Esercizi tipo
Ricerca Operativa

Andrea Auletta
Alessio Turetta
Luca Pierobon
Samuel Scarabottolo

1 Avvertenze

Questo documento non è fatto per chi non sa niente di Ricerca Operativa, piuttosto serve come quick reference per fare maggior parte degli esercizi d'esame.
Le fonti sono state prese da Prof. De Giovanni e vari documenti su questo MEGA.
Per dubbi, domande e possibili correzioni potete scrivere in DM a @Gyarik (Telegram)
o Gyarik#9854 (Discord).

2 Simplesso

2.1 Nozioni teoriche

Una base è degenere quando ci sono almeno due righe con lo stesso rapporto minimo (sulla colonna che entra appena trovata), risulterà cioè che almeno una esce dalla base con valore 0, mentre le altre assumeranno valore 0 rimanendo in base.
Il valore della funzione obiettivo al cambio base equivale al rapporto minimo moltiplicato per il costo ridotto del pivot, più il valore di b per il segno di z, oppure:

$$z_{base} = \frac{b_{pivot}}{pivot} \cdot c_{pivot} + z_{minimo} \cdot b_{sol}$$

Perché non è consentita l'operazione di pivot sull'elemento? Solitamente perché l'elemento cerchiato è il denominatore di un rapporto **NON** minimo.
Se il simplesso risulta in soluzione ottima, allora il problema duale avrà una soluzione ottima identica secondo il corollario della dualità forte.
Invece se il simplesso risulta in illimitatezza, allora il problema duale sarà inammissibile per il corollario della dualità debole.

2.2 Passaggi

2.2.1 Forma Standard

1. La funzione obiettivo deve essere min:
   a. Se passo da max a min inverto i segni della funzione obiettivo
2. Tutti i vincoli devono essere di uguaglianza:
   a. Da $\leq$ sommo una variable (slack) $+x_n$
   b. Da $\geq$ sottraggo una variable (slack) $-x_n$
3. I valori a destra delle equazioni devono essere positivi (se non lo sono si cambia il segno)
4. I domini devono essere sempre $\geq$
   a. Altrimenti creo una variabile $\hat{x}_i = -x_i$ e sostituisco x_i con $\hat{x}_i$

2.2.2 Tableau

1. Forma canonica: base (matrice identità) sormontata da 0 (nei costi ridotti)
2. Ammissibilità: Condizione $b_i \geq 0$
3. Soluzione ottima: costi ridotti ≥ 0. Se ottima salta a **Soluzione ottima**.
4. Problema illimitato: colonna del costo ridotto negativo con solo valori ≤ 0. Se illimitato l'esercizio finisce
5. Cambio base: entra la variabile con indice minore che ha costo minore negativo
6. Cambio base: esce il rapporto minimo tra valore della colonna b e valore nella colonna della variabile entrante
7. Re-iterazione dal punto 1.

2.2.3 Soluzione ottima

1. Se il problema originale è già minimo allora $z_{minimo} = -1 \cdot b$
2. Se il problema originale è massimo allora $z_{massimo} = -1 \cdot z_{minimo}$

3 Complementarietà Primale-Duale

3.1 Definizione teorica

Dati un problema primale $\min c^T x \ s.t. \ Ax \geq b, x \in \mathbb{R}^m_+$ e il corrispondente duale $\max u^T b \ s.t. \ u^T A \leq c, u \in \mathbb{R}^m_+$, e due vettori $\bar{x} \in \mathbb{R}^m_+$ 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, $u_i(a_i^T x - b_i) = 0, \forall i = 1 \dots m$, e $(c_j - u^T A_j)x_j = 0, \forall j = 1 \dots n$, dove a_i^T è la riga i-esima di A e A_j è la colonna j-esima di A.
Definizione matematica delle condizioni di complementarietà:

$$\min c^T x : x \geq 0,\ Ax \geq b$$
$$\max u^T b : u \geq 0,\ u^T A \leq c^T$$

$$\begin{array}{c} x \text{ e } u \text{ ottime} \\ \text{primale e duale } (risp.) \end{array} \iff \begin{array}{ll} Ax \geq b \wedge x \geq 0 & \text{(ammissibilità primale)} \\ u^T A \leq c^T \wedge u \geq 0 & \text{(ammissibilità duale)} \\ \left.\begin{array}{l} u^T(Ax - b) = 0 \\ (c^T - u^T A)x = 0 \end{array}\right\} & \text{(complementarietà)} \end{array}$$

3.2 Passaggi

3.2.1 Verifica vincoli problema primale

Sostituisci le x della soluzione data dall'esercizio sui vincoli e domini del problema dato.

3.2.2 Generazione problema duale

Usando la tabella qui sotto, scrivi il problema duale partendo dal problema iniziale, dove c_i è il coefficiente della x_i nella funzione obiettivo. La funzione obiettivo in u consiste dei b_j nel problema iniziale moltiplicati uno ad uno per le u in ordine.

Primale min (x)	**Duale** max (u)	**Primale** max (x)	**Duale** min (u)
$x_i \geq 0$	$\mathit{condizione} \leq c_i$	$x_i \geq 0$	$\mathit{cond} \geq c_i$
$x_i \leq 0$	$\mathit{cond} \geq c_i$	$x_i \leq 0$	$\mathit{cond} \leq c_i$
x_i *libera*	$\mathit{cond} = c_i$	x_i *libera*	$\mathit{cond} = c_i$
$\mathit{cond} \geq b_j$	$u_i \geq 0$	$\mathit{cond} \geq b_j$	$u_i \leq 0$
$\mathit{cond} \leq b_j$	$u_i \leq 0$	$\mathit{cond} \leq b_j$	$u_i \geq 0$
$\mathit{cond} = b_j$	u_i *libera*	$\mathit{cond} = b_j$	u_i *libera*

3.2.3 Condizioni di complementarietà primale-duale

Scegli equazioni che rispettano le condizioni di complementarietà,
Con $u_i \cdot (\mathit{condizione}\ 'i'\ \mathit{in}\ x + \mathit{numero\ a\ destra}) = 0$

- Se $\mathit{cond} \neq 0$ allora si ha una condizione di complementarietà
- Altrimenti $\mathit{cond} = 0$, quindi **NON** si deve considerare
- se la condizione in sé è di uguaglianza allora si ha una conseguenza di ammissibilità primale, cioè $\mathit{cond} = 0$ quindi **NON** si considera

Con $x_j \cdot (\mathit{condizione}\ 'j'\ \mathit{in}\ u + \mathit{numero\ a\ destra}) = 0$

- Se $x_j \neq 0$ allora si ha una condizione di complementarietà
- Altrimenti $x_j = 0$, quindi **NON** si deve considerare
- se la condizione in sé è di uguaglianza allora si ha una conseguenza di ammissibilità duale, cioè $\mathit{cond} = 0$ quindi **SI** deve considerare

3.2.4 Soluzione problema duale

Costruisci e risolvi un sistema di equazioni con tutte le equazioni considerate nel punto precedente; dovresti averne tante quante u ci sono, altrimenti il procedimento si è svolto erroneamente.

3.2.5 Verifica vincoli problema duale

Come nel primo passaggio, sostituisci le u calcolate sui vincoli e domini del problema duale, se sono tutti verificati allora l'esercizio è completato.

3.2.6 Conclusioni

Se tutti i passaggi precedenti sono risultati con esito positivo basta ripetere "Poiché entrambi i problemi primale e duale hanno soluzione ammissibile, ed x e u sono in scarti complementari per costruzione, allora la soluzione proposta è ottima", altrimenti se al passaggio 5 c'è un vincolo non rispettato allora non è ottima poiché la soluzione in u non è ammissibile.

4 Cammino minimo

Si usa Bellman-Ford quando si hanno pesi negativi oppure viene richiesto di trovare il percorso minimo in una certa quantità limitata di archi. Altrimenti si usa Dijkstra.

4.1 Bellman-Ford

L'algoritmo di Bellman-Ford è un algoritmo di ricerca di cammini minimi in un grafo che supporta archi con pesi negativi. Funziona eseguendo iterazioni successive su tutti gli archi del grafo, adattando i pesi dei cammini minimi a ciascun nodo fino a quando non viene raggiunta la stabilità. La complessità computazionale di questo algoritmo è $O(m \cdot n)$, dove n è il numero di nodi e m è il numero di archi nel grafo. In caso di presenza di un ciclo negativo nel grafo, l'algoritmo sarà in grado di rilevarlo.
Massimo numero di iterazioni = numero di nodi (da 0 a n).
Per le iterazioni successive controllo il nodo modificato e a quali nodi è collegato e verifico se il peso migliora (se migliora cambio altrimenti lascio il valore precedente).
Bellman-Ford si interrompe o quando raggiungo il numero massimo di iterazioni o quando compare per due volte di fila la stessa riga nella tabella.
Tabella di esempio:

Iterazione	**A**	**B**	**C**	**D**	**Aggiornati**
0	0_A	$+\infty$	$+\infty$	$+\infty$	A
1	0_A	5_A	$+\infty$	$+\infty$	A, B
...	...	...	...	...	...
4	0_A	-1_D	2_B	3_C	A, B, C, D

4.1.1 Grafo, Albero, Ciclo negativo

Se ho un ciclo negativo non è possibile fare né albero né grafo. Per trovare il ciclo negativo si osserva quale variabile viene modificata nell'ultima iterazione e si prosegue a ritroso, seguendo le indicazioni da dove arriva finché non si chiude il ciclo, oppure semplicemente quando, all'ultima iterazione di controllo, almeno un nodo viene aggiornato.
Per individuare l'albero si prendono i valori dell'ultima riga della tabella.
Per identificare il grafo si ricopia il grafo iniziale con anche i cammini minimi dei nodi, quindi si osservano i percorsi che sommati risultano lo stesso peso del nodo.

4.2 Dijkstra

L'algoritmo di Dijkstra per il problema del cammino minimo è un algoritmo di ricerca che ha una complessità computazionale di $O(n^2)$ o $O(m + n \log n)$, dove n è il numero di nodi e m è il numero di archi nel grafo. La complessità può essere migliorata utilizzando una coda di priorità per selezionare il prossimo nodo da visitare, riducendo la complessità a $O(m + n \log n)$. Questa complessità si basa sul fatto che l'algoritmo deve visitare tutti i nodi una volta e l'operazione di selezione del prossimo nodo richiede una ricerca nella coda di priorità.
Nella tabella di sviluppo S sono i nodi da verificare e $\hat{v}$ è il nodo che si sta verificando.
Dopo che si è verificato un nodo si mette un *, se non è controllato si lascia vuoto, se non migliora si mette un -, se il nodo è già chiuso e viene ricontrollato si mette una $\times$.

5 Branch and Bound (Teoria)

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	$[\text{min } LB\ figli;\ \text{min } SA^*\ generale]$	$[\text{max } SA^*\ generale;\ \text{max } UB\ figli]$
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.