cdot 5 = 0 \rightarrow u_4 = 0 \rightarrow \text{CCPD3}$$

4) Sistema equazioni da CCPD e A.D.

$$\begin{aligned}u_1 + 2u_2 + 2u_3 + 2u_4 &= 3 \text{ (CCPD1)} \\u_2 - u_3 - u_4 &= 0 \text{ (CCPD2)} \\u_4 &= 0 \text{ (CCPD3)} \\u_1 + u_2 - u_3 &= -1 \text{ (A.D.)}\end{aligned}$$

$$u_2 = 1, u_1 = -1, u_3 = 1, u_4 = 0$$

5) Verifica ammissibilità duale

$$u_2 > 0 \quad u_1 < 0 \text{ (corretti)}$$

$$u_3 > 0 \quad u_4 = 0 \text{ (corretti)}$$

Vincoli:

Scritto da Gabriel

$$\begin{aligned}u_1 + 2u_2 + 2u_3 + 2u_4 &\leq 3 \\u_1 + u_2 - u_3 &= -1 \\u_2 - u_3 - u_4 &\leq 0\end{aligned}$$

6)

x è ammissibile primale, u è ammissibile duale

è CCPD → ottime

Tutorato (3): Svolgimento esame 23-01-2021

1. Un'acciaieria acquista rottame di tre tipi (T1, T2, T3) per ottenere due leghe (LA, LB). I rottami hanno contenuto percentuale diverso di tre metalli, e le leghe ottenute dalla loro fusione devono avere un contenuto in percentuale massimo degli stessi metalli, come da tabella. Quando si fondono

assieme dei rottami, si ottiene una stessa quantità di lega. Il costo dei rottami, in Keuro a tonnellata, è di 2.2, 1.9 e 2.1 per T1, T2 e T3 risp.

Le leghe sono vendute in container

da 10 tonnellate ciascuno, al prezzo di 50 / 60 Keuro a container di LA / LB. Saranno venduti 100 container in tutto. Scrivere un modello di programmazione lineare per massimizzare i profitti (ricavi – costi) tenendo conto che:

- per la lega LA, si vogliono impiegare almeno 10 tonnellate di T1 e 5 di T2;
- si devono vendere almeno 20 container di LA e 40 di LB;
- gestire ciascun tipo diverso di rottame determina un costo fisso di 1000 euro;
- si vogliono acquistare almeno due tipi di rottame (acquisto minimo 10 tonnellate);
- **se si acquistano almeno 100 tonn. di T1, il prezzo scende a 2.0 keuro a tonn.**

Metallo	Composizioni rottami			massimo	
	T1	T2	T3	LA	LB
Piombo	40%	30%	25%	30%	45%
Zinco	35%	40%	35%	60%	35%
Stagno	25%	30%	40%	40%	55%

Introduciamo una variabile decisionale considera i container venduti per la lega $j \in \{A, B\}$

y_j : container venduti per la lega $j \in \{A, B\}$

$$\max 50y_A + 60y_B$$

Questo non basta; dobbiamo introdurre anche le tonnellate, in quanto suggerito dal primo vincolo:

q_i : tonnellate di rottami di tipo $i \in \{1,2,3\}$

$$\max 50y_A + 60y_B - 2.2q_1 + 1.9q_2 + 2.1q_3$$

(Si noti la sottrazione in quanto si tratta di costi, pertanto da sottrarre al ricavo)

Il primo vincolo dice: “per la lega LA, si vogliono impiegare almeno 10 tonnellate di T1 e 5 di T2”

Occorre decidere come miscelare i metalli rispetto alle leghe; quindi, si introduce una variabile a due indici.

x_{ij} : tonnellate di $i \in \{1,2,3\}$ usati per $j \in \{A, B\}$

Come tale, si modellano:

$$x_{1A} \geq 10, x_{2A} \geq 5$$

Ora consideriamo che “le leghe ottenute dalla loro fusione devono avere un contenuto in percentuale massimo degli stessi metalli:

$$0.4x_{1A} + 0.3x_{2A} + 0.25x_{3A} \leq 0.6 + y_A \text{ oppure } \leq 0.3(x_{1A} + x_{2A} + x_{3A})$$

$$0.35x_{1A} + 0.4x_{2A} + 0.35x_{3A} \leq 0.6(x_{1A} + x_{2A} + x_{3A})$$

$$0.25x_{1A} + 0.3x_{2A} + 0.4x_{3A} \leq 0.4(x_{1A} + x_{2A} + x_{3A})$$

$$0.4x_{1B} + 0.3x_{2B} + 0.25x_{3B} \leq 0.45(x_{1B} + x_{2B} + x_{3B})$$

$$0.35x_{1B} + 0.4x_{2B} + 0.35x_{3B} \leq 0.35(x_{1B} + x_{2B} + x_{3B})$$

Scritto da Gabriel

$$0.25x_{1B} + 0.3x_{2B} + 0.4x_{3B} \leq 0.55(x_{1B} + x_{2B} + x_{3B})$$

Le variabili y sono legate alle x ; in questo modo, leghiamo tutte le quantità x_{ij} perché “le leghe sono vendute in container da 10 tonnellate ciascuno”, il che significa che le singole tonnellate sono vendute con almeno un margine sulle 10 massime:

$$y_A = \frac{(x_{1A} + x_{2A} + x_{3A})}{10}, y_B = \frac{(x_{1B} + x_{2B} + x_{3B})}{10}$$

Pure le quantità q vanno legate alle x ; quindi;

$$q_1 = x_{1A} + x_{1B}, q_2 = x_{2A} + x_{2B}, q_3 = x_{3A} + x_{3B}$$

“Si devono inoltre vendere almeno 20 container di LA e 40 di LB” $\rightarrow y_A \geq 20, y_B \geq 40$

“Saranno venduti 100 container in tutto” $\rightarrow y_A + y_B = 100$

Si deve inoltre “modellare il costo fisso di 1000 euro sul tipo di rottame”, quindi:

z_i : variabile logica che vale 1 se compro il rottame di tipo $i \in \{1,2,3\}$, 0 altrimenti

Questo comporta che in funzione obiettivo si aggiunge il seguente pezzo:

$$-1(z_1 + z_2 + z_3)$$

(dove si moltiplica per -1 in quanto inteso come $-1,000$ essendo in k/euro il problema)

Attiviamo i vincoli sul rottame:

$$q_1 \leq Mz_1, q_2 \leq Mz_2, q_3 \leq Mz_3$$

M costante sufficientemente grande

“Si vogliono acquistare almeno due tipi di rottame”

$$z_1 + z_2 + z_3 \geq 2$$

Dato inoltre che si vogliono acquistare almeno 10 tonnellate, “se vuoi comprare quel tipo di rottame devi averne comprato almeno 10 tonnellate”; $q_1 \geq 10z_1, q_2 \geq 10z_2, q_3 \geq 10z_3$

Alternativamente, si può compattare come:

$q_i \geq 10z_i, \forall i \in \{1,2,3\}$ (che non cambia niente, ma lo sintetizza)

L’ultimo punto è “se si acquistano almeno 100 tonnellate di T1, il prezzo scende a 2 k/euro a tonnellata”

La decisione da prendere è data dall’acquisto scontato, dunque:

s : variabile logica che vale 1 se acquisto scontato, 0 altrimenti

In f.o. avrò:

$$\max 50y_A + 60y_B - 2.2q_1^S + 1.9q_2 + 2.1q_3 - 1(z_1 + z_2 + z_3) - 2.0q_1^S$$

Quindi, il dimensionamento è dato dall’acquisto non scontato (N) rispetto all’acquisto scontato (S):

$$q_1 = q_1^S + q_1^N \text{ (prezzo = prezzo scontato + prezzo normale)}$$

$$q_1^S \geq 100s \text{ (se acquisto scontato, esiste almeno la quantità 100 tonnellate)}$$

$$q_1^S \leq Ms \text{ (grande quanto vuole, in quanto lo sconto esiste)}$$

Se $s = 1$, a prescindere dalla sua grandezza, vale che: $q_1 = q_1^S$

Se $s = 0$, vale che acquistiamo a prezzo non scontato.

Domini: $y_j \in \mathbb{Z}_+, x_{ij} \in \mathbb{R}_+, q_i \in \mathbb{R}_+, z_i \in \{0,1\}, s \in \{0,1\}, q_i^S \in \mathbb{R}_+, y_i^N \in \mathbb{R}_+, \forall i \in \{1,2,3\}, \forall j \in \{A,B\}$

2. Si consideri il seguente problema di programmazione lineare:

$$\begin{array}{ll} \max & x_2 - 4x_3 \\ \text{s.t.} & x_1 - 2x_2 + x_3 \leq 2 \\ & x_1 - x_2 + 2x_3 \geq -2 \\ & -2x_1 + x_2 - x_3 \leq 0 \end{array}$$

3. $x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \leq 0$

a) lo si risolva con il metodo del simplesso, applicando la regola anticiclo di Bland;

b) possiamo dedurre qualche informazione sul corrispondente problema duale direttamente a partire dal risultato del punto precedente? in base a quale teorema?

$$\begin{array}{ll} \max & x_2 - 4x_3 \\ \text{s.t.} & x_1 - 2x_2 + x_3 \leq 2 \\ & x_1 + x_2 + 2x_3 \leq -2 \\ & -2x_1 + x_2 - x_3 \leq 0 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \leq 0 \end{array}$$

(Scritto in forma standard)

$$\begin{array}{ll} \min & -x_2 - 4\widehat{x}_3 \\ \text{s.t.} & x_1 - 2x_2 - \widehat{x}_3 + x_4 = 2 \\ & x_1 + x_2 + 2\widehat{x}_3 + x_5 = -2 \\ & -2x_1 + x_2 + \widehat{x}_3 + x_6 = 0 \\ & x_1 \geq 0, x_2 \geq 0, \widehat{x}_3 \leq 0 \end{array}$$

Punto (a)

$B_0 \sim \text{VAR SLACK/SURPLUS}$

entra x_2 , esce x_6
 entra $\widehat{x}_3$, esce x_2

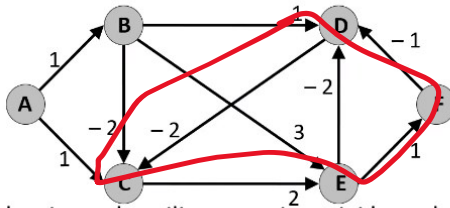
0	0	-1	0	3	-1	-1	4
0	0	-1	1	2	-1	0	8
1	0	4	0	1	-4	0	2
0	1	3	0	2	-1	0	5

IL MIN!

Punto (b)

In base al fatto che il problema primale è illimitato, il problema duale è inammissibile (corollario del teorema della dualità debole).

3. Calcolare i cammini minimi **con al più 5 archi** dal nodo A verso **tutti** gli altri nodi.



- si scelga l'algoritmo da utilizzare e si motivi la scelta;
- si applichi l'algoritmo scelto (riportare e giustificare i passi dell'algoritmo in una tabella);
- si riporti un cammino minimo con al più 5 archi da A verso C (giustificare)
- è possibile, basandosi solo sulla tabella del punto b e senza ulteriori calcoli, determinare l'esistenza di un ciclo negativo?

a) Nella scelta dell'algoritmo, notiamo che ci viene dato un massimo numero di archi da dover rispettare, pertanto si può applicare solo l'algoritmo di Bellman – Ford, l'unico che dà la possibilità di calcolo dei cammini minimi sulla base di un massimo numero di archi. Applicheremo Bellman – Ford fermandoci alla quarta iterazioni, con un numero di archi e iterazioni pari a $k \leq 5$

b)

Iterazione	Nodo A	Nodo B	Nodo C	Nodo D	Nodo E	Nodo F	Aggiornamenti
Inizio	0_A	$+\infty_A$	$+\infty_A$	$+\infty_A$	$+\infty_A$	$+\infty_A$	1
$h = 1$	0_A	1_A	1_A	$+\infty_A$	$+\infty_A$	$+\infty_A$	B, C
$h = 2$	0_A	1_A	-1_B	2_B	3_C	$+\infty_A$	C, D, E
$h = 3$	0_A	1_A	-1_B	1_C	1_C	4_E	D, E, F
$h = 4$	0_A	1_A	-1_B	-1_E	1_C	2_E	D, F
$h = 5$	0_A	1_A	-3_D	-1_E	1_C	2_E	D

c) Un cammino minimo (con al più 5 archi) può essere individuato seguendo la catena dei predecessori

$F \Rightarrow E \Rightarrow C \Rightarrow B \Rightarrow A$ che equivale al percorso con costo $1 + (-2) + 2 + 1 = 2$

A livello di algoritmo, si termina con $flag_aggiornato=true$, pertanto esiste un ciclo negativo, in quanto ad ogni iterazione è stata aggiornata almeno un'etichetta.

d) Basandosi solo sulla tabella e senza ulteriori calcoli, è possibile determinare il ciclo negativo perché siamo arrivati fino in fondo ai calcoli con etichette instabili (quindi, con nodi appena aggiornati o su cui si ha avuto almeno un aggiornamento)

e) È possibile disegnare albero e grafo dei cammini minimi? No, in quanto abbiamo etichette instabili (no soluzione ammissibile duale e quindi no cammini minimi).

Dimostriamo ad esempio esiste il ciclo $[C, D, E, F]$; come tale, partiamo dal nodo iniziale e seguiamo per i primi nodi quelli dettati dal percorso dei predecessori e poi seguiamo il percorso che vogliamo fare noi, in questo caso ciclo; come si vede, esiste un vertice che era già presente nel cammino e che, idealmente, lo migliora ad n iterazioni.

Risposte con limiti di archi (al più 5 archi)

e) È possibile disegnare albero e grafo dei cammini minimi? No, in quanto abbiamo etichette instabili (no soluzione ammissibile duale e quindi no cammini minimi). Fermandosi all'iterazione 5, vedo che c è stato aggiornato e, seguendo da questo la catena dei predecessori, arrivo a costruire l'albero dei cammini minimi.

Senza fare calcoli, non posso sapere se le etichette si stabilizzeranno e quindi dato che non faccio l'iterazione 6, non riesco a fare il grafo con cammini minimi.

5. Si consideri il seguente tableau del simplesso:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z	b
0	-1	0	-31	0	-2	0	-1	-7
0	10	0	400	0	0	1	0	100
1	-33	0	15	-2	1	0	0	330
0	32	1	1	5	-1	0	0	320

Senza operazioni di pivot e fornendo **giustificazione teorica delle risposte**:

- si può individuare una soluzione di base corrispondente? qual è? è ottima?
- su quali elementi sarebbe possibile effettuare il pivot secondo le regole del simplesso (indipendentemente dalle regole anticiclo)?
- considerando le variabili ordinate per indice crescente, quale sarà il cambio base secondo le regole del simplesso e applicando la regola di Bland?
- Qual è il valore della funzione obiettivo dopo il cambio base del punto c)?
- La soluzione di base ottenuta in seguito al cambio base del punto c) è degenerare oppure no?

a) Soluzione di base ottima data da $B = [x_7 \ x_1 \ x_3]$, in quanto ci sono le colonne della matrice identità. Avrò sicuramente una soluzione ammissibile perché $\bar{c} < 0$ (costo ridotto negativo), $\theta > 0$ (rapporto minimo positivo), dunque l'incremento della funzione obiettivo è positivo e migliora, dato che avremo

$$z_{min} = z + \bar{c}\theta > 0$$

b) Candidati per il cambio base (rapporti minimi) $\rightarrow x_2, x_4, x_6$

c) Con Bland esce x_3

d) $z_{new} = -(-z) + (-1)\frac{320}{32} = 7 - 10 = -3$ (si nota anche da qui non è ottima)

e) La soluzione è degenerare in quanto x_3 esce dalla base, ma x_7 assume valore 0, assieme alle altre rimanendo in base.

29/11/2022: Algoritmo di Dijkstra per il problema del cammino minimo

L'algoritmo di Dijkstra può essere applicato per calcolare i cammini minimi di un grafo $G = (N, A)$ i cui costi soddisfino il fatto di essere tutti non negativi, quindi: $c_{ij} \geq 0, \forall (i, j) \in A$.

Si noti che l'algoritmo è di *label setting*; rispetto infatti agli algoritmi label correcting che mantengono delle etichette π fissate come ottime tutte insieme solo all'ultima iterazione, qui fissa direttamente ad ogni iterazione il valore ottimo di un'etichetta, riducendo la complessità computazionale dell'analisi.

Esso determina i valori ottimi delle etichette usando due sottoinsiemi S e $\bar{S}$, con S che sono etichette fissate (già ottime) e $\bar{S}$ etichette che devono ancora essere fissate. Ad ogni iterazione, scegliamo un nodo che ha etichetta minima, spostando un nodo da S ad $\bar{S}$ (fase di label setting); se ho svuotato $\bar{S}$ mi fermo, altrimenti itero.

Per tutti i nodi che abbiano un nodo origine s e un insieme di successori Γ_i , avendo un'etichetta ottima, aggiorni il predecessore e l'etichetta, iterando.

Scritto da Gabriel

In pseudocodice:

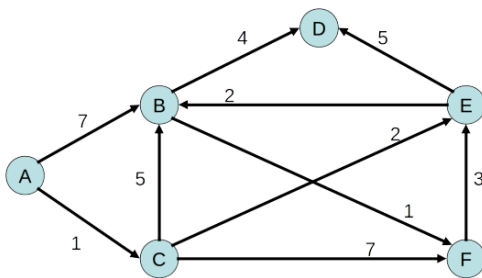
Algoritmo di Dijkstra

```

0)  $\pi_s := 0$ ; set  $p(s) := \wedge$ ; set  $S := \emptyset$ ; set  $\bar{S} := N$ ;
   for each  $v \in N \setminus \{s\}$  { set  $\pi_v := +\infty$ , set  $p(v) := \wedge$  }
1) set  $\hat{v} := \arg \min_{i \in \bar{S}} \{\pi_i\}$ 
   set  $S := S \cup \{\hat{v}\}$ ; set  $\bar{S} := \bar{S} \setminus \{\hat{v}\}$ ;
   if  $\bar{S} = \emptyset$  then STOP:  $\pi$  è ottimo.
2) for all (  $j \in \Gamma_{\hat{v}} \cap \bar{S} : \pi_j > \pi_{\hat{v}} + c_{\hat{v}j}$  ) do {
   set  $\pi_j := \pi_{\hat{v}} + c_{\hat{v}j}$ 
   set  $p(j) := \hat{v}$ ;
}
go to 1)

```

Vediamo un esempio, in cui possiamo applicare Dijkstra in quanto tutti gli archi hanno costo non negativo:



Ad ogni iterazione, scelgo il nodo ad etichetta minima $\bar{v}$ tra tutti quelli che non hanno etichetta fissata, mettendo un asterisco per indicare che è il nodo fissato (quindi, ad ogni iterazione controllo i nodi con numeri nella riga precedente e metto l'asterisco sul nodo che ha costo minimo). In tale modo, si inserisce l'etichetta corrente in S , continuando ad operare le verifiche del vincolo duale.

Il trattino si mette quando ho controllato un'etichetta per vedere se fosse ottima, ma non ho aggiornato. La crocetta/segno "x" si mette quando l'etichetta è già in $\bar{S}$ e non la devo considerare.

In questo modo, controllo tutti gli archi e, complessivamente, arrivo all'ultima iterazione in cui ho sempre preso il costo minimo e li ho controllati tutti.

it.	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	$\bar{S}$	$\hat{v}$
0	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A, B, C, D, E, F	
1	*	7 ($\wedge$)	1 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	B, C, D, E, F	A
2		6 ($\wedge$)	*		3 ($\wedge$)	8 ($\wedge$)	B, D, E, F	C
3		5 ($\wedge$)		8 ($\wedge$)	*		B, D, F	E
4		*		—		6 ($\wedge$)	D, F	B
5					x	*	D	F
6				*			$\emptyset$	D

*: etichetta fissata.

—: etichetta controllata ma non aggiornata.

x: etichetta non controllata perché il nodo è già fissato.

A Dijkstra non importa nulla dei nodi non raggiunti da $\bar{v}$; qua non controlliamo tutti gli archi, ma solo quelli che escono da $\bar{v}$ e stanno ancora in $\bar{S}$.

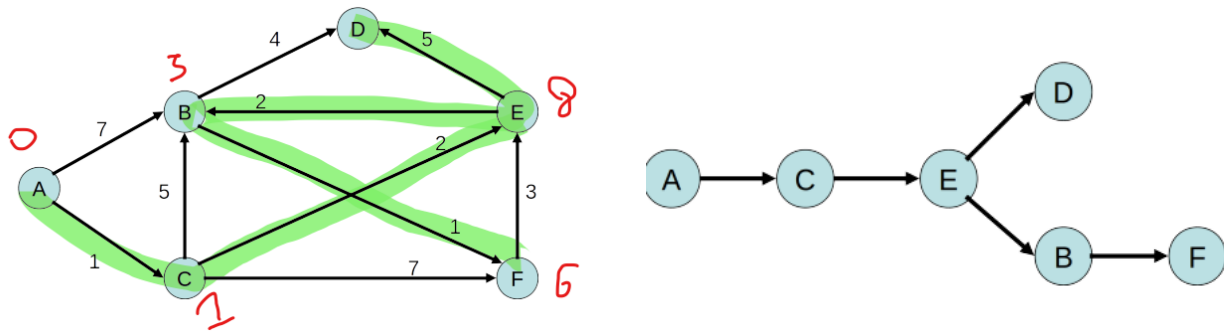
Nel momento in cui si fissano le etichette, le altre si lasciano vuote.

La verifica si opera sempre sulla base della condizione di Bellman (vincolo duale) sugli archi uscenti.

L'algoritmo ottiene una soluzione ottima e duale ammissibile.

Troviamo l'albero dei cammini minimi, seguendo la catena dei predecessori (segnata in verde).

Poi, segniamo le etichette ottime; in questo caso albero e grafo dei cammini minimi coincidono, non essendoci altri cammini minimi. Nell'immagine di destra, vediamo una rappresentazione equivalente:



Dato che, ad ogni iterazione, almeno un nodo lascia $\bar{S}$, serviranno $|N|$ iterazioni per la convergenza. Rispetto ad un algoritmo label correcting, l'algoritmo di Dijkstra limita le verifiche della condizione di Bellman (vincoli duali) ai soli archi uscenti dal nodo selezionato ed entranti in un nodo correntemente nell'insieme $\bar{S}$. In merito alla convergenza e correttezza:

Lemma 4 Alla fine di ogni iterazione, per ogni nodo $i \in N$:

- se $i \in S$, π_i è il costo di un cammino minimo dal nodo origine s a i , $p(i)$ è il predecessore di i su tale cammino, e tale cammino utilizza solo nodi in S ;
- se $i \in \bar{S}$, π_i è il costo di un cammino minimo dal nodo origine s a i che utilizza solo nodi in S per arrivare a i , e $p(i)$ è il predecessore di i su tale cammino.

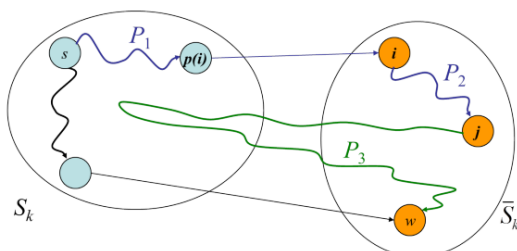
Concettualmente, non possiamo ottenere un percorso migliore, dati i predecessori e il fatto che, togliendo un nodo che è percorso minimo ad ogni iterazione, il percorso è ottimo. Infatti, avendo che i cammini contengono almeno un nodo in $\bar{S}$, ci sarà sempre un cammino di costo inferiore (lo stesso non vale se ci troviamo in S , in quanto non sarebbe possibile trovare cammini miglioranti).

Per la dimostrazione, consideriamo nodi $j \in \bar{S}$:

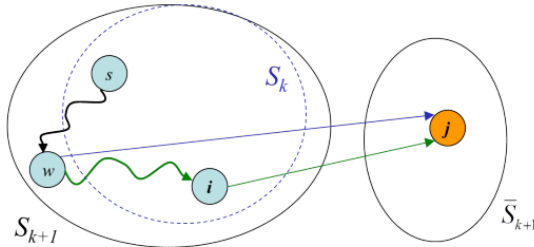
- j è successore di s ($j \in \Gamma_s$), pertanto l'unico cammino è quello che va da s a j con costo c_{sj} e si modifica l'etichetta aggiornando il costo;
- j non è successore di s ($j \notin \Gamma_s$), pertanto non esistono cammini da s a j e quindi l'etichetta di j non viene aggiornata.

Alla prima iterazione, ciò è vero. Induttivamente:

- alla fine dell'iterazione, si aggiunge un nodo w ai nodi precedenti, non potendo arrivarci con costi migliori dato che i nodi stanno in $\bar{S}$;
- il cammino di w utilizza nodi che stavano in $\bar{S}$ e dimostriamo che si tratta del cammino minimo, usando anche il suo predecessore;
- il costo minimo π_w è tale se utilizzo nodi in $\bar{S}_k$; se arrivo con un costo inferiore, dovrei per forza uscire da $\bar{S}_k$ e dai relativi predecessori (dato che è minimo a $\bar{S}_k$ lo sarà a $\bar{S}_{k+1}$);
- uscendo da k , qualsiasi altro costo è più grande di π_w , in particolare se consideriamo w_i , tutti con costi ≥ 0 (funziona perché, altrimenti, non esisterebbe più la catena dei predecessori);
- partendo quindi da un P_1 , avremo che P_2 e P_3 entrambi con costo non negativo, avranno costo minore, contraddicendo l'assurdo di cercare un arco "fuori"



- l'inserimento di w in S_{k+1} può portare alla riduzione del costo con cammino minimo da s a j , passando per $w \in S_{k+1}$. In particolare:
 - o se $j \in \Gamma_w$, avremo solo nodi in S_k ed è cammino di costo minimo;
 - o se $j \notin \Gamma_w$ la sua etichetta potrebbe essere migliorata, non essendo w e quindi non è già cammino minimo.



Proprietà 10 Dato un grafo pesato $G = (N, A)$ e un nodo origine $s \in N$, l'algoritmo di Dijkstra risolve il problema del cammino minimo dall'origine s verso tutti gli altri nodi in tempo $O(|N|^2)$.

Per la complessità:

- consideriamo una complessità lineare di analisi di tutti i nodi $\rightarrow O(n)$
 - cerchiamo il minimo e dobbiamo trovare un minimo dipendente dalla cardinalità di tutti gli elementi; usando una lista, avremo $n * O(n) = O(n^2)$
 - controlliamo gli archi che escono da un solo nodo, con $n * O(n) = O(n^2)$
 - la complessità sarebbe $O(n) + O(n^2) + O(n^2) = O(n^2)$
 - calcolando invece la complessità ammortizzata (cioè, considerando tutte le iterazioni e nel caso peggiore; se le operazioni costose sono poco frequenti, compenso "ammortizzando" con operazioni poco costose in modo deterministico), consideriamo ogni volta gli archi che escono dall'etichetta ottima; controlliamo *al massimo una volta* tutti gli archi.
- Avremo quindi $O(m)$, $m = |A|$

La complessità finale, usando S e $\bar{S}$ come liste è $O(|N| + |N|^2 + |A|) = O(|N|^2)$

Implementando invece S e $\bar{S}$ come heap, possiamo fare la ricerca di minimo in tempo costante, che ora ci costa almeno $O(n)$, dato che, se fosse min-heap, sta nella radice.

- Per creare un heap (*create*), costerà $O(1)$.
- Per l'inserimento (*insert*), dovremo mantenere la proprietà di heap (facendo in modo che il nodo padre sia più grande/più piccolo di tutti i nodi figli e sia bilanciato, quindi con numero minimo di figli), con costo $O(\log(n))$
- Si trova il minimo (*find-min*), che sarà nel nodo radice, quindi $O(1)$
- Quindi, eliminare il minimo (*delete-min*) costerà $O(\log(n))$, dato che percorriamo l'albero un numero di passi pari alla sua profondità
- L'aggiornamento dell'etichetta (*decrease-key*) costerà $O(\log(n))$, perché si cerca un elemento e poi si ristabiliscono le proprietà di heap

La complessità finale, usando S e $\bar{S}$ come heap, sarebbe $O(|A| (1 + \log(|N|) + 1)) = O(|A| \log(|N|))$

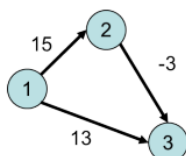
Quindi, avremo $O(n^2)$ vs $O(m \log(n))$; tuttavia:

- se avessimo a che fare con un grafo completo/denso, meglio implementare con le liste;
- se avessimo a che fare con un grafo sparso, meglio implementare con gli heap.

Se usassimo un Fibonacci heap, si dimostra che la complessità sarà $O(|A| + |N| \log |N|)$.

Un Fibonacci heap è una struttura dati che consiste in un insieme di alberi che seguono la proprietà min heap o max heap. Esso si chiama così perché gli alberi sono costruiti in modo tale che un albero di ordine n abbia almeno F_{n+2} nodi al suo interno, dove F_{n+2} è l' $(n + 2)$ -esimo numero di Fibonacci. Tutte le operazioni di heap sono più efficienti di quelle supportate da altri heap, dato che lavorano a costo ammortizzato $O(1)$

Attenzione che lavoriamo sull'ipotesi di costi positivi (se esiste anche solo un costo negativo, non possiamo applicare l'algoritmo di Dijkstra, anche se siamo sicuri che non esistono cicli negativi). Infatti, nell'esempio che segue, applicando Dijkstra si fissa il costo del nodo 3 a 13 alla prima iterazione (mentre, con Bellman-Ford, sarebbe 12):

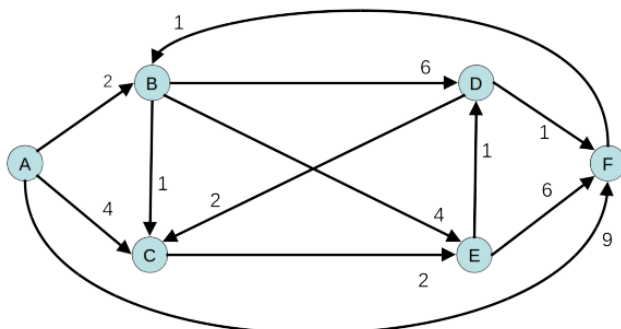


Grafo con costo negativo: non si può applicare Dijkstra.

Non andrei ad aggiornare altri nodi che non sono da $\bar{v}$, dato che Dijkstra non ragiona così.

Facciamo un altro esempio, in cui i costi sugli archi sono tutti non negativi e possiamo applicare l'algoritmo di Dijkstra, il più efficiente tra quelli visti.

Esempio 13 Calcolare i cammini minimi per il grafo in Figura 23 a partire dal nodo A, e riportare l'albero e il grafo dei cammini minimi.



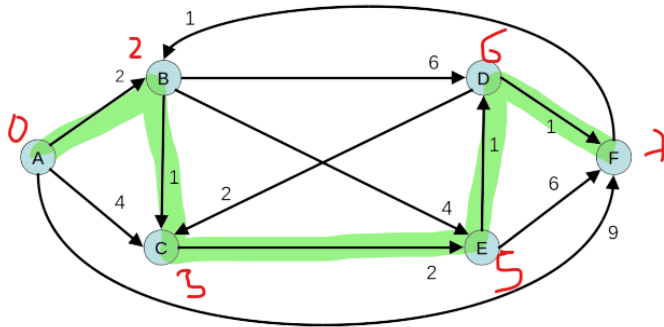
it.	nodo A	nodo B	nodo C	nodo D	nodo E	nodo F	$\bar{S}$	$\hat{v}$
0	0 ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	A, B, C, D, E, F	//
1	*	2 (A)	4 (A)	$+\infty$ ($\wedge$)	$+\infty$ ($\wedge$)	9 (A)	B, C, D, E, F	A
2		*	3 (B)	8 (B)	6 (B)		C, D, E, F	B
3			*		5 (C)		D, E, F	C
4				6 (E)	*	—	D, F	E
5			×	*		7 (D)	F	D
6		×				*	$\emptyset$	F

*: etichetta fissata.

—: etichetta controllata ma non aggiornata.

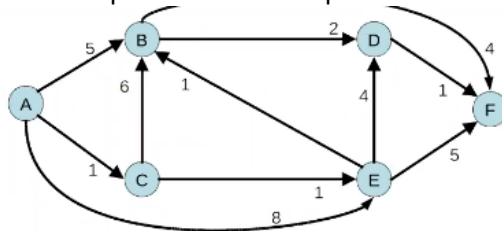
×: etichetta non controllata perché il nodo è già fissato.

Costruiamo albero e grafo dei cammini minimi (seguendo a ritroso la catena dei puntatori); anche in questo caso coincidono.



Vediamo alcuni esempi di situazioni un po' critiche:

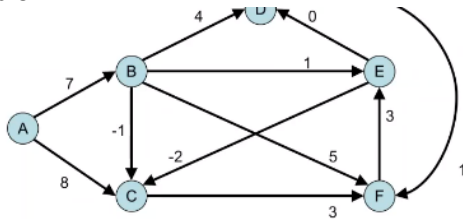
Esercizi...



- Trovare i cammini minimi da A verso tutti gli altri nodi: posso usare Bellman-Ford? Posso usare Dijkstra? Quale conviene usare?
 - Trovare i cammini minimi da A verso tutti gli altri nodi con al massimo 3 archi: quale algoritmo uso?
- Possiamo usare Bellman-Ford o Dijkstra, dato che il primo fa tutto in ogni caso, ma il secondo solo in caso di costi ridotti negativi. Conviene usare il secondo, dato che ha complessità computazionale migliore
 - Usiamo per forza Bellman-Ford quando abbiamo al massimo X archi; come nel caso della domanda "cammini minimi verso tutti gli altri nodi con al massimo 3 archi", si usa sempre BF
 - Se avessimo una situazione in cui fisso ad un certo punto un nodo, per esempio all'iterazione 3. In quel momento ho un cammino con al più 2 archi e che ha solo nodi in S. A priori non sappiamo quando fissiamo l'etichetta nel caso di Dijkstra; questo succedere in casi fortunati ma potrebbe succedere che i cammini minimi, in realtà, abbiano più archi; in quel caso non si può usare Dijkstra (controesempio rispetto al caso precedente).
Bellman-Ford ce lo garantisce, Dijkstra casualmente può arrivarci.
 - Normalmente, usiamo Dijkstra, avendo complessità $O(n \log n)$ piuttosto che la $O(n * m)$ di Bellman-Ford
 - Se ci viene detto che esiste un ciclo negativo, è possibile calcolare i cammini minimi verso tutti gli altri nodi? No, in quanto le etichette non sono stabili
 - Pur esistendo un ciclo negativo, posso calcolare un cammino con al massimo X archi? Sì, in quanto è cammino con massimo numero di hop.
 - Se avessi "7821" hop o un numero qualsiasi, comunque, ogni iterazione trova un costo migliore e mi troverei, tornando indietro un ciclo negativo.

Altro esempio:

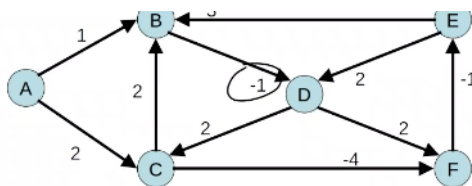
Esercizi...



- Trovare i cammini minimi da A verso tutti gli altri nodi: posso usare Bellman-Ford? Posso usare Dijkstra? Quale conviene usare?
- Trovare i cammini minimi da A verso tutti gli altri nodi con al massimo 3 archi: quale algoritmo uso?

- Qui non uso Dijkstra dato che ci sono archi con costi ridotti negativi, devo usare Bellman-Ford.
- Anche nel secondo caso, uso BF avendo il massimo numero di archi

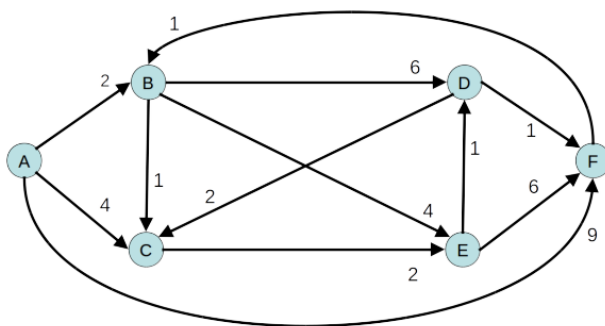
Esercizi...



- Trovare i cammini minimi da A verso tutti gli altri nodi: posso usare Bellman-Ford? Posso usare Dijkstra? Quale conviene usare?
- Trovare i cammini minimi da A verso tutti gli altri nodi con al massimo 3 archi: quale algoritmo uso?

- Qui non uso Dijkstra dato che ci sono archi con costi ridotti negativi, devo usare Bellman-Ford.
- Anche nel secondo caso, uso BF avendo il massimo numero di archi

Esercizio 2 Calcolare i cammini minimi con al massimo 3 archi per il grafo in Figura 23 a partire dal nodo A e indicare un cammino minimo da A a D con al massimo 3 archi (descrivendo il metodo seguito per ottenerlo).



Anche se non ci sono archi con costo negativo, NON possiamo applicare Dijkstra, in quanto l'unico algoritmo presentato in grado di gestire il vincolo sul massimo numero di archi è quello di Bellman-Ford.

Laboratorio 6 - Flussi con Reti

Ricordiamo la base del problema, cercando di implementare il problema del cammino minimo:

Variabili decisionali: $x_{ij} = \begin{cases} 1, & \text{l'arco } (i,j) \text{ è sul cammino minimo;} \\ 0, & \text{altrimenti.} \end{cases}$

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = \begin{cases} -1, & v = s; \\ +1, & v = d; \\ 0, & v \in N \setminus \{s, d\}. \end{cases} \\ & x_{ij} \in \{0, 1\} \equiv \mathbb{R}_+ \quad \forall (i,j) \in A \end{aligned}$$

Caso **particolare MCF**: un nodo domanda s e un nodo offerta d

Prendendo il grafo $G(N, A)$, abbiamo indici di nodi (N) ed archi (A). Abbiamo poi che i costi sono definiti come prodotto cartesiano sugli archi.

Notiamo che prendiamo nodo iniziale e nodo finale $v = s$ oppure $v = d$; per esempio, potremmo definire un parametro *symbolic* per indicare con $A = s$ e $d = F$.

Introduciamo un parametro di bilanciamento come $b\{N\}$ e successivamente la variabile $x\{A\}$

La f.o. viene definita sul doppio indice i, j all'interno di A , in qualità di prodotto cartesiano; pertanto, è come fosse definito da una coppia di indici.

I vincoli di bilanciamento sono indicizzati sui vertici v per ogni nodo N ; l'insieme A viene individuato da coppie di elementi, tale che la sommatoria sia scritta nei termini corretti

Il file *sp.mod* viene così definito:

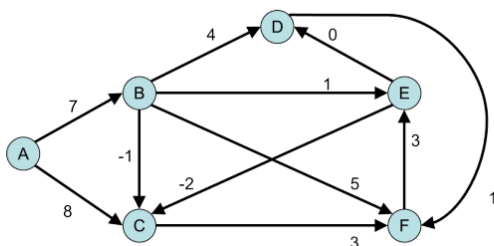
```
set N; #nodi
set A within N cross N; #archi definiti su prodotto cartesiano

param c{A}; #i costi sono definiti sugli archi,
#ma sono definiti come coppie di nodi
param b{N}; #parametro di bilanciamento

var x{A};
minimize f: sum{(i,j) in A} c[i,j] * x[i,j];
s.t. balance{v in N}: sum{(i,v) in A} x[i,v] - sum{(v,j) in A} x[v,j] = b[v];
```

Dopo aver impostato correttamente il modello con un file *.run* minimale, seguiamo questa impostazione per il file *.dat*:

Si determini con AMPL il cammino minimo da A a F nel seguente grafo



Prendiamo i nodi e raggruppiamo i vertici a coppie (possono essere sia tra parentesi che non); similmente, inseriamo tutti i costi indicizzando a coppie nuovamente.

Scritto da Gabriel

Per i vincoli di bilanciamento, inseriamo il vincolo come fosse funzione indicatrice.

Impostando il file .run includendo anche il file .dat, una variabile va fuori limite (*unbounded*); mettendo x come variabile binaria, scorre tutto liscio (penserebbe nel caso contrario possano essere negative).

Quindi, nel file .mod aggiungere: **var** $x\{A\}$ **binary**;

Il file .dat sarà come segue:

```
set N := A B C D E F;
set A := A B A C B C B E B F B D
        C F D F E C E D F E;
```

Qui non posso mettere i dati sotto forma di (A,B) mentre qui per “c” beccherebbe i dati come (tr) e non si può metterli tra parentesi (errore capitato).

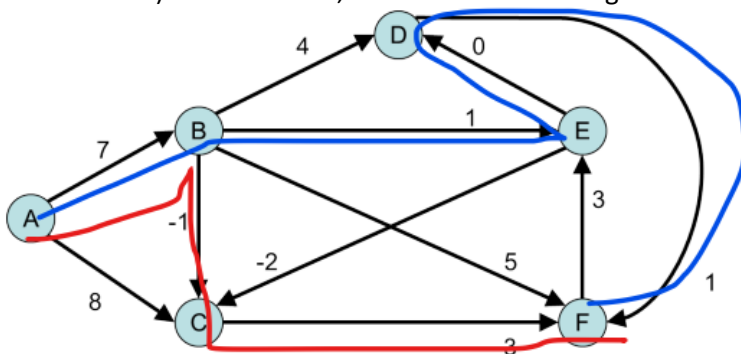
```
param c :=
A B 7
A C 8
B C -1
B E 1
B F 5
B D 4
C F 3
D F 1
E C -2
E D 0
F E 3
;
```

```
param b default 0 :=
A -1
F 1;
```

Il file .run invece come segue:

```
reset;
model sp.mod;
data sp.dat;
option solver cplex;
solve;
display f, x;
display {(i,j) in A: x[i,j] >0} x[i,j];
```

Dato che la matrice è TU, la soluzione trovata rimane comunque ottima. Cambiano l’ordine degli archi, trova sempre cammini minimi, ma in un diverso ordine (dato che applica algoritmi diversi per trovare la stessa soluzione). Come si vede, i cammini minimi nel grafo trovato, si ottiene:



Inserendo un’opzione di rilassamento dell’integralità nel file .run posta a 0, il problema viene risolto con variabili intere discrete. Posta ad 1, invece, usa variabili continue. Quindi, “risolvi con/senza vincoli di interezza”.

```
option relax_integrity 0 (oppure) 1;
```

Scritto da Gabriel

Noi vogliamo che la somma delle iterazioni sia minore di h , cioè numero di hop/iterazioni.

In questo caso, il numero massimo di hop deve essere 3.

Risolvendo con variabili continue, troviamo 3 soluzioni. Nonostante il vincolo in più, il risolutore trova variabili intere all'ottimo. Con il rilassamento continuo, trovo comunque variabili intere.

Quindi, nel file .mod, inseriamo la seguente cosa:

```
s.t. maxhop: sum{(i,j) in A} x[i,j] <= h;
```

Immaginiamo di essere in una rete stradale; sappiamo che ogni arco ha un costo e un consumo di energia w_{ij} associato rispetto al serbatoio W . Quindi, da aggiungere al modello il vincolo che modella i costi rispetto a variabili e consumo.

Il file .mod è come segue, inserendo W e modificando opportunamente il vincolo di max hop.

In particolare, il vincolo da inserire è $\sum_{(i,j)} w_{ij} x_{ij} \leq W$

```
set N; #nodi
set A within N cross N; #archi definiti su prodotto cartesiano

param c{A}; #i costi sono definiti sugli archi,
#ma sono definiti come coppie di nodi
param b{N}; #parametro di bilanciamento
param h := 3;
param W;

var x{A} binary;
minimize f: sum{(i,j) in A} c[i,j] * x[i,j];
s.t. balance{v in N}: sum{(i,v) in A} x[i,v] - sum{(v,j) in A} x[v,j] = b[v];
s.t. budget: sum{(i,j) in A} w[i,j]*x[i,j] <= W;
```

Modifichiamo il .mod come segue:

```
set N; #nodi
set A within N cross N; #archi definiti su prodotto cartesiano

param c{A}; #i costi sono definiti sugli archi,
#ma sono definiti come coppie di nodi
param b{N}; #parametro di bilanciamento
param h := 3;
param w{A};
param W;
var x{A} >= 0;
minimize f: sum{(i,j) in A} c[i,j] * x[i,j];
s.t. balance{v in N}: sum{(i,v) in A} x[i,v] - sum{(v,j) in A} x[v,j] = b[v];
s.t. budget: sum{(i,j) in A} w[i,j]*x[i,j] <= W;
```

Invece, il .run come segue:

```
set N := A B C D E F;
set A := A B  A C  B C  B E  B F  B D
        C F  D F  E C  E D  F E;

param : c      w:=
A B    7      32
A C    8      14
B C   -1      11
B E    1       2
B F    5      14
B D    4      10
C F    3      48
D F    1      33
E C   -2       9
E D    0      14
F E    3      18
;
```

```
param b default 0 :=
A -1
F 1;

param W := 50;
```

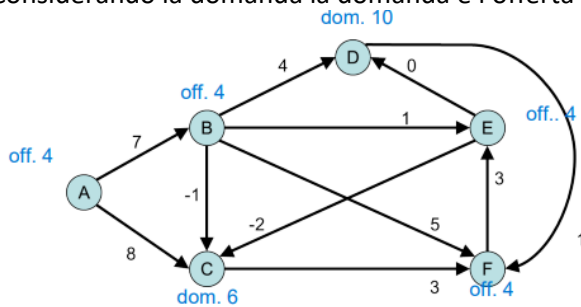
Quando aggiungo il vincolo sul budget, perdo le proprietà della matrice, forzo la variabile come binaria e risolvo ottenendo le variabili intere.

Ora, consideriamo il seguente esercizio:

Il grafo rappresenta una rete di concessionarie di automobili: i nodi corrispondono alle concessionarie e gli archi alla possibilità di spostare automobili tra concessionarie, con relativi costi per automobile (si consideri il valore assoluto). I nodi C e D hanno ordini che eccedono la disponibilità per rispettivamente 6 e 10 automobili, mentre le altre concessionarie hanno ciascuna un eccesso di 4 automobili. Le bisarche utilizzate permettono di trasferire al massimo sei automobili per ogni coppia. Determinare il piano di trasporti di costo minimo.

Si tratta di un problema di flusso di costo minimo, con quattro nodi offerta e due nodi domanda.

Considerando la domanda la domanda e l'offerta in base alla disponibilità, avremo la situazione seguente:



Per organizzare i trasporti, posso usare vari giri, naturalmente. Quindi, devo modellare il problema come flusso di costo minimo. In questo modo, modelliamo x_{ij} come quantità di auto da i a j .

In questo modo, sostituiamo il modello imponendo un vincolo di bilanciamento, usando il modello di prima con variabili intere. Non cambiano i costi e, per quanto riguarda i vincoli, rimane uguale.

Attenzione, invece, ad inserire le variabili come intere:

Cammino minimo: modello

Variabili decisionali: $x_{ij} = \begin{cases} 1, & \text{l'arco } (i, j) \text{ è sul cammino minimo;} \\ 0, & \text{altrimenti} \end{cases}$ quantità di auto da "i" a "j"

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,v) \in A} x_{iv} - \sum_{(v,j) \in A} x_{vj} = \begin{cases} -1, & v = s; \\ +1, & v = d; \\ 0, & v \in N \setminus \{s, d\}. \end{cases} \\ & x_{ij} \in \{0, 1\} \end{aligned}$$

Caso **particolare MCF**: un nodo domanda s e un nodo offerta d

Dobbiamo far uscire 4 unità in più rispetto a quelle che entrano; questo comporta una modifica in f.o. Il file `.mod` sarebbe quindi modificato in questo modo, commentando le parti che non servono, dunque budget e vincoli binari/max hop.

```

set N; #nodi
set A within N cross N; #archi definiti su prodotto cartesiano

param c{A}; #i costi sono definiti sugli archi,
#ma sono definiti come coppie di nodi
param b{N}; #parametro di bilanciamento
var x{A} integer >=0;
minimize f: sum{(i,j) in A} c[i,j] * x[i,j];
s.t. balance{v in N}: sum{(i,v) in A} x[i,v] - sum{(v,j) in A} x[v,j] = b[v];

```

Il file .dat è invece come segue, commentando w:

```

set N := A B C D E F;
set A := A B A C B C B E B F B D
        C F D F E C E D F E;

param : c :=      #w:=
A B  7          #32
A C  8          #14
B C -1          #11
B E  1          #2
B F  5          #14
B D  4          #10
C F  3          #48
D F  1          #33
E C -2          #9
E D  0          #14
F E  3          #18
;

param b default 0 :=
A -4
B -4
C 6
D 10
E -4
F -4
;

```

Inserendo 7 sul parametro *c* nel file .dat scritto qui sopra, il problema diventa *infeasible* come si vede:

```

ampl: include sp.run;
CPLEX 20.1.0.0: integer infeasible.
6 MIP simplex iterations
0 branch-and-bound nodes
No basis.
f = 0

```

```

x :=
A B  0
A C  0
B C  0
B D  0
B E  0
B F  0
C F  0
D F  0
E C  0
E D  0
F E  0
;

```

```
x[i,j]; #empty
```

Scritto da Gabriel

Ricerca operativa semplice (per davvero)

Questo significa che non esiste un'assegnazione di valori interi alle variabili che possa soddisfare tutti i vincoli oppure eliminando uno qualsiasi di questi vincoli sarà possibile risolvere il problema.

Il fatto che “c” non possa essere 7 non è di per sé un vincolo definito, ma implicito del problema (precondizione); in particolare, la somma di tutte le “b” deve essere 0; quindi, nel .mod inseriamo un controllo affinché la somma dei $b[i]$ sia = 0.

Quindi, si inserisce con la keyword *check* la seguente linea nel file .mod:

```
check: sum{i in N} b[i] = 0;
```

Reinserendo il vincolo $c = 7$ nel file .dat, ecco che si solleva l’eccezione:

```
ampl: include sp.run;
check at line 9 of sp.mod fails:
  check: sum{i in N} b[i] == 0;
```

Definiamo il vincolo di capacità sugli archi come segue nel file .mod, inserendo un parametro e il vincolo relativo:

```
param u{A};
s.t. capacity{(i,j) in A}: x[i,j] <= u[i,j];
```

Nel file .dat, inseriamo i vincoli di capacità, ma non rovinano le proprietà della matrice, dato che la struttura del problema mantiene lo stesso valore in matrice e vincoli.

Questo sotto risponde al vincolo “le bisarche utilizzate permettono di trasferire al massimo sei automobili per ogni coppia”.

param	:	c	u:=	#w:=
A B	7		6	#32
A C	8		6	#14
B C	-1		6	#11
B E	1		6	#2
B F	5		6	#14
B D	4		6	#10
C F	3		6	#48
D F	1		6	#33
E C	-2		6	#9
E D	0		6	#14
F E	3		6	#18

“I nodi C e D hanno ordini che eccedono la disponibilità per rispettivamente 6 e 10 automobili, mentre le altre concessionarie hanno ciascuna un eccesso di 4 automobili”.

```
param b default 0 :=
A -4
B -4
C 6
D 10
E -4
F -4
;
```

Se metto dei vincoli di capacità sugli archi, preservò le proprietà di TU sulla matrice; le variabili, comunque, vengono intere anche con rilassamento continuo. Quindi, dovrei “rassegnarmi” ad utilizzare algoritmi per trovare soluzioni intere.

Domanda: Cambierebbe la soluzione se le unità fossero frazionabili?

Risposta: Facendo un po’ di prove, su alcuni vincoli, la f.o. migliora (per esempio, modificando due vincoli e inserendo ad esempio 7/2, cioè 3.5 in AMPL); in altri casi, inserendo coefficienti minori, salgono vincoli impliciti sulle variabili (il loro valore deve essere dentro i bound, differenza fattibile con i termini che si hanno), nella maggior parte dei casi, la f.o. non si modifica e rimane con lo stesso valore.

Scritto da Gabriel

Quindi, per una variabile fuori base, ci si deve aspettare che la soluzione continui a essere valida per un intervallo di valori dei coefficienti delle variabili non fondamentali. L'intervallo dovrebbe includere tutti i valori inferiori del coefficiente e alcuni valori superiori. Se il coefficiente aumenta o diminuisce a sufficienza (e se l'inserimento della variabile nella base è fattibile), allora la soluzione cambia come visto qui.

Branch-and-Bound per problemi di Programmazione Lineare Intera

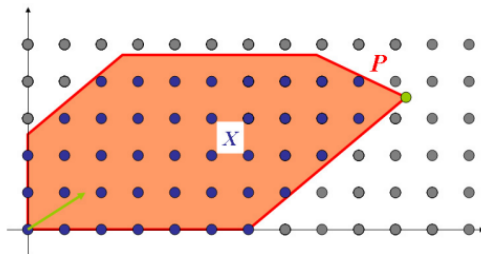
Consideriamo un problema PLI nella forma generica:

$$\begin{array}{ll} \min / \max & c^T x \\ \text{s.t.} & Ax = b \\ & x \in \mathbb{Z}_+^n \end{array}$$

considerando il poliedro P delle soluzioni ammissibili, ignorando il vincolo di interezza delle variabili:

$$P = \{x \in \mathbb{R}_+^n : Ax = b\}$$

Si definisce il *politopo* delle soluzioni ammissibili intere, il cui obiettivo è trovare una soluzione del problema PLI con la forma disequazione che si vede a lato.



$$x^* \in X: c^T x^* \leq c^T x, \forall x \in X$$

Per provare a risolvere, si possono provare diversi modi per trovare la soluzione (sfruttiamo il fatto che esiste un numero finito di soluzioni ottime, quindi graficamente nei punti blu)

- approssimare la soluzione, qualsiasi di questa si ottenga
 - o non funziona in quanto l'insieme delle soluzioni può essere esponenziale
- arrotondare la soluzione all'intero più vicino ("sperando" di trovare una soluzione ammissibile)
 - o non sempre è definitivamente individuabile
- scrivere il problema in forma TU, prendendo come forma ideale una formulazione lineare
 - o non si può sempre formulare un problema di punti come formulazione lineare; considerando che il problema non può essere risolto con metodi di programmazione lineare a variabili reali e l'approssimazione intera potrebbe non essere ottima/ammissibile, dato anche il numero di vincoli
- ipotizzare la soluzione per rilassamento continuo attraverso "piani di taglio/cutting plane method", che permettono iterando di "tagliare" progressivamente il problema inserendo vincoli che vengono violati dalla soluzione corrente e aggiungendoli al problema (da noi non trattati)

Consideriamo quindi un *problema di ottimizzazione combinatoria*, definito come segue:

$$\begin{array}{ll} \min / \max & f(x) \\ \text{s.t.} & x \in X \end{array}$$

dove X è un insieme *FINITO* di punti e $f(x)$ è una generica funzione obiettivo.

Un esempio di questi problemi è il citato caso del cammino minimo, oppure colorazione di un grafo, etc.

Il metodo esatto per problemi di ottimizzazione combinatoria che consideriamo si basa su uno schema enumerativo (brute force) che sfrutta la finitezza dello spazio delle soluzioni ammissibili.

Lo schema è detto *Algoritmo universale per ottimizzazione combinatoria* (definito Scemo del prof, ndr):

- 1 generare tutte le possibili soluzioni x ;
- 2 verificare l'ammissibilità della soluzione $x \in X$;
- 3 valutare $f(x)$
- 4 scegliere la x ammissibile cui corrisponde la migliore $f(x)$.

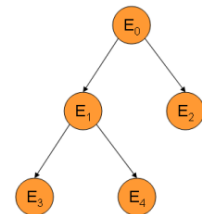
Si vedono subito i limiti:

- la valutazione di $f(x)$ potrebbe non essere banale
- la cardinalità di X potrebbe essere molto elevata.
- Questa genera due questioni:
 - o come generare lo spazio delle soluzioni ammissibili?
 - o come esplorare efficientemente lo spazio delle soluzioni?

Per capire come generare le soluzioni ammissibili, osserviamo che:

Dato un problema di ottimizzazione combinatoria $z = \max / \min \{f(x) : x \in X\}$ e data una suddivisione della regione ammissibile X in insiemi $X_1, \dots, X_n$: $\bigcup_{i=1}^n X_i = X$, sia $z^{(k)} = \max / \min \{f(x) : x \in X_k\}$. Allora la soluzione del problema $z = \max / \min_{k=1, \dots, n} z^{(k)}$.

Usiamo un algoritmo bottom-up per cercare la migliore soluzione, piuttosto che top-down. L'idea è di prendere X e dividerlo in sotto-insiemi più piccoli (approccio *divide et impera/divide and conquer*); in questo modo, dividendo le regioni ammissibili, scelgo la migliore tra le migliori soluzioni del problema. La divisione ricorsiva dell'insieme delle soluzioni dà luogo ad un *albero delle soluzioni ammissibili*.



Devo fare in modo che l'operazione di suddivisione di un nodo E_i non vada a perdere "dei pezzi"; compresa la soluzione ottima, valendo sempre ricorsivamente.

Tecnicamente, dato un problema P_0 e l'insieme delle soluzioni $E_0 = X$, con E_0 radice dell'albero ed E_i insieme di soluzioni associate, quanto a lato garantisce che le soluzioni presenti nel nodo padre debbano essere presenti in (almeno) uno dei suoi figli.

$$E_i = \bigcup_{j \text{ figlio di } i} E_j$$

Una caratteristica *auspicabile, ma non necessaria*, è la disgiunzione dei sottoinsiemi figli come segue, fermandosi nel momento in cui ogni nodo contiene una sola soluzione.

$$E_j \cap E_k = \emptyset, \forall j, k \text{ figlio di } i$$

Questa operazione di costruzione dei nodi figli per partizione di un nodo padre è detta branch.

Si cita il *problema dello zaino 0-1 (0-1 knapsack problem)*, evidentemente rappresentabile con un'operazione di branch (viene solo citata come esempio rappresentativo, col prof che al posto di c_i rappresenta p_i con lo stesso significato).

Gli scorsi anni veniva affrontato come esercizio anche in esame; ora viene mostrato come esempio di riferimento per il branching binario (puramente visivo e non pratico).

Il problema espresso in maniera più formale diventa:

- ognuno degli N oggetti possiede un peso w_i e un valore c_i ;
- si indica con W il peso massimo sopportabile dallo zaino;
- la possibilità che un oggetto venga inserito o meno nello zaino è espressa dalle variabili intere x_i .

La *funzione obiettivo* da massimizzare è:

$$Z = \sum_{i=1}^N c_i \cdot x_i.$$

I vincoli:

$$\sum_{i=1}^N w_i \cdot x_i \leq W.$$

In base al tipo di variabili x_i si ha poi la distinzione in:

- **problema dello zaino 0-1**

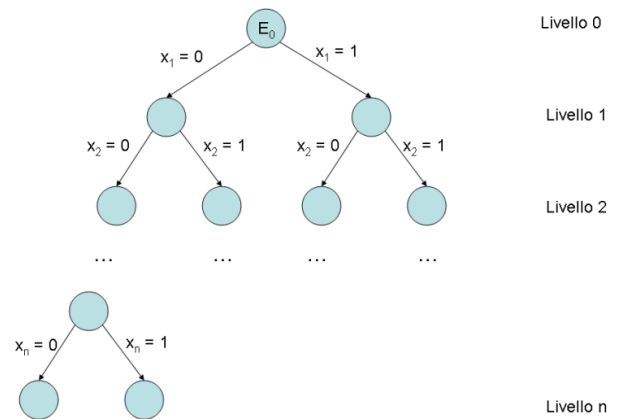
$$x_i = \{0, 1\} \quad \forall i = 1, \dots, N,$$

ogni oggetto può esserci o non esserci senza ripetizioni (abbiamo un solo esemplare di ciascun oggetto);

Consideriamo un problema di ottimizzazione combinatoria con n variabili binarie
 $x_i \in \{0,1\}, i = 1, \dots, n$.

In questo caso, dato un problema P_i e il corrispondente insieme di soluzioni ammissibili E_i , possiamo facilmente ottenere due sotto-problemi e due sottoinsiemi di E_i fissando una delle variabili binarie a 0 per un sotto-problema e a 1 per l'altro sotto-problema. Utilizzando questa regola di *branch binario* (ogni nodo è suddiviso in *due* nodi figli), otterremmo l'albero delle soluzioni.

$$x_i \in \{0,1\}, i = 1..n$$



Ad ogni passo, esplora le diverse direzioni ammesse; ci chiediamo:

- possiamo sfruttare il parallelismo?
- esiste un criterio per scartare cammini parziali?

Cerchiamo quindi un metodo che ci permetta di esplorare solo aree “buone” della regione ammissibile, cercando di escludere a priori che la soluzione ottima del problema si possa trovare in altre aree. Per fare questo, consideriamo, per ogni nodo dell'albero, un bound, ossia una valutazione ottimistica (non peggiore) del valore che la funzione obiettivo può assumere.

Ipotizzando una costruzione di branching costruito dal nodo sorgente ai nodi finali, costruiamo percorsi considerabili cammino minimo. Sviluppando gli alberi in profondità, otterremo un cammino minimo. Utilizziamo i bound per determinare il percorso delle soluzioni ammissibili (prendendo un certo insieme di foglie che abbia valore conveniente, dunque $\leq$ a quello attuale).

L'algoritmo di branch è anche definito *Algoritmo A* per path finding*, che è un algoritmo di ricerca su grafi che individua un percorso da un dato nodo iniziale verso un dato nodo goal (o che passi un test di goal dato). Questo permette di capire quali sono i nodi ammissibili (nodi “open”), escludendo a priori nodi che “collidono” con altri (nodi “closed”). Tramite la verifica di condizioni, vediamo se la soluzione raggiunta nei sottoinsiemi, valutata a mano a mano nel branching, sia sempre ottima (escludendo le soluzioni che *non portano* a quella ottima).

Ad esempio, consideriamo un problema di minimizzazione con variabili binarie, visto come combinazione di progetti e la scelta di almeno uno tra questi e supponiamo di disporre di una soluzione ammissibile di valore 10:

$$\min z = 11x_1 + 9x_2 + 10x_3 + y$$

$$\text{s.t. } x_1 + x_2 + x_3 \geq 1$$

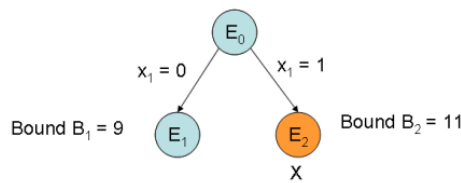
...molti altri vincoli su x_i e y (non formulabili)...

$$x_i \in \{0,1\}, \quad y \geq 0$$

È noto che $x_3 = 1, x_1 = x_2 = 0$ è ammissibile (con $z = 10$)

Supponiamo di associare, all'insieme delle soluzioni ammissibili per il problema di partenza, il nodo E_0 e di sviluppare il primo livello dell'albero di branching, fissando la variabile x_1 a 0 e a 1 e ottenendo due nodi figli E_1 ed E_2 .

Usiamo un branching binario, cercando di capire se le soluzioni possono essere ammissibili:



Il valore della funzione obiettivo è dato dai valori ottimi nei sottoinsiemi E_1 ed E_2 , visto come:

$$z = \min\{z^{(1)}, z^{(2)}\}$$

Supponiamo ora di riuscire a stabilire, con qualche ragionamento e senza considerare ad una ad una le soluzioni, che il valore delle soluzioni ammissibili per il problema, una volta fissato a 0 il valore di x_1 non possa essere minore di 9.

In altre parole, il valore della funzione obiettivo in corrispondenza di ciascuna delle soluzioni rappresentate dal nodo E_1 è sicuramente non inferiore a 9. 9 rappresenta quindi una valutazione ottimistica della funzione obiettivo per il sottoinsieme E_1 , un limite inferiore (*lower bound*) sotto il quale il valore della funzione obiettivo non può scendere, se consideriamo solo soluzioni in E_1 : $z^{(1)} \geq 9$.

Analogamente, supponiamo di disporre di un lower bound per E_2 e sia tale bound pari a 11: nessuna delle soluzioni con $x_1 = 1$ (soluzioni in E_2) ha un valore della funzione obiettivo più basso di 11: cioè $z^{(2)} \geq 11$

Assumendo con l'albero di prima che metta $x_1 = 0$ oppure $x_1 = 1$ avremo che la soluzione, escludendo vincoli, sarà 9 oppure 11.

Quello che stiamo dicendo è che: è facile ottenere un bound, dunque una valutazione ottimistica di un problema diviso in branch, quindi nei sottoproblemi.

Secondo la nostra prima osservazione avremo: $z = \min\{z^{(1)}, z^{(2)}\}$ e con $z \leq 10$, e $z^{(2)} \geq 11 \geq 10$ non è possibile trovare una soluzione con valore migliore di 10 tra le soluzioni nel nodo E_2 .

Quindi: E_2 non contiene sicuramente la soluzione ottima ed è inutile sviluppare ed esplorare il sotto-albero con radice E_2 . Lo stesso non vale per E_1 dato che una delle soluzioni potrebbe essere minore di 10 (dato che abbiamo assunto essere sicuramente ≤ 9) e quindi contenere la soluzione ottima.

Inutile quindi andare oltre il 10; cerchiamo nel ramo inferiore a 10 se esiste qualcosa di meglio.

Grazie all'approccio ricorsivo, riusciamo a trovare una specifica soluzione dentro ad un problema.

In generale, se è nota una soluzione ammissibile, la disponibilità di un bound associato ai nodi dell'albero di branch ci permette di "potare" (non sviluppare, *pruning*) sotto-alberi che sicuramente non contengono la soluzione ottima, cioè i sotto-alberi radicati in un nodo con bound non migliore del valore della f.o., sondando l'albero ("*fathom*", scartando alcuni nodi sulla base dell'algoritmo).

- $z = \min\{z^{(1)}, z^{(2)}\}$, e anche $z \leq 10$ ^X
- $z^{(1)} \geq 9$
- $z^{(2)} \geq 11(\geq 10)$ $\implies$ Possiamo **potare** E_2 !!!

Dalle osservazioni precedenti, è possibile definire il metodo del Branch and Bound (B&B) per la soluzione di problemi di ottimizzazione combinatoria, che permette di ottenere la migliore soluzione attualmente presente (*incumbent solution*), risolvendo un problema diverso (*rilassato*, togliendo/modificando vincoli ed evitando di sviluppare tutti i sottoalberi [bound]), con una valutazione ottimistica. La ricerca è breadth first sugli alberi sulla base dell'algoritmo A* definito prima.

- **Branch:** costruzione dell'albero delle soluzioni (enumerazione ricorsiva)
- **Soluzione ammissibile** (incumbent solution): valore possibile, ma non dimostrabilmente ottimo
- **Bound:** valutazione ottimistica della funzione obiettivo per le soluzioni associate ad un nodo (sottoalbero)
- **Fathom:** se il bound di un nodo non è migliore dell'incumbent, il relativo sottoalbero si può potare

L'esplorazione (*fathoming*) esegue un'enumerazione implicita nello spazio delle soluzioni; infatti dice, "se l'albero è peggiore da un punto di vista di soluzione, non considerarlo". Infatti, non esploro tutte le soluzioni, ma è come se le avessi generate.

Il metodo del Branch-and-Bound può essere schematizzato come segue. Dato un problema di ottimizzazione combinatoria $z = \min / \max \{f(x) : x \in X\}$, sia:

- P_0 : problema di ottimizzazione iniziale;
- L : lista dei nodi aperti. Ogni nodo è una coppia (P_i, B_i) , dove P_i è il sotto-problema e B_i è il relativo bound;
- $\bar{z}$: valore della migliore soluzione ammissibile.
- $\bar{x}$: migliore soluzione ammissibile corrente;

- Quando il problema è di minimo
 - o cerco un lower bound (LB) minimo e ho un upper bound (UB) come soluzione ammissibile
- Quando il problema è di massimo
 - o cerco un upper bound (LB) massimo e ho un lower bound (UB) come soluzione ammissibile

Inizializzazione: Esegui una stima ottimistica B_0 della funzione obiettivo e poni $L = \{(P_0, B_0)\}$, $\bar{x} = \emptyset$, $\bar{z} = +\infty(\min)[- \infty(\max)]$

Repeat:

Criterio di Stop: Se $L = \emptyset$, allora **stop**: $\bar{x}$ è la soluzione ottima.

Selezione nodo: Seleziona $(P_i, B_i) \in L$ per effettuare il branch

Branching: Dividi P_i in t sotto-problemi $P_j, j = 1..t$ ($\cup_j P_j = P_i$)

For each sottoproblema $j = 1..t$

Bounding: Valuta una stima ottimistica B_j ottenendo informazioni (soluzione "rilassata" x_j^R o inammissibilità)

Fathoming: Se P_j non è ammissibile, **continue**

Se B_j non è migliore di $\bar{z}$ ammissibile, **continue**

Se x_j^R ammissibile:

se x_j^R anche migliore di $\bar{z}$:

aggiorna $\bar{z} \leftarrow B_j$, $\bar{x} \leftarrow x_j^R$

elimina da L nodi k con L_k non migliore di $\bar{z}$;

continue

Ricorsione: Atrimenti, aggiungi (P_j, B_j) a L

Branch-and-Bound per problemi di Programmazione Lineare Intera

Per implementare un algoritmo B&B per uno specifico problema, bisogna determinare i seguenti elementi;

- a. Regole di Branching: come costruire l'albero delle soluzioni.
- b. Calcolo del Bound: come valutare i nodi.
- c. Regole di Fathoming: come chiudere e dichiarare sondati (fathomed) i nodi.
- d. Regole di esplorazione dell'albero: definire le priorità di visita dei nodi aperti.
- e. Come valutare una o più soluzioni ammissibili (soluzioni da confrontare con i bound per chiudere nodi).
- f. Criteri di stop: condizioni di terminazione dell'algoritmo.

Esaminando l'algoritmo, discorsivamente si individuano questi passi:

- in ogni nodo, valutiamo il bound (sapendo che la sol. non può essere più alta di P_0 all'iterazione iniziale)
- seleziono un nodo, togliendolo dalla lista e poi eseguendo il branching (dividendo in sottoproblemi che rimangono "in sospeso", da capire se ha senso inserirli sulla lista)
- per ogni sottoproblema, si cerca di capire cosa succederà "alla meglio" (in alcuni casi, si "rilassa" il problema, alleviando/togliendo alcuni vincoli; se il problema rilassato non fosse ammissibile, similmente non lo sarà anche il problema con più vincoli)

Valutando il bound, succede che:

- o la soluzione non è ammissibile
- o la soluzione non è migliore di quella ammissibile
- o la soluzione è ottima per il rilassamento ma anche tutto il problema (se è ottima per il problema rilassato, è ottima anche per il sottoproblema); potrebbe essere migliore anche della incumbent solution
 - in questo caso, cerco ricorsivamente un bound $\leq$ della incumbent solution e continuo ad esplorare se lo trovo, iterando

Vediamo un esempio:

Esempio (dummy): scelta ottima di appalti

Una grossa azienda di costruzioni edili deve decidere la combinazione ottimale degli appalti da accettare per la costruzione degli edifici A , B e C .

I profitti attesi per i tre edifici sono di 3, 5 e 7 milioni di euro rispettivamente.

L'azienda dispone di 4 ruspe speciali e gli edifici richiedono risp. 3, 2 e 3 ruspe.

È possibile inoltre affittare fino a due altre ruspe speciali per la durata dei lavori, al costo di un milione di euro a ruspa.

Associamo a ciascun appalto la variabile decisionale binaria x_i che vale 1 se si decide di accettare l'appalto, 0 altrimenti.

Consideriamo inoltre la variabile intera y associata al numero di ruspe affittate.

Le decisioni sono $24 = 8 * 3$ (numero di ruspe da affittare per ciascuno dei tre edifici)

Infatti, abbiamo che tutte le ruspe sono $8 = 3 + 2 + 3$ e questo vale per ciascuno dei tre edifici, da cui il numero precedente.

Per questo problema, abbiamo che:

Branch $\rightarrow$ Scegliere una decisione e creare un sottoproblema per ogni valore

Bound $\rightarrow$ Somma profitti di tutti gli appalti possibili meno costo ruspe fissate (valutazione imprecisa ma ottimistica e veloce, senza ragionamenti su ruspe "necessarie")

Il ragionamento del branching è:

- al primo livello, decido se fare l'appalto 1
- al secondo livello, decido se fare l'appalto 2
- al terzo livello, decido se fare l'appalto 3
- all'ultimo livello, decido quale ruspa prendere

Il ragionamento del bounding è:

- sommo i profitti di tutti gli appalti possibili meno costo ruspe fissate (quindi, ai primi livelli dell'albero, decido se prendere o non prendere l'appalto ("sì" oppure "no"); quindi, sommo tutti gli appalti che non sono stati ancora fissati a "no" e vado a sottrarre solo le ruspe affittate

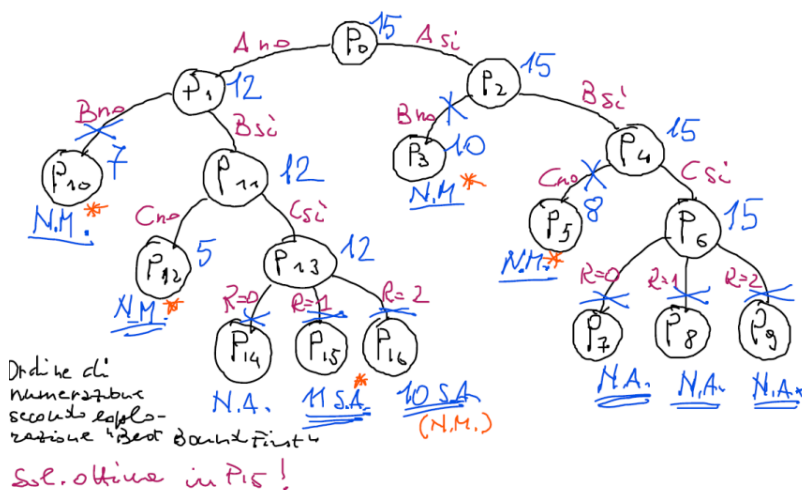
Esempio: albero di branch-and-bound

- A: 3 M\$ (milioni di euro), 3 ruspe
- B: 5 M\$ (milioni di euro), 2 ruspe
- C: 7 M\$ (milioni di euro), 3 ruspe

Descrivendo accuratamente i singoli passaggi (problema di massimo); si consideri che B_i è il bound per il nodo attuale, mentre P_i è il livello di profondità del nodo attuale. Per semplicità in questi passaggi si riporta per i calcoli l'uguaglianza con P_i e non con B_i (che letteralmente sarebbe il costo di bound, es. per P_0 sarebbe $B_0 = 15$ e non l'opposto):

- P_0 contiene tutte le possibili soluzioni; per il bound, dato che valutiamo in modo ottimistico tutti i costi, li sommiamo tutti ($3 + 5 + 7$ ottenendo 15) e non ne sottraggo nessuno, dato che non ce sono.
 - o *Lista = nessuno*
- Decidiamo se fare o no l'appalto A; togliamo dalla lista P_0 e creiamo due sottoproblemi P_1 e P_2 , che sono da analizzare, decidendo rispettivamente che facciamo A ("sì") e che non facciamo A ("no"). Nel nodo P_1 il miglior profitto è $7 + 5 = 12$. Essendo una valutazione ottimistica, non ho tuttora un motivo per buttare via il nodo e lo tengo in lista
 - o *Lista = P_1*
- $P_2 = 3 + 5 + 7 = 15$ (con A, B, C in sospenso). Non posso dire ancora nulla e procedo. Scendendo avremo P_3 e P_4 che sono rispettivamente i nodi per cui decido sicuramente di fare B e decido di non fare sicuramente B. In merito al bound, per B_3 abbiamo che siamo in "A sì, B no", quindi avremo $3 + 7 = 10$. Per il momento, ancora non chiudo nessun nodo e facciamo B_4 che sarà uguale a 15 dato che considero A, B e C. Togliamo inoltre P_2 dalla lista
 - o *Lista = P_1, P_3, P_4*
- Continuo ad esplorare il nodo con miglior bound, in questo caso P_4 e poi divido in P_5 e P_6 togliendo P_4 dalla lista. In P_5 avremo come valore 8 dato che siamo nel caso "C no", quindi $3 + 5$, mentre P_6 sarà dato da "C sì", quindi 15 come visto fino ad ora. Il nodo più promettente è P_6 (perché è problema di massimo)
- A questo punto, arrivo ad un punto di branching che deve farmi decidere quante ruspe affittare: quindi avremo 0, 1, 2 ruspe.
 - o Il primo branch dice "voglio fare tutti e 3 gli appalti, ma affitto 0 ruspe". Il nodo P_7 , quindi, rimane aperto ma non viene inserito in quanto non ammissibile
 - o Il secondo branch dice "voglio fare tutti e 3 gli appalti, ma affitto 1 ruspa". Come prima, non faccio nulla con una sola ruspa e P_8 non è ammissibile
 - o Il terzo branch dice "voglio fare tutti e 3 gli appalti, ma affitto 2 ruspe". Come prima, non faccio nulla con una sola ruspa e P_9 non è ammissibile
- *Lista: P_1, P_3, P_5*

- Ora andiamo verso P_1 in cui abbiamo "A no" e quindi abbiamo $5 + 7 = 12$. Poi P_{10} è dato da "B no", quindi sarà solo 7, mentre P_{11} è dato da "B sì", quindi $5 + 7 = 12$
- Andando verso P_{12} , avendo solo B, sarà 5, mentre P_{13} è dato da "C sì", quindi $5 + 7 = 12$. Tolgo poi P_{11} dalla lista, seleziono P_{13} che è il nodo più promettente e creo i tre sottoalberi.
- A questo punto, arrivo ad un punto di branching che deve farmi decidere quante ruspe affittare: quindi avremo 0,1,2 ruspe.
 - o Il primo branch dice "voglio fare B e C, ma affitto 0 ruspe". Il nodo P_{14} non viene inserito in quanto non ammissibile
 - o Il secondo branch dice "voglio fare B e C, ma affitto 1 ruspa". Il nodo P_{15} , dato che non ha altre decisioni da prendere, è soluzione ammissibile ed è quella che scelgo. In questo caso, prendo il costo trovato finora, quindi 12 e sottraggo le ruspe, quindi $12 - 1 = 11$. Ho quindi trovato una soluzione ammissibile migliore dell'incumbent.
 - Grazie a questa conclusione, posso escludere P_3 con $B_3 = 10$, ma non mi interessa perché ho già 11, similmente P_5 con $B_5 = 8$, similmente P_{10} con $B_{10} = 7$, P_{12} con $B_{12} = 5$, in quanto tutte soluzioni non migliori
 - o Il terzo branch dice "voglio fare B e C, ma affitto 2 ruspe". Qui avremo $12 - 2 = 10$, quindi la scarto in quanto anche qui è soluzione non migliorante



Termino quindi con la lista vuota e con la soluzione *incumbent* = 11

Esempio: regola alternativa per il calcolo dei bound (usando una procedura un po' più complessa):

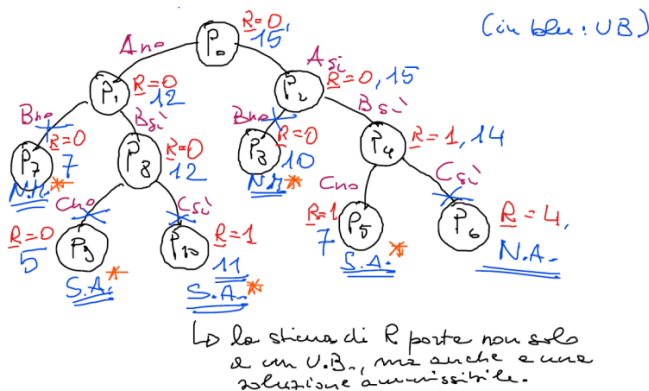
Bound: sommare i profitti di tutti gli appalti possibili e valutare una stima per difetto R delle ruspe necessarie (sulla base degli appalti fissati)

(A: 3 M\$, 3 ruspe B: 5 M\$, 2 ruspe C: 7 M\$, 3 ruspe)

Descrivendo accuratamente i singoli passaggi (problema di massimo); si consideri che B_i è il bound per il nodo attuale, mentre P_i è il nodo attuale. Per semplicità in questi passaggi si riporta P_i e non B_i (che letteralmente sarebbe il costo, es. per P_0 sarebbe $B_0 = 15$)

- Come prima $P_0 = 15$ e nuovamente uso P_1 e P_2 per dire "A no" e "A sì". In questo modo, $B_1 = 12$ e $B_2 = 15$. L'unica cosa che ho deciso da fare in P_2 è A; il bound costa di più perché ora ragiono sulle ruspe da affittare. Tolgo P_0 dalla lista e procedo, lasciando P_1 aperto
- Successivamente, ho P_3 e P_4 ; per il primo, essendo "B no", avrò $7 + 3 = 10$, mentre per il secondo ho fissato A + B, quindi significa che dovrò affittare almeno 5 ruspe (per B), quindi significa che dovrò affittarne almeno una. Significa quindi che il numero di ruspe è $R \geq 1$. In questo caso, quindi, il bound sarà $15 - 1 = 14$ per B_4
- Avanzo con P_4 , nodo migliore da un punto di vista di costi. Quindi, avremo P_5 e P_6 per "C no" e "C sì" rispettivamente. Per P_5 avremo quindi $3 + 5 = 8 - 1 = 7$ (sottraendo la ruspa di prima). Per quanto riguarda P_6 si ha che noi avremmo "A sì, B sì, C sì" per branching, ma questo richiederebbe di affittare almeno 4 ruspe, che non sarebbe possibile. In tutto abbiamo 8 ruspe, ma il problema è dato dal vincolo detto e segniamo il nodo come non ammissibile, chiudendo il nodo

- Il nodo P_7 è dato da "B no", quindi vale 7 essendoci solo C, mentre $B_8 = 12$ per P_8 dato che ho solo "A no", quindi $5 + 7 = 12$.
- Scendo ancora prendendo il nodo migliore, quindi P_8 e creo P_9 e P_{10} . Di nuovo, abbiamo che P_9 è dato da 5, in quanto "A no, B sì, C no", e P_{10} . Quando arriviamo a P_9 siamo arrivati a non avere più altro da fare, pertanto P_9 è dato con $R = 0$. Avendo fissato tutte le variabili, abbiamo che P_9 è soluzione ammissibile, pertanto *incumbent* = 5. Per P_{10} avremo che vale 12 e col bound abbiamo le ruspe = 1 e la soluzione è ammissibile nuovamente, quindi $12 - 1 = 11$
- Successivamente avremo che possiamo chiudere P_7 in quanto soluzione non migliorante, avendo costo 7 essendo "A no, B no". Aggiorniamo la soluzione migliore attualmente, quindi avremo *incumbent* = 7.
- Per lo stesso motivo, posso considerare come soluzione ammissibile P_5 che vale correttamente 7, essendo il costo con $3 + 5 - 1$, in quanto abbiamo selezionato una ruspa in fondo



Termino quindi con la lista vuota e con la soluzione *incumbent* = 7

Tra le due varianti delle regole, non si ha una soluzione migliore: dipende dalla complessità computazionale del problema. Il Branch and Bound è un compromesso computazionale tra complessità ed efficacia.

In generale, per la progettazione:

- le regole di branching permettono di costruire un albero di soluzioni ammissibili secondo il principio divide et impera, partendo da un problema e partizionandolo in sottoproblemi più piccoli e semplici. In base al problema, si adegua la strategia di branching (quanti figli generare, come effettuare il branching, etc.)
- il calcolo del bound serve per calcolare una stima ottimistica della migliore soluzione che:
 - o per problemi di minimo è un lower bound
 - o per problemi di massimo è un upper bound

Questo viene calcolato ad ogni nodo e il numero di nodi da valutare potrebbe essere molto elevato (esponenziale nei casi peggiori). Il metodo deve essere quindi rapido nel calcolo e il più stringente possibile (quindi, quando ha una soluzione ammissibile, devo potare un maggior numero di nodi).
- Regole di fathoming: Andiamo a dichiarare esplorati i nodi che sicuramente non contengono una soluzione ottima. I nodi non sviluppati e relativi sottoalberi sono definiti *potato*. In particolare, se chiude un nodo se si verifica almeno una tra queste condizioni
 - o [N.M.] Assenza di soluzione migliorante, quindi la valutazione B_i non migliora una soluzione ammissibile nota
 - o [S.A.] Valutazione ottimistica è anche di soluzione ammissibile, quindi non ha senso continuare a migliorare sapendo che non si possono trovare soluzioni ammissibili migliori a quella calcolata

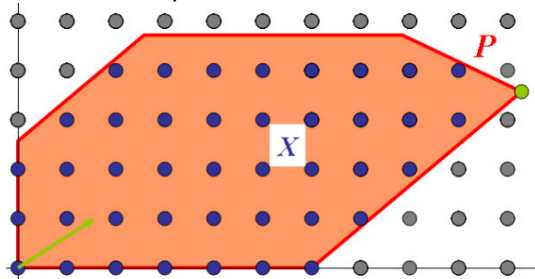
- [N.A.] Sottoproblema non ammissibile ($E_i = \emptyset$), dato che avremo l'insieme delle condizioni che determinano il problema più le condizioni che hanno generato i progenitori; potremmo avere conflitti.
- Strategie di esplorazione:
 - Depth First: Prendo il nodo più profondo e quindi tendo a chiudere molti nodi per ammissibilità e rimangono pochi nodi contemporaneamente aperti. Per contro, presenta il rischio di esplorare completamente sottoalberi con soluzioni scadenti
 - Best Bound First: Scelgo il nodo con bound migliore (limita il numero di nodi visitati, ma l'esplorazione si ferma a livelli poco profondi, compromettendo la sicurezza dei bound scelti)
 - Regole miste, usando diversi criteri per evitarne gli svantaggi
- Valutazione di soluzioni ammissibili: è necessario disporre di soluzioni ammissibili buone, ma si deve valutare il compromesso tra la qualità
- Criteri di arresto: si ferma quando tutti i nodi sono dichiarati *fathomed* (metodo esatto)

Dato un PLI del tipo:

$$\begin{aligned} z_I &= \max c^T x \\ Ax &\leq b \\ x &\geq 0 \\ x_i &\in \mathbb{Z}, \quad i \in I. \end{aligned} \quad (1)$$

dove $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ e $I \subseteq \{1, \dots, n\}$ è l'insieme degli indici delle *variabili intere*. Le variabili x_i , $i \notin I$ sono invece dette *variabili continue*. Se il problema ha sia variabili intere che variabili continue, allora è detto un *problema di programmazione lineare intera mista*, mentre se tutte le variabili sono intere, il problema è detto di *programmazione lineare intera pura*.

Togliendo il punto verde dal PLI rappresentato sul piano, significa risolvere il problema ignorando il vincolo di interezza e permette di ottenere un bound:



$$\begin{aligned} z_L &= \max c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

$$z_L \geq z_I \quad z_L \text{ è un UB!}$$

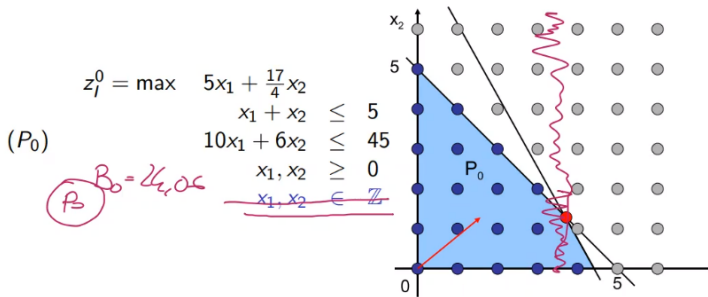
$$\begin{aligned} z_I &= \max c^T x \\ Ax &\leq b \\ x &\geq 0 \\ x_i &\in \mathbb{Z}, \quad i \in I. \end{aligned}$$

Quanto a lato (seconda da sx) rappresenta il rilassamento lineare del problema, definito appunto ignorando il vincolo di interezza sul problema con variabili continue (terza da sx), che è un problema di programmazione lineare intera. In questo modo, la soluzione per rilassamento lineare rappresenta un upper bound della soluzione PLI di partenza.

Vediamo ora come è possibile definire le varie componenti di un metodo di Branch and Bound per problemi di programmazione lineare intera.

Consideriamo il seguente problema, con relativa regione ammissibile e rilassamento continuo (questo si applica quando si usa il metodo del simplesso, altrimenti normalmente si usa il rilassamento lineare, come anche indicato dalle dispense):

Problema P_0



Rilassamento lineare: $x_1 = 3.75, x_2 = 1.75$, con valore $z_L^0 = 24.06$

$$x_1 \leq 3 \quad x_2 \geq 4$$

$$x_1 \in [3, 3.75] \quad x_2 \in [3.25, 4]$$

Risolvendo il rilassamento lineare di (P_0) , otteniamo la soluzione ottima $x_1 = 3.75, x_2 = 1.75$, con valore $z_L^0 = 24.06$.

Dunque abbiamo ottenuto un upper bound per il valore ottimo z_I^0 di (P_0) , ovvero $z_I^0 \leq 24.06$. Ora, poiché in una soluzione ottima di (P_0) x_1 avrà valore intero, allora la soluzione ottima soddisferà $x_1 \leq 3$ oppure $x_1 \geq 4$. Dunque la soluzione ottima di (P_0) sarà la soluzione migliore tra le due soluzioni dei problemi (P_1) e (P_2) così definiti:

Consideriamo inoltre l'aggiunta dei vincoli di branching indicati con sottolineatura:

$$z_I^1 = \max \quad 5x_1 + \frac{17}{4}x_2$$

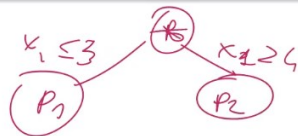
$$(P_1) \quad \begin{aligned} x_1 + x_2 &\leq 5 \\ 10x_1 + 6x_2 &\leq 45 \\ \underline{x_1} &\leq 3 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

$$z_I^2 = \max \quad 5x_1 + \frac{17}{4}x_2$$

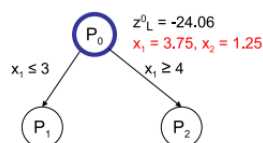
$$(P_2) \quad \begin{aligned} x_1 + x_2 &\leq 5 \\ 10x_1 + 6x_2 &\leq 45 \\ \underline{x_1} &\geq 4 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

Si consideri che non perdo soluzioni, ma tolgo l'unica soluzione non ammissibile:

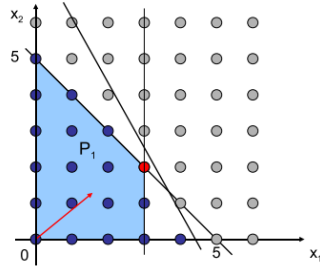
- Non perdo soluzioni intere: $E_1 \cup E_2 = E_0, z_I = \max\{z_I^1, z_I^2\}$
- z_L^0 esclusa!



Diremo che abbiamo fatto *branching sulla variabile x_1* . Si noti che in questo modo la soluzione $(3.75, 1.25)$ non appartiene al rilassamento lineare di (P_1) o (P_2) . Possiamo rappresentare graficamente i sotto-problemi e i rispettivi bound mediante un albero, detto albero di Branch-and-Bound.

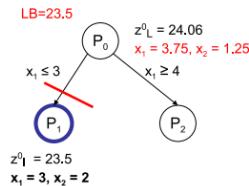


Considerando inoltre il problema, avremo che per il rilassamento continuo, otteniamo una soluzione ammissibile. L'unica foglia non potato P_2 è l'unico problema attivo, dopo aver trovato la soluzione ottima in P_1 . Aggiorniamo $incumbent = 23.5$.



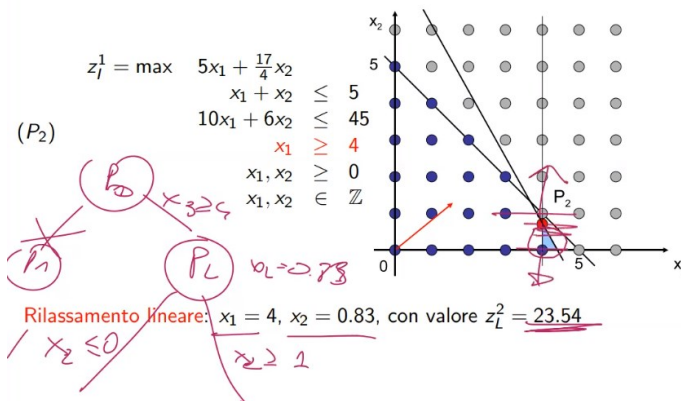
La soluzione ottima del rilassamento lineare di (P_1) è $x_1 = 3, x_2 = 2$, con valore $z_L^1 = 23.5$. Si noti che tale soluzione è intera, e dunque, poiché $z_I^1 \leq z_L^1$, in tal caso $(3, 2)$ è anche la soluzione ottima intera. Dunque non occorre fare ulteriore branching per il nodo (P_1) , che può dunque essere potato. Diciamo che (P_1) è *potato per ammissibilità del bound (S.A.)*. Si noti inoltre che la soluzione ottima di (P_0) avrà valore $z_I^0 \geq z_I^1 = 23.5$. $(3, 2)$ è la *soluzione intera corrente*, ovvero la miglior soluzione intera trovata fino a questo punto, il cui valore è $\bar{z} = 23.5$.

L'albero di Branch-and-Bound è ora il seguente.



Successivamente, otteniamo una soluzione frazionaria, che non ci va bene e rieseguiamo il branching. Ci spostiamo quindi nello spazio, risolvendo all'ottimo P_2 con due successivi sottoproblemi.

Problema P_2



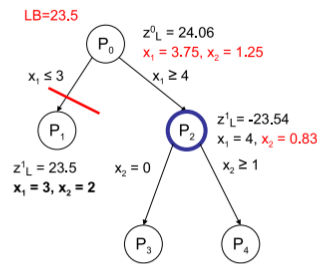
La soluzione ottima del rilassamento lineare di (P_2) è $x_1 = 4, x_2 = 0.83$, con valore $z_L^2 = 23.54$. Dunque $z_I^2 \leq 23.54$, e 23.54 è un upper-bound al valore ottimo di (P_2) . Si noti che $\bar{z} = 23.5 < 23.54$, dunque (P_1) potrebbe avere soluzione migliore della soluzione intera corrente. Poiché la componente x_2 della soluzione ottima di (P_2) ha valore 0.83 , facciamo branching su x_2 , ottenendo i seguenti due sotto-problemi (P_3) e (P_4) di (P_2) .

L'albero di Branch-and-Bound è ora il seguente.

Branch da P_2 su variabile frazionaria $x_2 = 0.83$

$$\begin{aligned} z_1^1 = \max \quad & 5x_1 + \frac{17}{4}x_2 \\ & x_1 + x_2 \leq 5 \\ & 10x_1 + 6x_2 \leq 45 \\ & x_1 \geq 4 \\ & x_2 \leq 0 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \\ & (P_3) \end{aligned} \quad \begin{aligned} z_2^1 = \max \quad & 5x_1 + \frac{17}{4}x_2 \\ & x_1 + x_2 \leq 5 \\ & 10x_1 + 6x_2 \leq 45 \\ & x_1 \geq 4 \\ & x_2 \geq 1 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \\ & (P_4) \end{aligned}$$

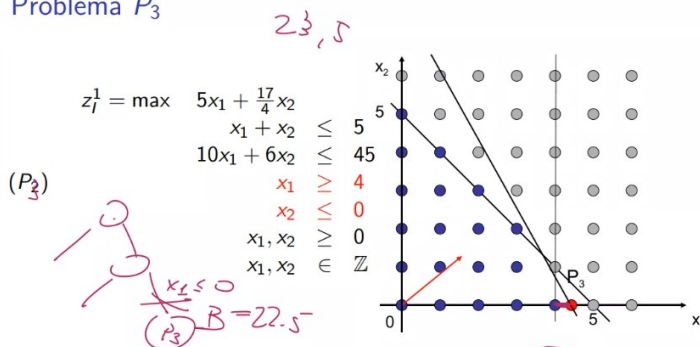
$$E_3 \cup E_4 = E_2$$



I nodi attivi sono (P_3) e (P_4) . Risolviamo il rilassamento lineare di (P_3) (rappresentato in figura),

Chiudo successivamente il nodo essendo non migliorante:

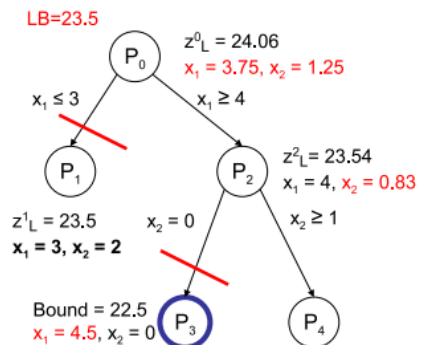
Problema P_3



Rilassamento lineare: $x_1 = 4.5, x_2 = 0$, con valore $z_L^3 = 22.5$

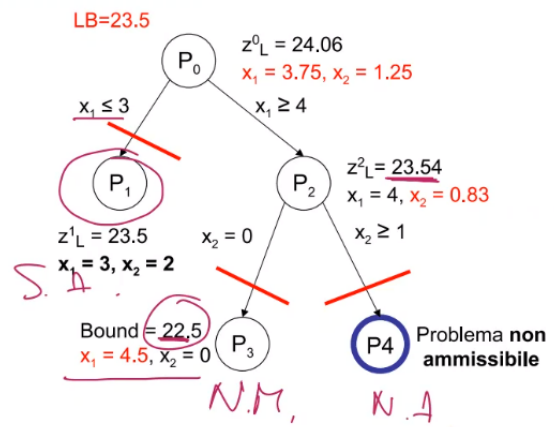
$z_L^3 \leq \bar{z}$: nodo potato per **N.M.**

ottenendo soluzione ottima $x_1 = 4.5, x_2 = 0$, con valore $z_L^3 = 22.5$. Dunque la soluzione ottima intera di (P_3) avrà valore $z_1^3 \leq 22.5$, ma poiché abbiamo già determinato una soluzione ottima intera con valore 23.5, è inutile esplorare ulteriormente la regione ammissibile di (P_3) poiché sappiamo che non vi sarà nessuna soluzione intera di valore maggiore di $22.5 < 23.5$. Possiamo dunque *potare il nodo (P_3) perché non migliorante (N.M.)*. L'albero di Branch-and-Bound corrente, rappresentato nella figura successiva, contiene un unico problema attivo, ovvero (P_4) .



A questo punto, avremo che nessuna soluzione continua/intera soddisfa il problema, chiudendo il problema in quanto non ammissibile. In particolare, potiamo il nodo P_4 per inammissibilità e dunque, avremo che la soluzione ottima è ottenuta in $(3,2)$.

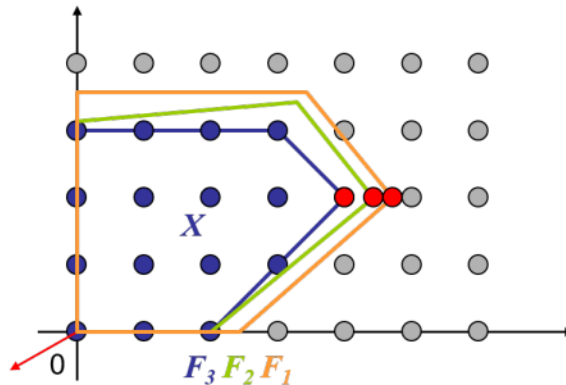
Il branching definito per rilassamento continuo su componenti frazionarie viene detto *dicotomico*, che permette volendo di dare priorità ad un valore piuttosto che un altro in branching (ulteriore vincolo di bound).



Laboratorio 7 – Branch and Bound con AMPL

Noi sappiamo valutare bene un bound per un problema PLI trascurando i vincoli di interezza.

Per esempio, per trovare nel seguente grafico una soluzione, creiamo un problema di PLI, “in scatolando” i punti blu all’interno di un politopo (con vari colori, indicando un “bounding” di volta in volta più stretto/tight). Più vicina la formulazione è al politopo blu, meglio è il bound. Ogni bound viene calcolato con il rilassamento continuo.



- F_2 è migliore di F_1 : fornisce bound più stringenti (più vicini all’ottimo): UB più bassi (per problemi di max) o LB più alti (per problemi di min)
- F_3 è la formulazione ideale: permette di risolvere il problema al nodo radice (senza branching). Come dice il nome, non è sempre possibile scriverla.

Consideriamo che, in problemi come il cammino minimo, ipotizzando i vertici come interi, non serve risolverli con il rilassamento continuo. In alcuni contesti, se particolarmente complesso per presenza di tanti nodi, si usa la tattica del *diving* per esplorarli tutti. Normalmente, basta il B&B.

Ora, creiamo un file *kp.mod* (*kp* per *knapsack*, essendo il problema dello zaino):

Esempio “Zaino binario”

Risolvere con il metodo del Branch-and-bound:

$$\begin{aligned} \max \quad & 22.5x_1 + 30.0x_2 + 40.3x_3 + 11.1x_4 + 14.7x_5 + 9.1x_6 \\ \text{s.t.} \quad & 10x_1 + 15x_2 + 21x_3 + 6x_4 + 8x_5 + 5x_6 \leq 47 \\ & x_1, \dots, x_6 \in \{0, 1\} \end{aligned}$$

```
var x1 <= 1 >= 0;
var x2 <= 1 >= 0;
var x3 <= 1 >= 0;
var x4 <= 1 >= 0;
var x5 <= 1 >= 0;
var x6 <= 1 >= 0;
```

```
maximize f: 22.5*x1 + 30.0*x2 + 40.3*x3 + 11.1*x4 + 14.7*x5 + 9.1*x6;
s.t. v: 10*x1 + 15*x2 + 21*x3 + 6*x4 + 8*x5 + 5*x6 <= 47;
```

Ora, creiamo un file *kp.run* nel quale mostriamo tutte le variabili. Si noti che esiste una sola variabile frazionaria (x_4), mentre le altre rimangono intere. La risoluzione iniziale conduce ad avere 94.65.

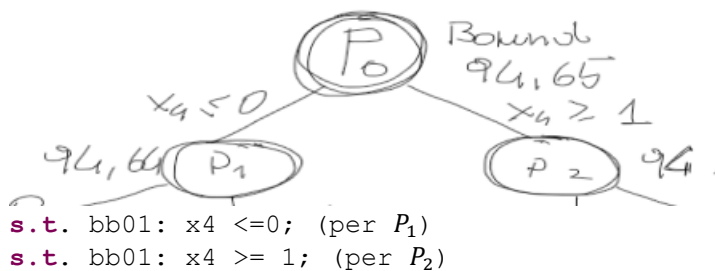
```
reset;
model kp.mod;
solve;
display f;
display x1, x2, x3, x4, x5, x6;
```

Implementando il primo branching secondo l’ordine BBF (Best Bound First), si nota che si aggiunge nel file *.mod* seguendo l’implementazione grafica. La valutazione segue entrambe le strade (P_1 e P_2 , dato che cerchiamo ogni volta il bound migliore. In questo caso, eseguiamo il branching rispetto ad x_4 .

Durante tutto l'esercizio, si assume che le uguaglianze siano intese come disuguaglianze nell'albero, cioè che andando a sx si abbia $\leq$, mentre andando a dx si abbia $\geq$.

Note importanti di ragionamento

- Si considera lo sviluppo su un altro branch nel momento in cui ho un nodo non ammissibile e l'altro invece sì, quindi torno "ad un altro nodo aperto".
- La scelta di x_i per il branching non è casuale; si prendono le variabili frazionarie (quindi, quelle non intere ma decimali) ottenute per quel nodo e lo segno subito
- In caso di scelta tra variabili frazionarie, prendo quella più vicina all'intero
- *A meno che non diversamente specificato*, meglio arrotondare per eccesso tale da avere meno decimali nei calcoli successivi

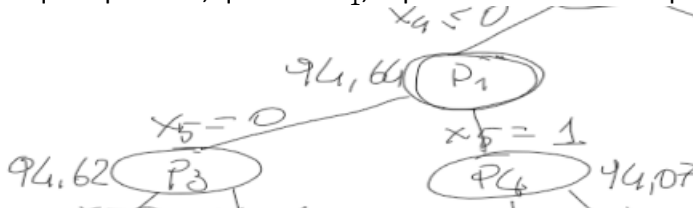


Risolvendo, si trovano i bound, come si vede con l'inserimento del vincolo $x_4 \geq 1$. In questo caso, si va sempre ad arrotondare rispetto al decimale (quindi, la soluzione qui sotto si arrotonda per 94.30; sulle slide, i decimali sono tutti shiftati avanti di un decimale, quindi qui è 94.31):

```
ampl: include kp.run;
MINOS 5.51: optimal solution found.
3 iterations, objective 94.3047619
f = 94.3048
```

```
x1 = 1
x2 = 1
x3 = 0.761905
x4 = 1
x5 = 0
x6 = 0
```

Lasciamo i nodi aperti, proseguendo per il secondo ramo di branching, andando a prendere il valore più alto tra quelli presenti, quindi da P_1 , esplorando entrambe le possibilità:



Prima possibilità (P_3) $\rightarrow$ Risultato: 94.0667 = 94.07

s.t. bb01: $x_4 \leq 0$;

s.t. bb02: $x_5 \leq 0$;

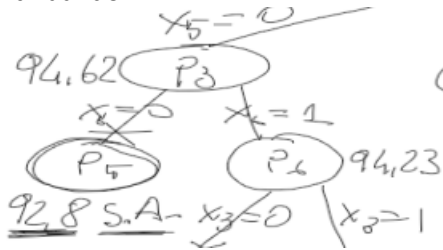
Seconda possibilità (P_4) $\rightarrow$ Risultato: 94.62

s.t. bb01: $x_4 \leq 0$;

s.t. bb02: $x_5 \geq 1$;

Per entrambi i risultati, abbiamo dei coefficienti decimali, dunque frazionari e che non contribuiscono correttamente alla soluzione.

Continuando:



Prima possibilità (P_5) → Risultato: 92.8

s.t. bb01: $x_4 \leq 0$;
s.t. bb02: $x_5 \leq 0$;
s.t. bb03: $x_6 \leq 0$;

In questo caso, mi fermo, dato che ho trovato una soluzione ammissibile. Infatti, tutti i coefficienti x_i sono interi e non frazionari:

```
ampl: include kp.run;
MINOS 5.51: optimal solution found.
3 iterations, objective 92.8
f = 92.8
```

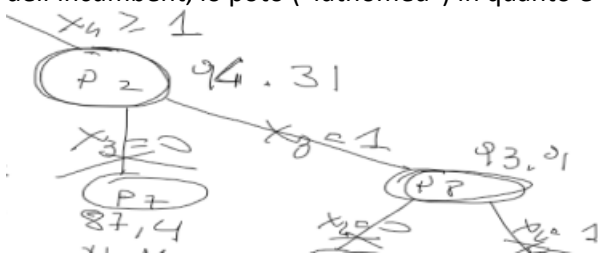
```
x1 = 1
x2 = 1
x3 = 1
x4 = 0
x5 = 0
x6 = 0
```

Aggiorniamo *incumbent*, che diventa pari a 92.8.

Seconda possibilità (P_6) → Risultato: 94.2238 (teniamo aperto in quanto i coefficienti sono frazionari)

s.t. bb01: $x_4 \leq 0$;
s.t. bb02: $x_5 \leq 0$;
s.t. bb03: $x_6 \geq 1$;

Andiamo all'altro ramo dalla parte di P_2 , stavolta eseguendo il branching rispetto ad x_3 . Scelgo P_8 in quanto è la soluzione più grande (migliore) e non ha x_i frazionari. Direttamente, essendo 87.4 più piccolo dell'*incumbent*, lo poto ("fathomed") in quanto è non migliorante.



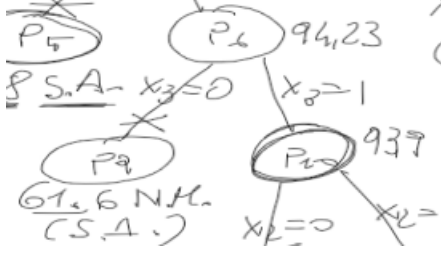
Per P_7 avremo 87.4:

s.t. bb01: $x_4 \geq 1$;
s.t. bb02: $x_3 \leq 0$;

Per P_8 avremo 93.9:

s.t. bb01: $x_4 \geq 1$;
s.t. bb02: $x_3 \geq 1$;

Proseguendo nel branching rispetto a P_6 :



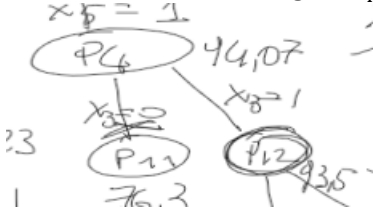
Per P_9 avremo 61.6 (soluzione ammissibile ma non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \leq 0$;
- s.t. bb03: $x_6 \geq 1$;
- s.t. bb04: $x_3 \leq 0$;

Per P_{10} avremo 93.9 (nodo da esaminare successivamente, lo teniamo aperto):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \leq 0$;
- s.t. bb03: $x_6 \geq 1$;
- s.t. bb04: $x_3 \geq 1$;

Continuiamo il branching da P_4 :



Per P_{11} avremo 76.3 (soluzione ammissibile ma non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \geq 1$;
- s.t. bb03: $x_3 \leq 0$;

Per P_{12} avremo 93.5 (nodo da esaminare successivamente, lo teniamo aperto):

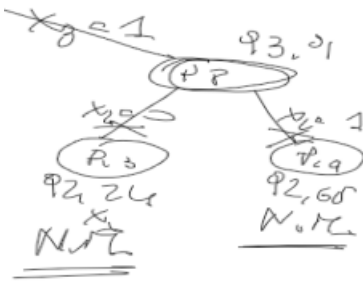
- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \geq 1$;
- s.t. bb03: $x_3 \geq 1$;

Supponiamo di voler fermare nelle iterazioni; non abbiamo trovato la soluzione ottima, tuttavia abbiamo trovato una soluzione ammissibile (*incumbent* = 92.8) e non guadagneremo mai più della soglia iniziale (94.65). Sviluppando l'albero, abbiamo operato tante divisioni in sottoproblemi. La migliore stima del nodo rimasto aperto, fino ad ora, avremo 93.9; qualsiasi strada che abbiamo, guadagneremo meno.

Al massimo, convergeremo 93.9; *man mano che si sviluppa l'albero, la stima all'ottimo diventa più precisa*. La f.o. è compresa tra 92.8 e 93.9, perdendo al massimo 1.1 milioni di euro.

In questo caso, essendo un problema di massimo, sviluppando il problema, il bound diventa sempre più piccolo.

Sviluppiamo il branching da P_8 :



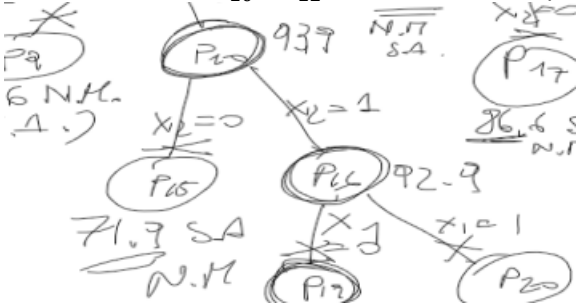
Per P_{11} avremo 92.24 (soluzione non ammissibile dato che non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \geq 1$;
- s.t. bb02: $x_3 \geq 1$;
- s.t. bb03: $x_2 \leq 0$;

Per P_{11} avremo 92.65 (soluzione non ammissibile dato che non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \geq 1$;
- s.t. bb02: $x_3 \geq 1$;
- s.t. bb03: $x_2 \geq 1$;

Continuiamo con P_{10} e P_{12} che sono rimasti aperti (inserendo sempre i vincoli dei predecessori):



Per P_{15} avremo 71.9 (soluzione non ammissibile dato che non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

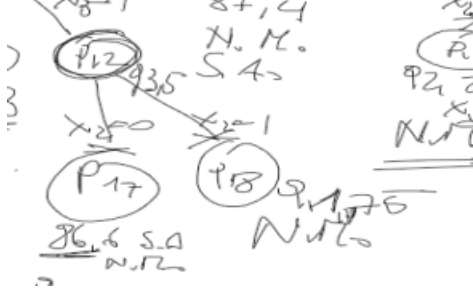
- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \leq 0$;
- s.t. bb03: $x_6 \geq 1$;
- s.t. bb04: $x_3 \leq 1$;
- s.t. bb05: $x_2 \leq 0$;

Per P_{16} avremo 92.9 (soluzione non ammissibile dato che non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \leq 0$;
- s.t. bb03: $x_6 \geq 1$;
- s.t. bb04: $x_3 \geq 1$;
- s.t. bb05: $x_2 \geq 1$;

Teniamo quest'ultima soluzione essendo migliorante in ottica branching.

Vado ora verso P_{12} creando P_{17} e P_{18} .



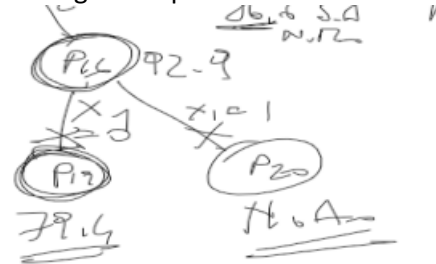
Per P_{17} si ottiene 86.6 (soluzione ammissibile dato che tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \geq 1$;
- s.t. bb03: $x_3 \leq 1$;
- s.t. bb04: $x_2 \leq 0$;

Per P_{18} si ottiene 91.75 (soluzione non ammissibile dato che non tutte le x_i sono intere e sarebbe non migliorante e lo poto "fathomed"):

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \geq 1$;
- s.t. bb03: $x_3 \geq 1$;
- s.t. bb04: $x_2 \geq 1$;

Proseguendo poi con l'ultimo ramo:



Per P_{20} si ha una soluzione del tipo *impossible deduced bounds* (dato che ci sarebbe errata corrispondenza tra lower bound ed upper bound la soluzione sarebbe non ammissibile per P_{20}).

- s.t. bb01: $x_4 \leq 0$;
- s.t. bb02: $x_5 \geq 0$;
- s.t. bb03: $x_6 \geq 1$;
- s.t. bb04: $x_3 \geq 1$;
- s.t. bb05: $x_2 \geq 1$;
- s.t. bb06: $x_1 \geq 1$;

In questa esecuzione, troverei Per P_{19} invece si ha una soluzione non migliorante.

Il B&B possiede sempre una valutazione ottimistica (upper bound, valore oltre il quale non andare per problemi di massimo, oppure lower bound, valore oltre il quale non scendere per problemi di minimo); tramite questi, capiamo sempre se la soluzione può essere ammissibile.

- Branching: binario
- Bound: rilassamento continuo (usare AMPL!)
- Fathoming: standard
- Esplorazione: a piacere (Best Bound First)
- Valutazione soluzioni ammissibili: nessuno (da rilassamento intero)
- Stop: lista nodi aperti vuota

Risolviamo un altro problema:

Esempio "PLI generico"

Risolvere con il metodo del Branch-and-bound:

$$\begin{aligned} \max \quad & 3x_1 - 8x_2 + 3x_3 - 8x_4 + 13x_5 \\ \text{s.t.} \quad & -2x_1 + 7x_2 + 4x_3 + 1.5x_4 + 9x_5 \leq 16 \\ & -6x_1 + 5x_2 + 5x_3 + 7.2x_4 - 3x_5 \geq 7 \\ & x_1, \dots, x_5 \in \mathbb{Z}^+ \end{aligned}$$

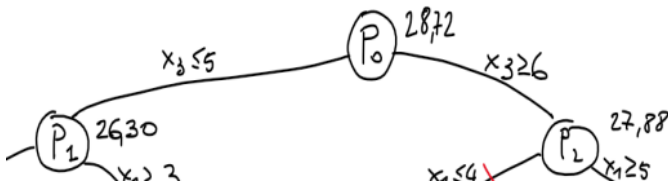
Creiamo il corrispondente modello, molto semplice, nel file *pli.mod*:

```
var x1 >=0;
var x2 >=0;
var x3 >=0;
var x4 >=0;
var x5 >=0;
var x6 >=0;
maximize f: 3*x1 - 8*x2 + 3*x3 - 8*x4 + 13*x5;
s.t. v1: -2*x1 + 7*x2 + 4*x3 + 1.5*x4 + 9*x5 <= 16;
s.t. v2: -6*x1 + 5*x2 + 5*x3 + 7.2*x4 - 3*x5 >= 7;
```

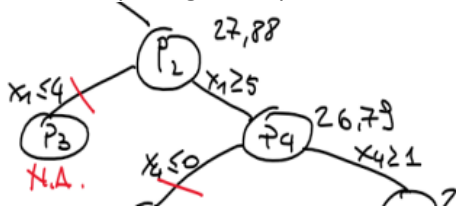
Poi, il corrispondente file *pli.run*:

```
reset;
model pli.mod;
solve;
display f;
display x1, x2, x3, x4, x5, x6;
```

Adottiamo quindi la stessa logica esplorata nel dettaglio di prima, sempre con ordine BBF. Ci sono due variabili frazionarie, ma non abbiamo un criterio preciso di scelta. Prendiamo, quindi, la variabile frazionaria più vicina al corrispondente intero e useremo questo come vincolo. Eseguiamo quindi branching su x_3 , ponendo da una parte ≤ 5 e dall'altra $x_3 \geq 6$:



Eseguiamo il branching sui due nodi precedenti. Per BBF scelgo chiaramente P_2 essendo problema di massimo e proseguo su quello:



Inseriamo i vincoli di branching e nel caso di P_3 avremo che non è ammissibile.

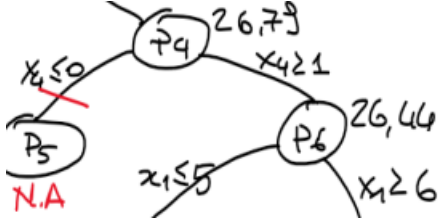
```
s.t. bb01: x3 >= 6;
s.t. bb02: x1 <= 4;
```

```
ampl: include pli.run;
presolve, constraint v2:
    all variables eliminated, but lower bound = 1 > 0
Infeasible constraints determined by presolve.
```

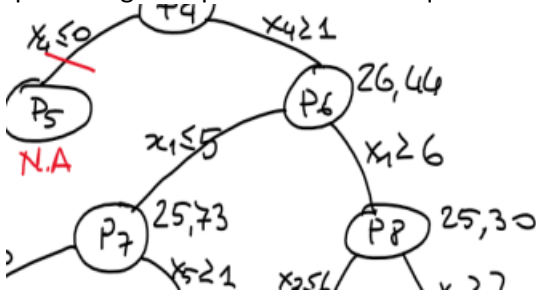
Correttamente, l'altro risulta 26.79:

s.t. bb01: $x_3 \geq 6$;
s.t. bb02: $x_1 \geq 5$;

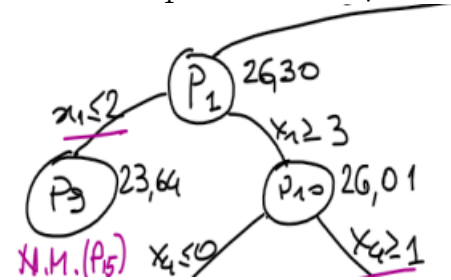
Proseguendo con P_4 , avremo una soluzione non ammissibile e l'altra corretta, che teniamo aperta.



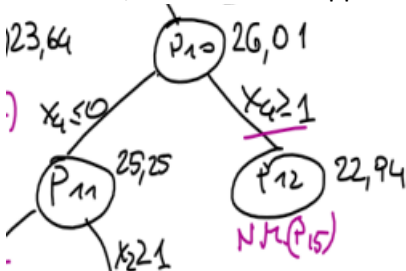
Ancora, avremo P_7 e P_8 . Anche qui, avremo due soluzioni aperte. Tra le due, sceglieremo comunque P_7 ma qui rimangono aperte entrambe e quindi torniamo verso P_1 , che era rimasto aperto.



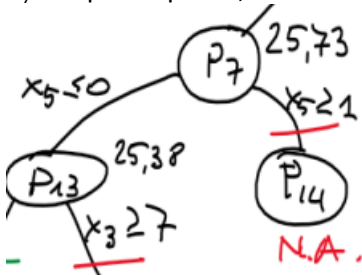
Torniamo a P_1 che è rimasto aperto.



Tra le due, andremo a sviluppare su P_{10} essendo il maggiore e proseguiamo con il branching:



In questo punto, troviamo entrambi i nodi aperti, quindi fermo lo sviluppo in questa direzione e vado verso P_7 . In questo punto, trovo che P_{13} è una soluzione possibile, mentre P_{14} sarà considerata non ammissibile.



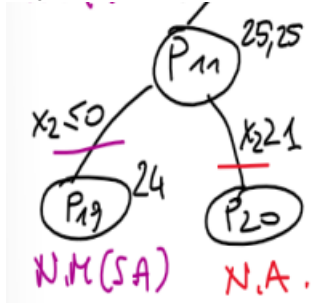
Nel ramo di P_{13} , arrivo verso P_{15} , che, come si vede, è soluzione ammissibile:

```
ampl: include pli.run;
MINOS 5.51: optimal solution found.
1 iterations, objective 25
f = 25
```

```
x1 = 5
x2 = 0
x3 = 6
x4 = 1
x5 = 0
x6 = 0
```

Contrariamente a ciò, P_{16} non è ammissibile. Vari nodi esplorati, come P_3 e P_{12} sono considerati non miglioranti, in quanto minori a 25, attuale valore peraltro di *incumbent*.

Concludendo, nell'ultimo ramo di branch, dove troveremo P_{19} soluzione ammissibile ma non migliorante, P_{20} non ammissibile semplicemente.



Troviamo l'incumbent dopo 15 nodi; provare a risolvere lo stesso problema con arrotondamenti

Si consideri il problema dell'Esempio "PLI generico". Quale sarebbe lo sviluppo dell'albero di B&B con le seguenti varianti:

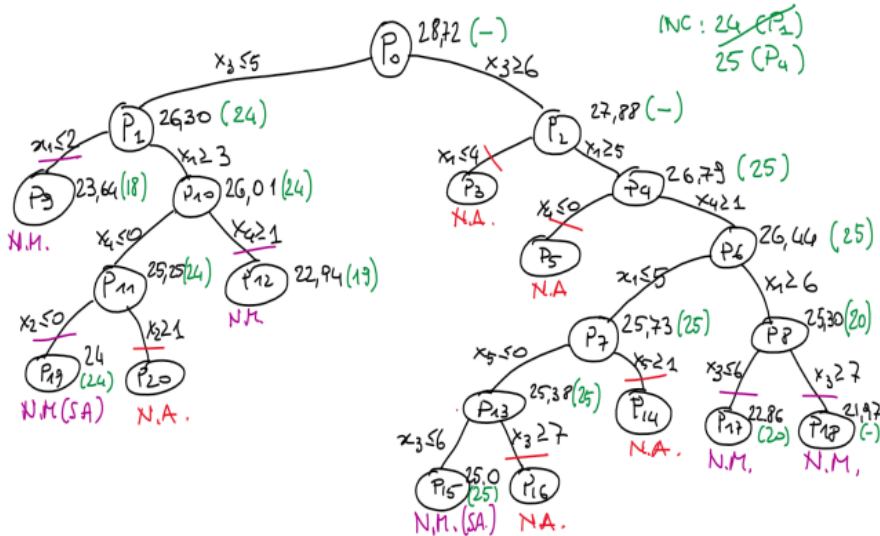
- variante A: provare a generare, ad ogni nodo, una soluzione ammissibile approssimando per difetto la soluzione frazionaria ottenuta con il rilassamento continuo
- variante B: migliorare il bound osservando che tutti i coefficienti e tutte le variabili della funzione obiettivo, **nello specifico problema in esame**, sono interi, quindi il valore della funzione obiettivo è intero.

Andando a provare fisicamente la versione A delle varianti (soluzione ammissibile approssimando per difetto la soluzione frazionaria) di cui si riporta l'albero completo sotto, alcune osservazioni:

- Tutto ciò che viene segnato non ammissibile è tale perché usando AMPL risulta "infeasible"
- La prima soluzione ammissibile trovata è 24, poi eseguendo il B&B a destra si trova 25
- Rispetto all'albero precedente, l'approssimazione permette di chiudere prima l'albero, ma non di risparmiare il numero di nodi valutati

Esempio "PLI generico - A": soluzione

"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)

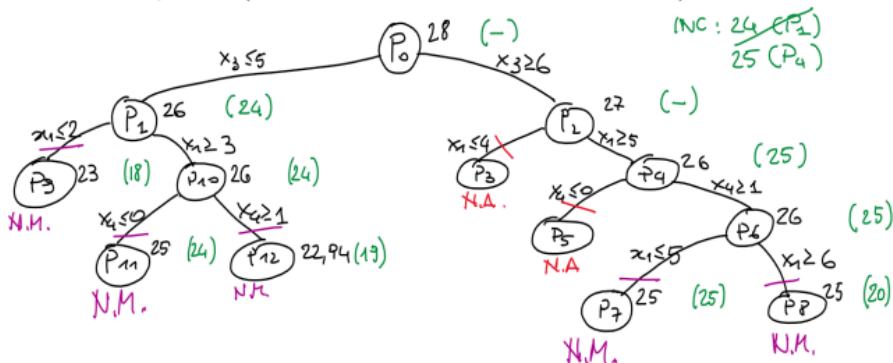


Andando a provare fisicamente la versione B delle varianti (migliorare il bound mettendo ad intero approssimando per difetto [quindi utilizza anche la variante A, ecco perché A+B nell'immagine]) la valutazione della f.o. in quanto *nello specifico problema* variabili ed f.o. sono interi), di cui si riporta l'albero completo sotto, alcune osservazioni:

- Approssimare all'intero inferiore individua meno nodi e permette di chiudere subito altri nodi ancora, altrimenti reiterati fino a successivi bound (quindi, si affina meglio la valutazione). Questo funziona, essendo problema di massimo.

Esempio "PLI generico - A e B": soluzione

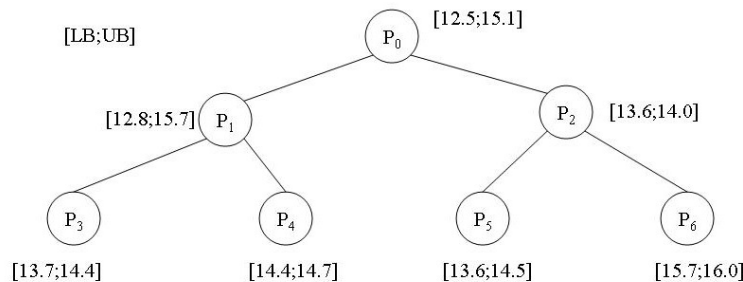
"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)



Esercizi sul Branch and Bound

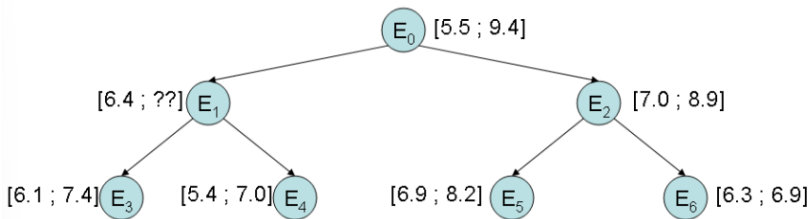
Esercizio 1 Si consideri il seguente albero di sviluppo del Branch and Bound per un problema di ottimizzazione combinatoria con funzione obiettivo di minimo:

Esercizio



1. Come si può capire che si tratta di un problema di minimo?
 2. È possibile chiudere dei nodi? Se sì, quali?
 3. In quale intervallo è sicuramente compreso il valore della funzione obiettivo?
 4. Quale nodo sarà sviluppato per primo da una strategia *best first*?
 5. Si supponga che lo sviluppo di cui al punto precedente porti a due nodi figli, di cui uno è relativo ad un insieme di soluzioni vuoto. Si dia un esempio di valori di lower e upper bound relativi al secondo nodo che consentano di riconoscere subito la soluzione ottima del problema.
1. Si capisce che è un problema di minimo perché i valori contrassegnati come LB sono crescenti di padre in figlio nell'albero e pertanto possono essere associati a valutazioni ottimistiche di problemi di minimo via via più vincolati. I valori contrassegnati come UB non sono decrescenti di padre in figlio e non possono essere associati a valutazioni ottimistiche di problemi di massimo via via più vincolati. I valori UB sono quindi le valutazioni della funzione obiettivo di minimo in corrispondenza di soluzioni ammissibili.
 2. La migliore soluzione ammissibile vale 14.0 (vedi nodo P_2). Quindi è possibile chiudere i nodi P_4 e P_6 perché non miglioranti.
 3. L'ottimo della funzione obiettivo è compreso tra 13.6 (il miglior lower bound - nodo P_5) e 14.0 (migliore soluzione disponibile).
 4. Il nodo che sarà sviluppato per primo in una strategia *best bound first* è quello che ha la valutazione più promettente (LB più basso) tra quelli che rimangono aperti, cioè il nodo P_5 .
 5. Nel caso ipotizzato, rimangono aperti il nodo P_3 con $(LB, UB) = (13.7, 14.4)$ e un nodo P_7 con $(LB, UB) = (lb, ub)$. Basta quindi che sia $lb = ub$ (lb corrisponde a una soluzione ammissibile) per poter chiudere il nodo in esame P_7 e che $lb \leq 13.7$, per poter chiudere P_3 . Deve inoltre essere $lb \geq 13.6$, per compatibilità con il lower bound del nodo padre P_5 . Ad esempio $(LB, UB) = (13.65, 13.65)$, o $(LB, UB) = (13.7, 13.7)$, o $(LB, UB) = (13.6, 13.6)$ etc.

Esercizio 2 Si consideri il seguente albero di B&B relativo ad un problema di massimo. Ad ogni nodo sono stati valutati sia un upper bound (UB) sia una soluzione ammissibile (SA), come riportato accanto ad ogni nodo, nel formato [SA;UB].



- Min o Max? → Problema di massimo → [SA; UB]
 - Se fosse problema minimo, di padre in figlio il LB cresce; tuttavia, andando da E_1 ad E_3 il LB passa da 6.4 a 6.1 e non cresce; sicuramente non è problema di massimo
 - Se fosse problema massimo, di padre in figlio l'UB diminuisce; andando in basso, di fatto si ha questa condizione
 - Per i punti ??, andremo ad inserire un valore compreso tra 9.4 (se vogliamo che sia problema di max, non deve essere superiore al nodo del padre) e inferiore al maggiore dei figli), quindi $7.4 \rightarrow 7.4 \leq ?? \leq 9.4$
- Intervallo ottimo?
 - Ci serve un incumbent, che viene cercato tra tutti i possibili LB (cerco il più grande tra i LB essendo di massimo); l'incumbent è 7.0
 - Per gli UB, cerco tra i soli nodi aperti, quindi E_3, E_4, E_5, E_6
 - Prendo il valore più alto tra i nodi aperti, quindi 8.2
 - Quindi [7.0; 8.2]
- Nodi da chiudere?
 - Controllo se l'UB sia migliore della soluzione incumbent in mano (quindi, 7), quindi $\geq$ e chiudo tutti i nodi che promettono di meno di S.A.
 - Posso chiudere E_6 , in quanto 6.9 non è migliore di 7
 - Posso chiudere E_4 , in quanto 7 non è migliore di 7
- Qual è il nodo esplorato con una strategia best bound first?
 - Si sceglie il nodo con il miglior UB, quindi il nodo E_5
- Supponiamo di sviluppare il nodo di best bound first E_5 e di ottenere due nodi E_7 e E_8 , nel quale E_8 porta ad una soluzione ammissibile, mentre E_7 porta a due valori. Quali sono possibili valori per LB e UB tali che chiudo tutti i nodi (riconosco subito la soluzione ottima)?
 - Controllo tra i nodi aperti, quindi E_3 ed E_8
 - Il LB deve essere $\geq$ all'incumbent dei nodi che si vogliono chiudere (quindi, al loro UB), mentre l'UB deve essere compatibile con il fatto di essere figlio del nodo best-bound first, quindi essere $\leq$ UB del nodo padre. Prenderò qualsiasi nodo con $LB \geq 7.4$ e $UB \leq 8.2$.
 - Per chiudere anche lo stesso E_8 basterà prendere un qualsiasi valore dentro a questo intervallo.

Consigli dell'esame:

- 2 ore e un quarto di durata
- Pianificare l'esame, sapendo che occorre fare bene l'esercizio 1 (scrivendo domini, non scrivendo cose non lineari, inserendo alcuni vincoli logici)
- Il progetto conviene farlo dopo l'esame, solo nel caso si voglia migliorare il voto; l'importante è fare il progetto e consegnarlo nell'anno in corso; modalità e dettagli si discutono con il prof, previa sua accettazione

Tutorato del prof: Svolgimento alcuni esercizi Appello 10/02/2020 Tema A

Si possono individuare le seguenti variabili:

// x_i : numero di pantaloni secondo il modello $i \in \{1,2\}$

// y_j : numero di spezzoni tagliati secondo gli schemi $j \in \{A, B, C, D\}$

Passando ai vincoli, abbiamo che:

- "si vogliono tagliare almeno 10 spezzoni secondo lo schema A"
$$y_A \geq 10$$
- "i pantaloni di modello 1 devono essere al più il doppio dei pantaloni di modello 2"
$$x_1 \leq 2 * x_2$$

La f.o è come segue (considerando ricavi e costi):

$$\begin{aligned} \max & 80x_1 + 110x_2 - 2y_A - 4y_B - 3y_C - 1y_D \\ \text{s. t.} \\ & x_i \in Z_+, i \in \{1,2\}, y_j \in Z_+, j \in \{A, B, C, D\} \end{aligned}$$

Considero che la quantità di pantaloni sia $\leq$ al numero di spezzoni tagliati:

$$2x_1 + 3x_2 \leq 5y_A + 7y_B + 13y_C + 11y_D \text{ //tasche}$$

$$2x_1 + 2x_2 \leq 4y_A + 10y_B + 5y_C + 8y_D \text{ //gambe}$$

$$5x_1 + 4x_2 \leq 12y_A + 4y_B + 8y_C \text{ //rinforzi}$$

Dobbiamo dire che abbiamo al massimo 500 spezzoni:

$$y_A + y_B + y_C + y_D \leq 500$$

Altro vincolo sta sul tempo:

$$\frac{1}{2}y_A + 3y_B + y_C + y_D \leq 700$$

- "si possono tagliare spezzoni di al massimo tre schemi diversi"

z_j : 1 se taglio $j \in \{A, B, C, D\}$, 0 altrimenti

$$z_A + z_B + z_C + z_D \leq 3$$

Occorre attivare le variabili:

$$y_A \leq Mz_A, y_B \leq Mz_B, y_C \leq Mz_C, y_D \leq Mz_D$$

Oppure (stessa cosa):

$$y_j \leq Mz_j, j \in \{A, B, C, D\}$$

Per M costante sufficientemente grande

- "si vogliono ottenere rinforzi da almeno due schemi diversi"

Non ci sono rinforzi su D, quindi:

$$z_A + z_B + z_C \geq 2$$

Per i 6 crediti va bene, per i 7 serve considerare i vincoli spuri. Mettendo tutte le variabili ad 1, nessuno mi impedisce di mettere tutte le variabili ad 1 senza produrre nulla.

Quindi, per gli spuri:

Scritto da Gabriel

$y_A \geq z_A$ //devo avere almeno un rinforzo da qualche parte

E similmente:

$$y_B \geq z_B, y_C \geq z_C$$

- “ci sono quattro linee di produzione, ciascuna dedicata a uno schema diverso, e il costo complessivo per il setup delle linee di produzione è di 3000 euro se si usano fino a due linee, 3500 se ne usano di più”

La decisione è capire se pago o meno 500 euro in più, tanto comunque pagherò sempre almeno 3000.

In f.o., si usa una variabile binaria $w \in \{0,1\}$, per cui:

$$\max 80x_1 + 110x_2 - 2y_A - 4y_B - 3y_C - 1y_D - 500w$$

Complicandosi la vita, si potrebbe scrivere, al posto di $500w$

$$-3500w - 3000(1 - w)$$

Vogliamo legare a w il fatto di usare 2 o 4 linee diverse, quindi, usando l'idea di prima:

vale 1 quando $w = 4$, vale 0 quando $w = 2 \rightarrow 4w + 2(1 - w)$ oppure $2 + 2w$

$$z_A + z_B + z_C + z_D \leq 4w + 2(1 - w)$$

$$z_j \in \{0,1\}, w \in \{0,1\}$$

Esercizio 5 – PDF di Modellazione

5. Un risparmiatore ha a disposizione, all'inizio di aprile, un budget di 100.000 euro e vorrebbe disporre, all'inizio di agosto, di almeno 150.000 euro attraverso un mix di investimenti. Tutti gli investimenti sono a disposizione all'inizio di ciascuno dei prossimi mesi (da aprile a luglio) e le loro caratteristiche sono riassunte nella seguente tabella, in termini di durata (in mesi), rendimento percentuale e livello di rischio:

Investimento	durata (mesi)	rendimento	rischio
A	1	10%	2
B	2	19%	3
C	3	33%	5
D	1	15%	4

Scrivere il modello di programmazione lineare che aiuti il risparmiatore ad arrivare alla cifra desiderata minimizzando il livello di rischio e tenendo conto che:

- il capitale rientrato alla fine di un investimento è immediatamente a disposizione per altri investimenti;
- a inizio maggio, non è possibile investire nello stesso periodo sia in B che in C;
- per poter usufruire dell'investimento A a inizio aprile è necessario investire, nello stesso periodo, almeno 10.000 euro in B e 30.000 euro in D;
- è possibile investire solo multipli interi di 1.000 euro.

x_{ij} : € investiti in $i \in \{A, B, C, D\}$ all'inizio del mese $j \in \{1, 2, 3, 4\}$

$$x_{A1} + x_{B1} + x_{C1} + x_{D1} \leq 100000 \text{ //(aprile)}$$

Si considera che ritornano tutti gli investimenti e i capitali che ho investito:

$$x_{A2} + x_{B2} + x_{C2} + x_{D2} \leq 100000 + 0.1x_{A1} - x_{B1} - x_{C1} + 0.15x_{D1} \text{ //(maggio)}$$

// $x_{C3} = 0$ in soluzioni ottime, perché tanto tornerebbe dopo 3 mesi

$$x_{A3} + x_{B3} + x_{C3} + x_{D3} \leq 100000 + 0.1x_{A2} + 0.19x_{B1} + 0.15x_{D1} + 0.1x_{A2} + 0.15x_{D2} - x_{C1} - x_{B2} - x_{C2} \text{ //(giugno)}$$

$$x_{A4} + x_{D4} \leq 100000 + 0.1x_{A1} + 0.19x_{B1} + 0.33x_{C1} + 0.15x_{D1} + 0.1x_{A2} + 0.19x_{B2} + 0.15x_{D2} + 0.1x_{A3} + 0.15x_{D3} - x_{C2} - x_{B3} \text{ //(luglio)}$$

$$0.1(x_{A1} + x_{A2} + x_{A3} + x_{A4}) + 0.19(x_{B1} + x_{B2} + x_{B3}) + 0.33(x_{C1} + x_{C2}) + 0.15(x_{D1} + x_{D2} + x_{D3} + x_{D4}) \geq 150000 \text{ //(agosto)}$$

Per la f.o., si deve minimizzare il livello di rischio:

$$\min 2(x_{A1} + x_{A2} + x_{A3} + x_{A4}) + 3(x_{B1} + x_{B2} + x_{B3}) + 5(x_{C1} + x_{C2}) + 4(x_{D1} + x_{D2} + x_{D3} + x_{D4})$$

- “a inizio maggio, non è possibile investire nello stesso periodo sia in B che in C”

Logicamente, si esprime come un OR logico

$$y_{B2}, y_{C2} \in \{0,1\}, \text{ con } y_{ij} = 1 \text{ se } x_{ij} = 0, 0 \text{ altrimenti}$$

$$y_{B2} + y_{C2} \leq 1$$

Occorre poi attivare le variabili:

$$x_{B2} \leq M y_{B2}, x_{C2} \leq M y_{C2}$$

Per M costante sufficientemente grande

- “per poter usufruire dell’investimento A ad inizio aprile è necessario investire, nello stesso periodo, almeno 10000 euro in B e 30000 euro in D”

// $z_B = 1$ se almeno 10000 in B, 0 altrimenti

// $z_D = 1$ se almeno 30000 in D, 0 altrimenti

Quando A vale qualcosa, almeno una delle due tra B e D vale 0:

$$x_{A1} \leq M z_B, x_{A1} \leq M z_D$$

$$x_{B1} \geq 10000 z_B, x_{D1} \geq 30000 z_D$$

Poi:

- “è possibile investire solo multipli interi di 1000 euro”

Questo ha a che fare con i domini:

$$x_{ij} \in R_+,$$

Ho due modi per scriverlo:

$x_{ij} \in Z_+$, (modificando tutti i vincoli esprimendo tutto in migliaia di euro, es. 150000 come 150)

$w_{ij} \in Z_+, x_{ij} = 1000 w_{ij}$ (vincolo più generale, per esprimere tutto in funzione delle migliaia di euro)

Tutorato del prof: Soluzioni ad alcuni esercizi di “Varie”

Condizioni di ottimalità del simplesso → (da inserire)

Condizioni di complementarità primale-duale

- (Slide 27/32 NO, in quanto dedicate alla dualità forte)
- (Prima definizione slide 28/32)
- (Forma delle condizioni che applichiamo a 31/32, come enunciato, poi usiamo quelle a 32/32)

- A) enunciare le condizioni di complementarità primale-duale in generale e B) applicare tali condizioni per dimostrare che $(x_1, x_2, x_3) = (1, 0, -2)$ è soluzione ottima del seguente problema:

$$\begin{array}{llllll} \max & 3 & x_1 & - & x_2 & \\ \text{s.t.} & & x_1 & + & x_2 & \geq 1 \\ & 2 & x_1 & + & x_2 & + & x_3 & \leq 0 \\ & 2 & x_1 & - & x_2 & - & x_3 & = 4 \\ & 2 & x_1 & & & - & x_3 & \geq -1 \\ & x_1 \geq 0 & x_2 \text{ libera} & & & x_3 \leq 0 \end{array}$$

A) Dati due problemi generici *mix* e *max* nelle forme:

$$\begin{array}{l} \min c^T x \quad Ax \sim b \quad x \sim 0 \\ \max u^T b \quad u^T A \sim c \quad u^T \sim 0 \\ \bar{x} \text{ e } \bar{u} \text{ ottime} \\ \Leftrightarrow \\ \bar{u}_i (u^T A_j - c_j) = 0 \\ \bar{x}_j (a_i^T x - b_i) = 0 \end{array}$$

Con $\bar{x}$ ammissibile primale e $\bar{u}$ ammissibile duale

1) Verifica dell'ammissibilità primale della soluzione data

X è ammissibile primale?

$$\begin{aligned}
 x_1 + x_2 &= 1 \text{ (OK)}, \\
 2x_1 + x_2 + x_3 &= 0 \text{ (OK)} \\
 2x_1 - x_2 - x_3 &= 4 \text{ (OK)} \\
 2x_1 - x_3 &> 1 \text{ (OK)} \\
 x_1 > 0, x_2 &\text{ libera}, x_3 < 0
 \end{aligned}$$

2) Passaggio al problema duale

$$\begin{aligned}
 \min u_1 + 0u_2 + 4u_3 - 1u_4 \\
 \text{s. t. } u_1 + 2u_2 + 2u_3 + 2u_4 &\geq 3 \\
 u_1 + u_2 + u_3 &= -1 \\
 u_2 - u_3 - u_4 &\leq 0 \\
 u_1 \leq 0, u_2 &\leq 0, u_3 \text{ libera}, u_4 \leq 0
 \end{aligned}$$

3) Condizioni di complementarità primale-duale

$$\begin{aligned}
 x_1(u_1 + 2u_2 + 2u_3 + 2u_4 - 3) &= 0 \rightarrow 1(u_1 + 2u_2 + 2u_3 + 2u_4 - 3) \rightarrow u_1 + 2u_2 + 2u_3 + 2u_4 = 3 \\
 x_2(u_1 + u_2 + u_3 + 1) &= 0 \rightarrow \text{Vero per ammissibilità duale } (x_2 = 0 \text{ per il fatto delle condizioni di complementarità, non tanto perché } x_2 = 0) \\
 x_3(u_2 - u_3 - u_4 - 0) &= 0 \rightarrow u_2 - u_3 - u_4 = 0 \\
 u_1(x_1 + x_2 - 1) &= 0 \rightarrow u_1 * 0 \rightarrow 0 \rightarrow // \\
 u_2(2x_1 + x_2 + x_3 - 0) &= 0 \rightarrow u_2 * 0 = 0 \rightarrow // \\
 u_3(2x_1 - x_2 - x_3 - 4) &= 0 \rightarrow \text{Vero per ammissibilità primale} \\
 u_4(2x_1 - x_3 - 1) &= 0 \rightarrow u_4 * 5 = 0 \rightarrow u_4 = 0
 \end{aligned}$$

4) Sistema di equazioni per l'imposizione delle condizioni di complementarità primale duale (ccpd) trovate e delle condizioni di ammissibilità duale:

$$\begin{aligned}
 u_1 + 2u_2 + 2u_3 + 2u_4 &= 3 \text{ (CCPD)} \\
 u_1 + u_2 + u_3 &= -1 \text{ (ammissibilità duale)} \\
 u_2 - u_3 - u_4 &= 0 \text{ (CCPD)} \\
 u_4 &= 0 \text{ (CCPD)}
 \end{aligned}$$

$$u_1 = -1, u_2 = 1, u_3 = 1, u_4 = 0$$

5) Verifica ammissibilità duale

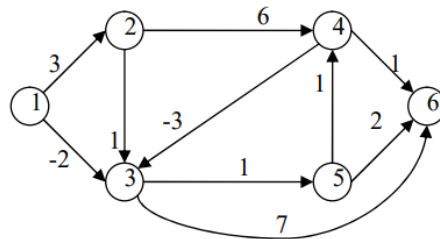
$$u_1 < 0, u_2 > 0, u_3 \text{ libera}, u_4 = 0$$

Primo vincolo duale a posto per costruzione, secondo vincolo duale a posto per costruzione, terzo vincolo duale a posto per costruzione

6) Conclusioni

La u è ammissibile duale ed è in scarti complementare con la x data $\Rightarrow$ ottime

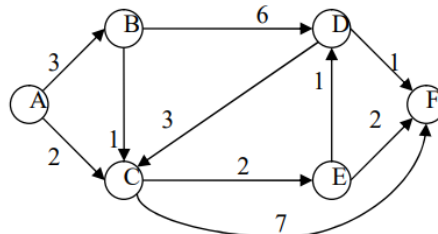
Si consideri il seguente grafo:



- si scelga un algoritmo per determinare i cammini minimi dal nodo 1 verso tutti gli altri nodi e si motivi la scelta;
- si applichi l'algoritmo scelto (riportare e giustificare i passi dell'algoritmo in una tabella);
- L'algoritmo ha individuato un ciclo negativo? Giustificare la risposta.
- Riportare l'albero e il grafo dei cammini minimi, oppure il ciclo negativo (in ogni caso, si descriva il procedimento utilizzato).

Qui usiamo Bellman-Ford e non Dijkstra

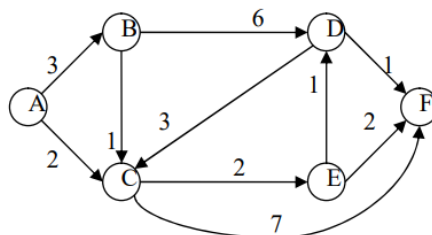
Si consideri il seguente grafo:



- si scelga il miglior algoritmo tra quelli presentati per determinare i cammini minimi dal nodo A verso tutti gli altri nodi e si motivi la scelta;
- si applichi l'algoritmo scelto (riportare e giustificare i passi dell'algoritmo in una tabella);
- si disegni l'albero e il grafo dei cammini minimi, descrivendo il procedimento usato.

Qui possiamo applicare Dijkstra, essendo migliore computazionalmente

Si consideri il seguente grafo:



- si scelga il miglior algoritmo tra quelli presentati per determinare i cammini minimi CON MASSIMO 4 ARCHI dal nodo A verso tutti gli altri nodi e si motivi la scelta;
- si applichi l'algoritmo scelto (riportare e giustificare i passi dell'algoritmo in una tabella);
- si ricavi un cammino con al più 4 archi da A verso E e un cammino minimo con al più 3 archi da A a F: DESCRIVERE IL PROCEDIMENTO.
- è possibile, ricavare direttamente dalla tabella ottenuta albero e/o grafo dei cammini minimi? Giustificare la risposta.

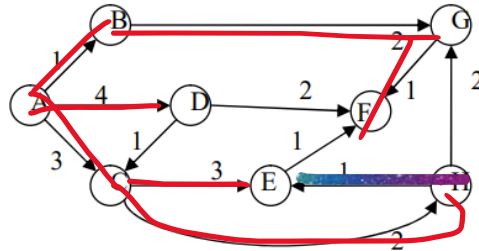
Qui possiamo applicare Bellman-Ford, avendo il maggior numero di archi; può succedere che applicando Dijkstra arriviamo alla convergenza, ma qui non è corretto applicarlo. Si consideri che applichiamo sempre Bellman-Ford con i nodi aggiornati. Le etichette non sono stabili, dato che ci sono cammini con al massimo 4 hop e non trovo particolari alberi (avrei bisogno di ulteriori iterazioni).

Domanda: La complessità computazionale di Dijkstra da cosa dipende?

Risposta: Dal numero di nodi e dal numero di archi (soprattutto quest'ultimo)

Svolgiamo il seguente esercizio:

Si consideri il seguente grafo:



- si scelga un algoritmo appropriato e si motivi la scelta;
- si calcolino i cammini minimi dal nodo A verso tutti gli altri nodi (i passi dell'algoritmo vanno riportati in una tabella e giustificati);
- si disegni l'albero e il grafo dei cammini minimi, descrivendo il procedimento usato.

a) Il miglior algoritmo in questo contesto è Dijkstra, in quanto più efficiente e usabile in quanto tutti i costi ridotti sono positivi.

b) Ora, il calcolo dei cammini minimi, riportati in tabella. Si ricorda che:

- $\hat{v}$ rappresenta l'etichetta minima di ogni iterazione
- $\bar{S}$ rappresentano le etichette ancora da fissare
- Il segno * rappresenta l'etichetta fissata
- Il segno - rappresenta l'etichetta controllata ma non aggiornata
- Il segno x rappresenta l'etichetta non controllata perché il nodo è già fissato
- Gli spazi vuoti servono per indicare che non considero più l'etichetta nelle varie iterazioni in quanto fissata
- L'algoritmo termina quando non ci sono più nodi in $\bar{S}$
- Ad ogni passo, seleziono il minimo e il nodo raggiunto arrivando dal minimo

Iterazione	Nodo A	Nodo B	Nodo C	Nodo D	Nodo E	Nodo F	Nodo G	Nodo H	$\bar{S}$	$\hat{v}$
Inizio	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	A, B, C, D, E, F, G, H	
$h = 1$	*	1_A	3_A	4_A					B, C, D, E, F, G, H	A
$h = 2$		*					3_B		C, D, E, F, G, H	B
$h = 3$			*		6_C			5_C	D, E, F, G, H	C
$h = 4$						4_G	*		D, E, F, H	G
$h = 5$						*			D, E, H	F
$h = 6$			x	*		x			E, H	D
$h = 7$					-				E	H
$h = 8$					*	x			$\emptyset$	E

c)

Pertanto, come abbiamo visto per gli algoritmi label correcting in caso di convergenza, è possibile derivare l'albero (risp. il grafo) dei cammini minimi attraverso i puntatori ai predecessori (risp. la verifica della saturazione dei vincoli duali sugli archi).

Concretamente, avremo che:

- In rosso riportiamo l'albero dei cammini minimi
- In arcobaleno riportiamo il grafo (composto come sempre da *tutti* i cammini minimi, quindi fissando le etichette ottime con lo stesso colore e scegliendo come cammino sia l'albero che tutte le altre etichette con costo $\leq$); in questo caso, grafo ed albero sono diversi.
- Come si riconoscono sul tableau del simpleso le condizioni di illimitatezza per un problema di minimo? Giustificare la risposta.
- Si enunci e si giustifichi la regola adottata dal metodo del simpleso per la selezione della variabile uscente nelle operazioni di cambio base.
- Le condizioni impongono di avere un coefficiente di costo ridotto con tutti i coefficienti sotto di esso ≤ 0 e la condizione specifica per il valore della f.o. $+z = \bar{z}_B + \bar{c}_h x_h$ e procedendo, il valore in base sarà $x_{B_i} = \bar{b}_i - \bar{c}_{i_h} x$. Quest'ultima è ≥ 0 e, per qualsiasi valore, si avrà un valore della soluzione ammissibile; tuttavia, la f.o. va a $-\infty$, in quanto $\bar{c}_h < 0$.
- Supponendo di avere una variabile a costo ridotto negativo che facciamo entrare in base. Con gli stessi ragionamenti ed equazioni di prima e avremo $x_h \leq$ di un certo valore. Con il sistema di prima, scelgo la variabile che esce dalla base come il minimo dei rapporti, essendo sicuro che quella sarà 0, mentre tutte le altre saranno diverse da 0. Essendo problema di ammissibilità, devo garantire che le variabili siano tutte ≥ 0 ; per fare questo, scelgo le variabili che rispettano la regola del rapporto minimo. Quindi $\bar{b}_i - \bar{c}_{i_h} x \geq 0$.

Nel caso delle risposte sulle complessità, basti guardare le dispense, nel caso aggiungere alcune considerazioni.

Si consideri il seguente tableau del simpleso:

	x_1	x_2	x_3	x_4	x_5	b
$-z$	0	1/2	0	1	0	9
x_3	0	1/2	1	2	0	0
x_5	0	0	0	-1	1	2
x_1	1	-1/2	0	1	0	1

Indicare, senza svolgere operazioni di pivot, 3 basi ottime (nei termini delle variabili che le compongono) del corrispondente problema di programmazione lineare.

Basi ottime:

- $[x_3, x_5, x_1] \rightarrow$ Data direttamente dal problema

Sfrutto il fatto che la base è degenere (con stessa soluzione ottima, ma la base è diversa)

Con la regola del rapporto minimo, individuo basi alternative

$$B_2 = [x_2, x_5, x_1]$$

$$B_3 = [x_4, x_5, x_1]$$

Nel tableau capiamo che ci sono infinite soluzioni ottime avendo uno zero in più nella riga dei costi ridotti, con le stesse soluzioni ammissibili facendo l'operazione di pivot.