Modellazione

Funzione Obiettivo e Vincoli Principali

- La funzione obiettivo (min/max) definisce l'obiettivo da ottimizzare
- I vincoli si dividono in:
  - Quantitativi: limitano risorse/capacità usando variabili con commenti esplicativi
  - Qualitativi: gestiscono relazioni logiche tra variabili usando costanti M sufficientemente grandi

Attivazioni e Domini

Le attivazioni seguono la forma: $x \leq M * y$, dove:

- M è una costante sufficientemente grande ma non eccessiva
- y è tipicamente una variabile binaria

I domini possibili sono:
- $\mathbb{Z}_+$: Variabili intere non negative
- $\mathbb{R}_+$: Variabili reali non negative
- $\{0, 1\}$: Variabili binarie per decisioni sì/no

Vincoli secondari

- Almeno: $\geq$
- Al massimo: $\leq$
- Esattamente: $=$
- Apro x ma non apro y: $x \leq (1 - y)$
- Budget: Variabile apposita
- Sconto/Penalità: SCELGO/POTREBBE succedere (binaria)
- Costi fissi: Variabile apposita

Vincoli impliciti

Questi dipendono dallo schema del problema!

- Disponibilità ($x \leq numero$)
- Richiesta ($x >= numero$)

Dualità

- Enunciato CCPD

Enunciato delle condizioni di complementarietà primale duale:

Dati un problema primale $\min c^T x$ s.t. $Ax \geq b, x \in \mathbb{R}^n_+$ e il corrispondente duale $\max u^T b$ s.t. $u^T A \leq c, u \in \mathbb{R}^m_+$, e due vettori $\bar{x} \in \mathbb{R}^n_+$ e $\bar{u} \in \mathbb{R}^m_+$, $\bar{x}$ e $\bar{u}$ sono soluzioni ottime rispettivamente per il primale e per il duale se e solo se: $\bar{x}$ è ammissibile primale, $\bar{u}$ è ammissibile duale, $u_i(a_i^T x - b_i) = 0, \forall\ i = 1 \ldots m$, e $(c_j - u^T A_j)x_j = 0, \forall\ j = 1 \ldots n$, dove a_i^T è la riga i-esima di A e A_j è la colonna j-esima di A.

Pratica

- Passo al problema duale
- CCPD
  - Primali
  - Duali
- Sistema CCPD E ammissibilità duale
- Verifica ammissibilità duale
- Applicazione dualità forte

Teoria

- Dualità debole
  - Primale illimitato - Duale inammissibile
- Dualità forte
  - Stesso valore f.o. primale e duale (per verifica!)

(Happy Ending)
1. x è ammissibile primale (come da verifica)
2. u è ammissibile duale (come da costruzione e da verifica)
3. x, u sono in scarti complementari
4. Le due soluzioni sono ottime per i rispettivi problemi primale e duale
5. Per verifica si confrontino i valori delle f.o. dei problemi primale e duale; saranno uguali per il teorema della dualità forte

(Bad Ending) → Si noti che il problema chiede "dimostrare se è ottima", cosa che negli esercizi d'esame non capita mai

1. La soluzione trovata è l'unica soluzione del sistema di cui al punto 4 e, quindi, l'unica che soddisfa i vincoli duali di uguaglianza e che è in scarti complementari con la soluzione primale data. Tale soluzione però non è ammissibile per il problema duale. Pertanto non è possibile trovare nessuna soluzione ammissibile duale che sia in scarti complementari con la soluzione primale data, che, quindi, non è ottima

Simplesso

Pratica

La base deve essere:

- In forma canonica (matrice identità nelle colonne)
- Ammissibile (termini noti non negativi)

Forma standard richiede:

- Obiettivo di minimo
- Vincoli di uguaglianza (usando slack)
- Variabili non negative (usando variabili cappello)

Ad ogni iterazione verificare:

- Ottimalità: costi ridotti tutti ≥ 0
- Ammissibilità: termini noti ≥ 0
- Illimitatezza: colonna di costo ridotto negativo con coefficienti ≤ 0
- Base
  - Colonne matrice identità in ordine sparso
  - Base ammissibile (segno quali variabili)
- Forma standard
  - Minimo
  - Portare tutto ad uguaglianza (slack)
  - Se variabili negative --> aggiungere le cappello
- Ad ogni step
  - È ottima?
    - Costi ridotti tutti positivi (oppure ≥ 0)
  - È ammissibile?
    - Esiste una variabile negativa nella colonna più a destra $\overline{b_i}$
  - È illimitata?
    - Costi ridotti tutti negativi (oppure ≤ 0)
- Bland
  - Chi entra?
    - Variabile con indice minore e costo ridotto negativo
  - Chi esce
    - Rapporto minimo e prendo l'indice minore
- Alla fine
  - Saturo
    - Slack maggiore di 0
  - Lasco
    - Slack pari a 0

Sia che si parta da un problema di massimo che da un problema di minimo, la z finale sarà il valore della z finale moltiplicato per -1

Teoria

- Individuo la base ottima
- Pivot
  - Ammissibile se Bland e rapporto minimo validi!
- Calcolo valore f.o.
  - $\{f.o.\} * \{-1\} + \{x_i\ ottimo\} * \{rapporto\ minimo\}$
- Problema illimitato
  - Vedi sopra

Se il problema primale è illimitato, il problema duale è inammissibile (teorema della dualità debole)

- Problema con base degenere
  - Il valore della f.o. non migliore/non peggiora (non cambia) con almeno ≥ 2 variabili con stesso rapporto minimo

Bellman-Ford

- Costi negativi
- Meno efficiente
- Numero massimo di archi
- Capita ciclo negativo

Iterazione	Nodo A	Nodo B	Nodo C	Nodo D	Nodo E	Nodo F	Aggiornamenti
Inizio	$0_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	A
$h = 1$	$0_\wedge$	3_A	2_A	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	B, C
$h = 2$	$0_\wedge$	3_A	2_A	9_B	4_C	9_C	D, E, F
$h = 3$	$0_\wedge$	3_A	2_A	5_E	4_C	6_E	D, F
$h = 4$	$0_\wedge$	3_A	2_A	5_E	4_C	6_E	F

Branch and Bound

- Incumbent = Valore ottimo ATTUALMENTE
- Prendo il minimo/massimo sempre considerando nodi aperti o tutti i nodi (incumbent)
- Best Bound First = Max/Min migliore a seconda
- Range ottimo = Viene dato dalle proprietà del problema
- Ultimo punto del problema = Nuovo intervallo ottimo

Tabella molto semplice da seguire per rispondere ad ogni domanda d'esame sul Branch and Bound:

	min $[LB;\ SA]$	max $[SA;\ UB]$
min o max	LB sempre crescente	UB sempre decrescente
Chiusura nodi	LB $\geq$ SA più bassa	UB $\leq$ SA più alta
Intervallo ottimo	$[\min\ LB\ figli;\ \min\ SA^*\ generale]$	$[\max\ SA^*\ generale;\ \max\ UB\ figli]$
Nuovo nodo	LB = SA	SA = UB
Valori ottimi	LB padre $\leq$ SA $\leq$ LB min**	UB max** $\leq$ SA $\leq$ UB padre

* Non è detto che la SA sia ottima se ci sono nodi con LB più basso (min) o UB più alto (max)
** Nodi aperti senza considerare il padre.
Scelta del nodo da sviluppare: Best Bound First (BBF) prende il nodo aperto più promettente (LB più basso in min, UB più alto in max), mentre il Depth First prende il nodo aperto al livello più profondo.

6. Si vuole risolvere con **AMPL** un problema di trasporto di alberi da un insieme di origini I a un insieme di destinazioni J. Ciascuna origine i mette a disposizione O_i alberi e ciascuna destinazione richiede D_j alberi. Il costo unitario di trasporto da i a j è C_{ij} e si ha un costo fisso F_i per l'organizzazione dei trasporti da ciascuna origine i. Non è inoltre possibile organizzare il trasporto in più di N origini. Il modello per la minimizzazione dei costi è riportato affianco e utilizza le variabili x_{ij} per indicare il numero di alberi trasportati da i a j, e y_i che vale 1 se si organizza il trasporto da i, 0 altrimenti.

$$\min \sum_{i\in I, j\in J} C_{ij}\, x_{ij} - \sum_{i\in I} F_i\, y_i$$
$$\text{s.t.} \sum_{i\in I} x_{ij} \geq D_j \quad , \quad \forall j \in J$$
$$\sum_{j\in J} x_{ij} \leq O_i\, y_i \quad , \quad \forall i \in I$$
$$\sum_{i\in I} y_i \leq N$$
$$x_{ij} \in \mathbb{Z}_+,\ \ y_i \in \{0,1\},\ \ \forall\, i \in I,\ \ j \in J$$

a. Si traduca nel linguaggio **AMPL** il modello proposto (**file .mod**).

b. Si produca il **file .dat** per l'istanza con origini Croazia, Svezia, Gran Bretagna e Canada (disponibilità di 1000, 2000, 3000 e 4000 alberi rispettivamente), destinazioni Italia, Francia e Germania (con richieste di 5000, 3000 e 2000 rispettivamente), $N = 3$, costi fissi F_i di 1000 euro per tutte le origini, e costi di trasporto verso Italia, Francia e Germania (nell'ordine) pari a: dalla Croazia 10, 20 e 30 euro; dalla Svezia 40, 50 e 60 euro; dalla Gran Bretagna 70, 80 e 90 euro; dal Canada 100, 110 e 120 euro.

c. Si scriva uno script di AMPL (**file .run**) che risolve l'istanza specificata e visualizza il valore della funzione obiettivo e delle variabili per una soluzione ottima.

ATTENZIONE: se vincoli matriciali nel dat, allora metti (tr) per la forma (i, j)

Punto a)

```
set I;              set J;
param O{I};         param D{J};
param C{I,J};       param F{I};
param N;
var x{I,J} >=0 integer;
var y{I} binary;
minimize fo: sum{i in I, j in J} C[i,j]*x[i,j] - sum{i in I} F[i]*y[i];
s.t. d{j in J}: sum{i in I} x[i,j] >= D[j];
s.t. o{i in I}: sum{j in J} x[i,j] <= O[i] * y[i];
s.t. n: sum{i in I} y[i] <= N;
```

Punto b)

```
set I := Croazia Svezia GranBretagna Canada;
set J := Italia Francia Germania;
param :         F     O :=
Croazia         1000  1000
Svezia          1000  2000
GranBretagna    1000  3000
Canada          1000  4000;
param D := Italia 5000 Francia 3000 Germania 2000;
param N := 4;
param C :       Italia      Francia     Germania :=
Croazia         10          20          30
Svezia          40          50          60
GranBretagna    70          80          90
Canada          100         110         120;
```

Punto c)

```
reset;
model ampl.mod;
data ampl.dat;
option solver cplexamp;
solve;
display fo, x, y;
```

Si utilizza una tabella che riporta una riga per ogni iterazione dell'algoritmo. Ogni colonna della tabella è dedicata ad un nodo e riporta, iterazione dopo iterazione, l'evoluzione delle rispettive etichette. L'ultima colonna riporta i nodi aggiornati nel corso dell'iterazione: all'iterazione successiva è sufficiente controllare solo gli archi uscenti da questi nodi (vincolo di Bellman).

La tabella riporta al riga 0 di inizializzazione e una riga per ogni iterazione. All'iterazione h si controllano gli archi (i, j) uscenti da ciascun nodo i nella colonna $Aggiornati$ alla riga $h-1$, e si aggiornano i costi e i predecessori del nodo j all'iterazione h qualora l'etichetta del nodo i all'iterazione $h-1$ più il costo dell'arco (i, j) sia strettamente minore dell'etichetta corrente del nodo j.

L'algoritmo si ferma qualora la lista dei nodi aggiornati sia vuota (convergenza delle etichette ai costi dei cammini minimi da A verso gli altri nodi).

Dijkstra

- Solo costi positivi
- Più efficiente

Legenda della tabella (la scrivo io per chiarezza di contenuto questa, ndr; occorre giustificare i passi):
- $\hat{v}$ 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 (normalmente ha stesso costo rispetto a quello che attualmente ha e quindi non vado ad aggiornare)
- Il segno x rappresenta l'etichetta non controllata perché il nodo è già fissato

Iterazione	Nodo A	Nodo B	Nodo C	Nodo D	Nodo E	Nodo F	$\bar{S}$	$\hat{v}$
Inizio	$0_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	A, B, C, D, E, F	
$h = 1$	*	3_A	2_A	$+\infty_\wedge$	$+\infty_\wedge$	$+\infty_\wedge$	B, C, D, E, F	A
$h = 2$		-	*	9_B	4_C	9_C	B, D, E, F	C
$h = 3$		*		5_E	-	6_E	D, E, F	B
$h = 4$				-	*	-	D, F	E
$h = 5$				*		-	F	D
$h = 6$						*	$\emptyset$	F

Ad ogni iterazione, percorriamo il grafo e scegliamo il percorso con costo minore. Ad ogni iterazioni, scegliamo e fissiamo un'etichetta che ha costo minore, aggiungendo un'etichetta all'insieme di quelle fissate e controllando quanto accade per le altre (segnando etichette controllate ma non aggiornate oppure non controllate perché il nodo è già fissato).

Si procede inoltre con le verifiche della condizione di Bellman (vincolo duale) sugli archi che escono dall'etichetta minima e portano nodi nell'insieme dei nodi da fissare. Le etichette calcolate e i relativi puntatori rappresentano, rispettivamente, una soluzione duale ammissibile e i predecessori su dei cammini dall'origine ai diversi nodi.

L'algoritmo termina quando tutte le etichette sono state fissate, cioè quando tutti i nodi di $\bar{S}$ vengono trasferiti in S e quando tutte le etichette sono state correttamente fissate, quindi $\bar{S} = \emptyset$

Cammini minimi

- Un cammino minimo dal nodo i al nodo j rappresenta la sequenza di archi che collega i due nodi con costo totale minimo
- Si indica con π(j) il predecessore del nodo j nel cammino minimo

Strutture derivate

1. **Albero dei cammini minimi**
   - Esiste solo in assenza di cicli negativi e vincoli sul numero di archi
   - Fornisce esattamente un cammino minimo dall'origine a ogni nodo destinazione
   - Si costruisce seguendo la catena dei predecessori π(j)
2. **Grafo dei cammini minimi**
   - Include tutti i possibili cammini di costo minimo tra coppie di nodi
   - Può esistere anche con vincoli sul numero di archi
   - È un sovrainsieme dell'albero (quando quest'ultimo esiste)
   - Include archi (i,j) dove il vincolo duale è saturo: π(j) = π(i) + c(i,j)
3. **Ciclo negativo**
   - Rende impossibile definire sia l'albero che il grafo dei cammini minimi
   - Si identifica quando le etichette continuano ad aggiornarsi dopo n iterazioni
   - Si ricostruisce seguendo la catena dei predecessori da un nodo aggiornato nell'ultima iterazione