Immagine che contiene tavolo

Descrizione generata automaticamente

Si considera l’introduzione di una variabile decisionale per il tipo di semola:

: numero di quintali di semola del tipo

: numero di quintali prodotti per il lotto

Vogliamo massimizzare la quantità di semola *prodotta* (non da produrre, quelli sono altri vincoli) e avremo:

Sappiamo che il mulino dispone di 10000 euro per le granaglie; queste sono valutate al quintale (dividere per 100):

Abbiamo i vincoli di produzione: quantità di semola da produrre quantità di semola a disposizione:

(produzione della semola dei due tipi <= disponibilità lotti di tipo 1)

(produzione della semola dei due tipi <= disponibilità lotti di tipo 2)

(produzione della semola dei due tipi <= disponibilità lotti di tipo 3)

Ora passiamo ai vincoli del problema:

* “si possono acquistare al massimo 5 unità di lotto 3”
* “la semola normale deve essere almeno il doppio della semola integrale e non più del quadruplo”
* “le granaglie del lotto 1 e 2 sono incompatibili e non possono essere contemporaneamente acquistate”

Introduciamo quindi una variabile logica binaria.

variabile logica che vale 1 se si effettua rifornimento dal fornitore di tipo per le granaglie del tipo , 0 altrimenti

L’attivazione è come segue:

Siccome non possiamo acquistare contemporaneamente, introduciamo i vincoli per fare in modo che l’acquisto dei due tipi di semola sia dei tipi N e I sia <= alla somma delle quantità prodotte dai lotti.

Ad esempio, consideriamo:

Questo vincolo garantisce che i grani del lotto 1 non possano essere acquistati contemporaneamente, perché se sono maggiori di 0, il lato sinistro della disuguaglianza sarebbe maggiore del lato destro, quindi sto effettivamente comprando tutti i tipi di semola.

Similmente per il lotto 2 con stesso ragionamento,

“l’impurità media delle scorte di granaglia di tipo A deve essere inferiore all’1.6%”

Domini:

Immagine che contiene tavolo

Descrizione generata automaticamente

Per risolvere questo problema, introduciamo due variabili che considerano come indici gli insiemi di rotte e di navi, nello specifico:

: numero di tonnellate per il tipo di nave per la rotta

: numero di navi per il tipo di nave per la rotta

È noto che vogliamo soddisfare la domanda sulle rotte minimizzando i costi e la funzione obiettivo assumerà la forma che segue:

Andiamo a modellare i vari altri vincoli, per esempio sul numero di navi:

(navi di tipo A)

(navi di tipo B)

(navi di tipo C)

Altri vincoli da inserire sono i seguenti:

* “sulla rotta 1 ci possono essere al massimo 10 navi di tipo A”;
* “sulla rotta 2 può effettuare servizio un solo tipo di nave”

Andremo ad introdurre un’apposita variabile binaria:

* : variabile binaria che vale 1 se utilizziamo navi del tipo per la rotta ,

altrimenti

A questa condizione, si introducono i vincoli di relazioni logica tra

* “se le navi di tipo B sono utilizzate sulla rotta 2, allora queste non possono essere utilizzate né sulla rotta 1, né sulla rotta 3”

A livello logico è un vincolo del tipo , considerando però le rotte:

Anche qui, per collegamento logico, vanno attivate le variabili logiche che non sono , in quanto è già attivata di per sé da come è scritto il vincolo (avendo che sarebbe 1).

Quindi, diventerebbe:

In ultimo, consideriamo i vincoli di capacità massima, quindi “ciascuna nave del tipo X può portare fino ad un massimo di tonnellate per quel tipo X”.

Per considerare valido quanto fatto fino ad ora, inseriamo tutti i singoli domini:

( e per ognuna delle tre variabili, messo una sola volta per sintesi)

(Similmente, fa parte dell’insieme reale in quanto rappresenta il numero di tonnellate)

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Attenzione che:

- la funzione obiettivo è di minimo e si cambia di segno

- Data , si va a cambiare segno a quando la si introduce poi, avendo i termini noti non negativi, si va a cambiare segno alle variabili di slack (e, se ci fossero altre variabili oltre a quelle di slack, si cambia segno anche a quelle)

Immagine che contiene tavolo

Descrizione generata automaticamenteImmagine che contiene tavolo

Descrizione generata automaticamentePer il pivot, siccome non siamo in forma canonica, scelgo di volta in volta un elemento utile per le operazioni di Gauss-Jordan. Faccio entrare in base :



Faccio poi entrare in base (ho evidenziato in rosso gli elementi di pivoting):

Immagine che contiene tavolo

Descrizione generata automaticamente



Si fa poi entrare in base facendo pivot sull’elemento in rosso, tale che otteniamo finalmente la forma canonica.

Immagine che contiene tavolo

Descrizione generata automaticamente

La base si compone sulle colonne dove appaiono i coefficienti della matrice identità, quindi

Si nota che è ammissibile (avendo tutte le colonne di > 0 ). Non sappiamo se sia ottima (avendo costi ridotti negativi) ma non è illimitata (infatti, tutti i coefficienti sono positivi sotto colonne con costi ridotti negativi).

Partiamo con il simplesso e decidiamo la variabile che entra in base. Per la regola di Bland, scegliamo la prima variabile in ordine tra quelle con coefficienti di costo ridotto negativo (quindi, tra scelgo ).

La variabile che esce dalla base si capisce rispetto al rapporto nella posizione della variabile che entra in base, quindi .

Per scegliamo come elemento di pivoting come visto nel tableau precedente.

Immagine che contiene tavolo

Descrizione generata automaticamente

* F.C.? Sì
* è ammissibile? Sì 🡪 tutti i sono (cioè, tutta l’ultima colonna di destra)
* è ottima? Non lo so 🡪 Esiste qualche coefficiente di costo ridotto (condizione sufficiente di ottimalità; se fossero tutti uguali a zero, allora sarebbe ottima)
* è illimitata? Non lo so 🡪 Vado a vedere se in corrispondenza di colonne con costi ridotti strettamente minori di 0 abbiamo sotto una colonna tutta negativa 🡪 Esiste almeno un valore strettamente positivo in corrispondenza di qualche valore negativo e quindi non possiamo concludere con certezza
* chi entra in base? 🡪
* chi esce dalla base 🡪

ed eseguo il pivoting rispetto all’elemento riquadrato poco fa.

Immagine che contiene tavolo

Descrizione generata automaticamente

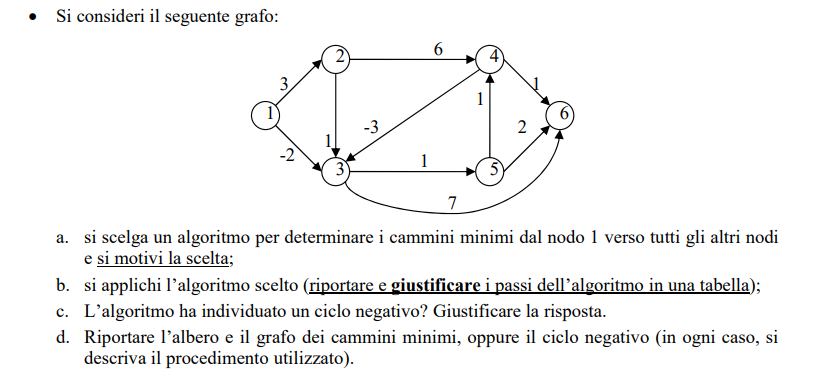
* F.C.? Sì
* è ammissibile? Sì 🡪 tutti i sono (cioè, tutta l’ultima colonna di destra)
* è ottima? Si 🡪 Non esiste qualche coefficiente di costo ridotto

La soluzione ottima del problema è

Inoltre, abbiamo

con vincoli saturo in quanto ha valore zero e lasco, per valore > 0 (si ricordi che lasco e saturo si va a

dire sulle variabili di slack aggiunte).



a) In questo grafo, utilizziamo Bellman-Ford, in quanto esistono archi con costi ridotti negativi.

b) Ora, il calcolo dei cammini minimi, riportati in tabella (controllando ad ogni passo miglioramenti rispetto alla riga precedente e segnando in colonna apposita gli aggiornamenti effettuati; i predecessori sono segnati come semplici pedici senza parentesi). Dato che non siamo limitati dai max-hop, andremo a creare la tabella iterando un numero pari di volte al numero di nodi (quindi, facendoli tutti):

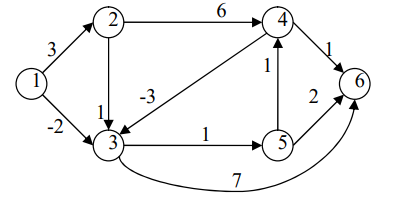
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo 1 | Nodo 2 | Nodo 3 | Nodo 4 | Nodo 5 | Nodo 6 | Aggiornamenti |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | // |
|  |  |  |  |  |  |  | // |
|  |  |  |  |  |  |  | // |



c) L’algoritmo non ha individuato un ciclo negativo in quanto non ha terminato con *flag\_aggiornato=true*, cioè non ha aggiornato fino alla fine delle iterazioni.

d) Per individuare l’albero dei cammini minimi, si percorre a ritroso la catena dei predecessori partendo dall’ultima iterazione, quindi, con costo . Si evidenzia in rosso il cammino minimo.

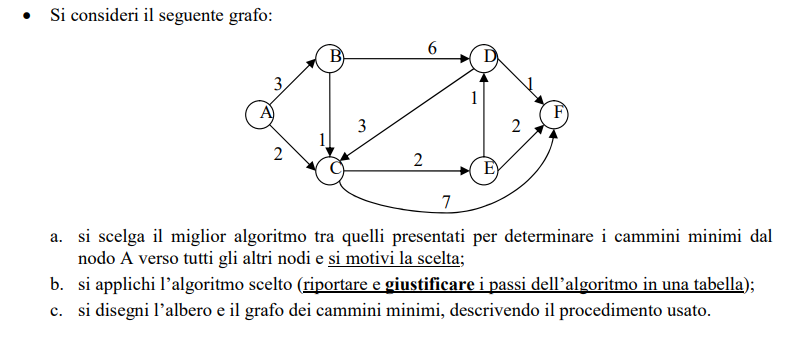
Per individuare il grafo dei cammini minimi, composto da *tutti* i cammini minimi, esamino il grafo e scrivendo le etichette ottime, verifico se esistono altri cammini minimi. In effetti, esistono facendo un altro percorso, passando per e per . Si ottiene sempre costo 1.



Legenda:

* Rosso – Albero
* Arcobaleno - Grafo





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:

* rappresenta l’etichetta minima di ogni iterazione
* 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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo | Nodo | Nodo | Nodo | Nodo | Nodo |  |  |
|  |  |  |  |  |  |  |  |  |
|  | \* |  |  |  |  |  |  |  |
|  |  | - | \* |  |  |  |  |  |
|  |  | \* |  |  | - |  |  |  |
|  |  |  |  | - | \* | - |  |  |
|  |  |  |  | \* |  | - |  |  |
|  |  |  |  |  |  | \* |  |  |

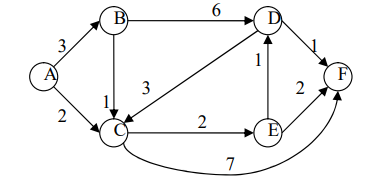
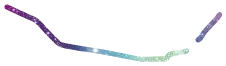
Ad ogni iterazione, percorriamo il grafo e scegliamo il percorso con costo minore. Ad ogni iterazioni, scegliamo e fissiamo un’etichetta che ha costo minore, scremando ad ogni iterazione quelle da controllare e avere sempre in mano il costo minimo. Anche in questo caso, come per gli algoritmi label correcting in caso di convergenza alla soluzione ottima, le etichette calcolate e i relativi puntatori rappresentano, rispettivamente, una soluzione duale ammissibile e i predecessori su dei cammini dall’origine ai diversi nodi. Questo funziona non avendo costi negativi.

c)

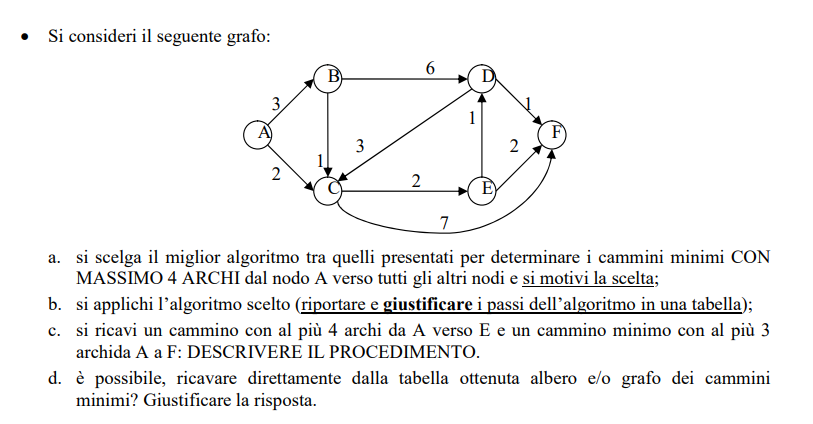
Pertanto, come abbiamo visto per gli algoritmi label correcting in caso di convergenza, è possibile derivare l’albero (risp. il grafo) dei cammini minimi attraverso i puntatori ai predecessori (risp. la verifica della saturazione dei vincoli duali sugli archi).

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 e scegliendo come cammino sia l’albero che tutte le altre etichette con costo )







1. 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
2. Ora, il calcolo dei cammini minimi, riportati in tabella (controllando ad ogni passo miglioramenti rispetto alla riga precedente e segnando in colonna apposita gli aggiornamenti effettuati; i predecessori sono segnati come semplici pedici senza parentesi):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo A | Nodo B | Nodo C | Nodo D | Nodo E | Nodo F | Aggiornamenti |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

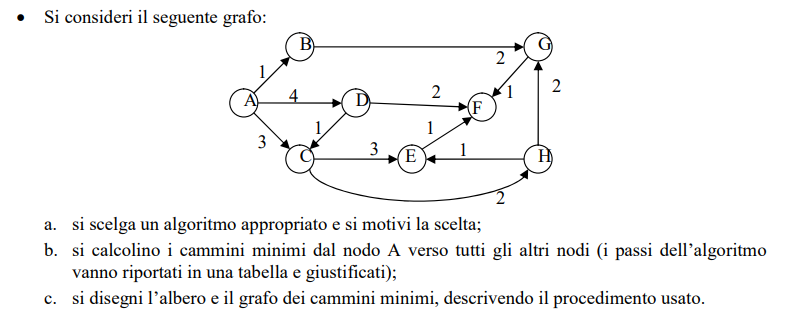
Le etichette di una riga sono ottenute controllando i vincoli duali su tutti gli archi uscenti dai nodi “aggiornati” della riga (iterazione) precedente secondo la dove è uno degli archi uscenti da un nodo aggiornato all’iterazione precedente, l’etichetta corrente (sulla riga corrente) del nodo , è l’etichetta del nodo all’iterazione (riga) precedente e è il costo dell’arco .

1. Un cammino minimo con al più archi da verso , seguendo la catena dei predecessori e considerano lo stesso , viene dato da con costo

Un cammino minimo con al più archi da verso , considerando lo stesso e seguendo la catena dei predecessori, viene dato da con costo . Usando solo i predecessori, non abbiamo un albero dei cammini minimi.

Per l’individuazione di entrambi i cammini minimi, seguo come detto i predecessori partendo dal nodo indicato e diminuendo di 1, considerando la diminuzione delle iterazioni ad ogni passo e guardando la riga precedente. Infatti, nel caso si risolva un problema di cammino minimo con un massimo numero di hop, non si può parlare di albero dei cammini minimi.

1. L’albero dei cammini minimi può essere ottenuto dalla tabella seguendo la catena dei predecessori (solo nel caso della prima richiesta), mentre per quanto riguarda il grafo, questo viene dato da tutti e soli i cammini minimi; in questo caso specifico, coincide con il percorso con al più 3 archi, in quanto quello da 4 non può essere considerato dati i motivi enunciati sopra.



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:

* rappresenta l’etichetta minima di ogni iterazione
* 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

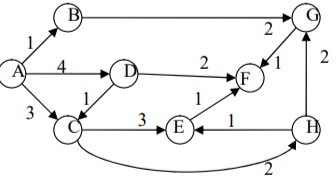
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo | Nodo | Nodo | Nodo | Nodo | Nodo | Nodo | Nodo |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | \* |  |  |  |  |  |  |  |  |  |
|  |  | \* |  |  |  |  |  |  |  |  |
|  |  |  | \* |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | \* |  |  |  |
|  |  |  |  |  |  | \* |  |  |  |  |
|  |  |  |  | \* |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | \* |  |  |  |  |  |

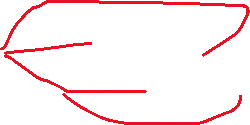
Ad ogni iterazione, percorriamo il grafo e scegliamo il percorso con costo minore. Ad ogni iterazioni, scegliamo e fissiamo un’etichetta che ha costo minore, scremando ad ogni iterazione quelle da controllare e avere sempre in mano il costo minimo. Anche in questo caso, come per gli algoritmi label correcting in caso di convergenza alla soluzione ottima, le etichette calcolate e i relativi puntatori rappresentano, rispettivamente, una soluzione duale ammissibile e i predecessori su dei cammini dall’origine ai diversi nodi. Questo funziona non avendo costi negativi.

In questo esercizio, si presentavano più etichette con lo stesso costo minimo; ho scelto in tutti i casi l’etichetta con indice inferiore.

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).

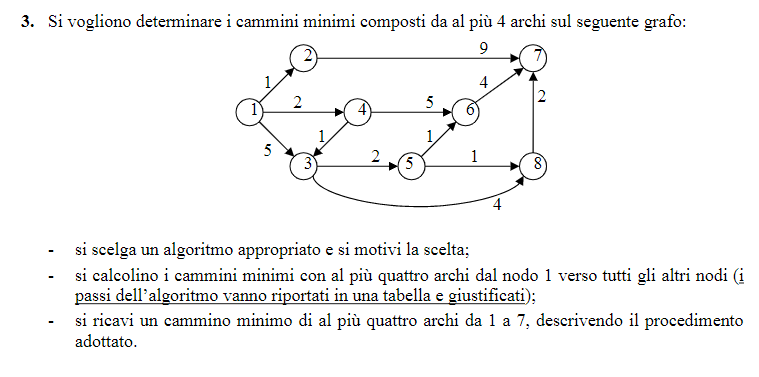




Concretamente, avremo che:

* 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 ); in questo caso, grafo ed albero sono diversi.





1. 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
2. Ora, il calcolo dei cammini minimi, riportati in tabella (controllando ad ogni passo miglioramenti rispetto alla riga precedente e segnando in colonna apposita gli aggiornamenti effettuati; i predecessori sono segnati come semplici pedici senza parentesi):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterazione | Nodo 1 | Nodo 2 | Nodo 3 | Nodo 4 | Nodo 5 | Nodo 6 | Nodo 7 | Nodo 8 | Aggiornamenti |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Le etichette di una riga sono ottenute controllando i vincoli duali su tutti gli archi uscenti dai nodi “aggiornati” della riga (iterazione) precedente secondo la dove è uno degli archi uscenti da un nodo aggiornato all’iterazione precedente, l’etichetta corrente (sulla riga corrente) del nodo , è l’etichetta del nodo all’iterazione (riga) precedente e è il costo dell’arco .

Grazie al fatto di utilizzare l’etichetta del nodo precedente, viene assicurata la scelta del cammino minimo con archi (qua si può fare un qualsiasi esempio numerico dove si va a scegliere, prendendo ad esempio l’ultima iterazione, qui un cammino che ha costo migliore considerando l’etichetta precedente e non quella corrente, qui , dimostrando la validità di quanto fatto).

1. Un cammino minimo con al massimo archi da viene fatta seguendo la catena dei predecessori, quindi, partendo dal nodo (ultimo nodo in generale) con si considera quindi:

Questo equivale al percorso , con costo 9.

Immagine che contiene testo

Descrizione generata automaticamente

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.

Immagine che contiene testo

Descrizione generata automaticamente



1) Verifica dell’ammissibilità primale della soluzione data

2) Passaggio al duale

3) Applicazione delle condizioni di complementarietà primale-duale

Primo vincolo primale: 🡪 🡪 (prima condizione)

Secondo vincolo primale: 🡪 🡪 // (non si deducono condizioni di complementarietà su )

Terzo vincolo primale: 🡪 🡪 (seconda condizione)

Quarto vincolo primale di uguaglianza, pertanto non ci sono da imporre condizioni di complementarietà con la relativa variabile

Primo vincolo duale di uguaglianza: non si impongono condizioni di complementarietà con (in quanto la condizione è diretta conseguenza dell’ammissibilità duale; l’equazione sarà comunque da considerare come condizione di ammissibilità duale

Secondo vincolo duale: (terza condizione)

Terzo vincolo duale: 🡪//

4) Sistema delle condizioni CCPD e ammissibilità duale trovate

(ccpd)

(ccpd)

(ammissibilità duale)

(ammissibilità duale)

5) Verifica ammissibilità duale

La soluzione duale trovata:

* Soddisfa i tre vincoli duali:
* Soddisfa i vincoli di dominio:

6) Conclusioni

Abbiamo a disposizione una soluzione primale e una soluzione duale tali che:

• è ammissibile primale (come da verifica);

• è ammissibile duale (come da costruzione e da verifica);

• e sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale

(Per verifica, i valori delle funzioni obiettivo sono uguali, infatti valgono entrambe 4 e verifica il corollario della dualità forte)

Immagine che contiene testo

Descrizione generata automaticamente

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.

Immagine che contiene testo

Descrizione generata automaticamente



1) Verifica dell’ammissibilità primale della soluzione data

2) Passaggio al problema duale

3) Applicazione delle condizioni di complementarietà primale-duale

Primo vincolo primale: 🡪 🡪 (prima condizione)

Secondo vincolo primale: È vincolo di uguaglianza, non ci sono da imporre condizioni con la variabile duale , visto che la condizione deriva dall’ammissibilità primale

Terzo vincolo primale: 🡪 🡪 // (non si possono dedurre condizioni)

Primo vincolo duale: 🡪 🡪 (seconda condizione)

Secondo vincolo duale: 🡪 🡪 (terza condizione

Terzo vincolo duale: 🡪 🡪 //

4) Sistema delle condizioni CCPD e ammissibilità duale trovate

(ccpd)

(ammissibilità duale)

(ammissibilità duale)

Risolvendo 🡪

5) Verifica ammissibilità duale

La soluzione duale trovata:

soddisfa i vincoli duali ()

soddisfa i vincoli di dominio (

6) Conclusioni

Abbiamo a disposizione una soluzione primale e una soluzione duale tali che:

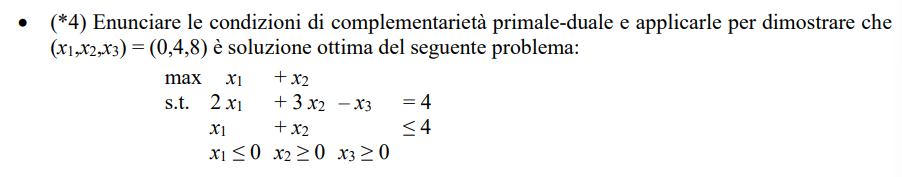
• è ammissibile primale (come da verifica);

• è ammissibile duale (come da costruzione e da verifica);

• e sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale

(Per verifica, i valori delle funzioni obiettivo sono uguali, infatti valgono entrambe e verifica il corollario della dualità forte)



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.

Immagine che contiene testo

Descrizione generata automaticamente



1) Verifica dell’ammissibilità primale della soluzione data

2) Passaggio al duale

3) Applicazione delle condizioni di complementarietà primale-duale

- il primo vincolo primale è di uguaglianza; non ci sono da imporre condizioni con la relativa variabile duale (la condizione è diretta conseguenza dall’ammissibilità primale

- il secondo vincolo primale 🡪 🡪 (non è possibile dedurre alcuna condizione di complementarietà su )

- il primo vincolo duale 🡪 🡪 0 🡪 //

- il secondo vincolo duale 🡪 🡪 (prima condizione)

- il terzo vincolo duale è di uguaglianza: non ci sono condizioni da imporre sulla relative variabile primale (tuttavia, la condizione sarà comunque da considerare come condizione di ammissibilità duale) 🡪 (seconda condizione)

4) Sistema di equazioni per l’imposizione delle condizioni di complementarietà primale duale (ccpd) trovate e delle condizioni di ammissibilità duale

1]

2]

5) Verifica ammissibilità duale

La soluzione duale trovata:

* soddisfa i tre vincoli duali ()
* soddisfa i vincoli di dominio ( libera, )

6) Conclusioni

Abbiamo a disposizione una soluzione primale e una soluzione duale tali che:

• è ammissibile primale (come da verifica);

• è ammissibile duale (come da costruzione e da verifica);

• e sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale

(Per verifica, i valori delle funzioni obiettivo sono uguali, infatti valgono entrambe 4 e verifica il corollario della dualità forte)

Immagine che contiene tavolo

Descrizione generata automaticamente

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. In particolare, valgono le seguenti condizioni di ottimalità:

Immagine che contiene testo

Descrizione generata automaticamente



1) Verifica dell’ammissibilità primale della soluzione dat

2) Passaggio al duale:

8

3) Applicazione delle condizioni CCPD:

non si può dire nulla

non si può dire nulla

🡪 non ci sono da imporre condizioni di complementarietà con la relativa variabile duale , in quanto la condizione è diretta conseguenza dell’ammissibilità primale

🡪 (prima condizione)

(seconda condizione)

🡪 Il vincolo duale è di uguaglianza, non ci sono condizioni da imporre con la relativa variabile , in quanto la condizione è di diretta conseguenza dell’ammissibilità duale: comunque è da considerarsi come terza condizione

(quarta condizione)

4) Sistema di equazioni per l’imposizione delle condizioni di complementarietà primale duale (ccpd) trovate e delle condizioni di ammissibilità duale

(ccpd)

(ammissibilità duale)

(ammissibilità duale)

(ammissibilità duale)

5) Verifica ammissibilità duale

La soluzione duale trovata:

* soddisfa i tre vincoli duali (
* soddisfa i vincoli di dominio (

6) Conclusioni

Abbiamo a disposizione una soluzione primale e una soluzione duale tali che:

• è ammissibile primale (come da verifica);

• è ammissibile duale (come da costruzione e da verifica);

• e sono in scarti complementari (per costruzione).

Pertanto, le due soluzioni sono ottime per i rispettivi problemi primale e duale

Svolgimento del prof

1. Dati due problemi generici *mix* e *max* nelle forme:

Con

1) Verifica dell’ammissibilità primale della soluzione data

X è ammissibile primale?

2) Passaggio al problema duale

3) Condizioni di complementarietà primale-duale

🡪 Vero per ammissibilità duale ( per il fatto delle condizioni di complementarietà, non tanto perché

//

//

🡪 Vero per ammissibilità primale

4) Sistema di equazioni per l’imposizione delle condizioni di complementarietà primale duale (ccpd) trovate e delle condizioni di ammissibilità duale:

(CCPD)

(ammissibilità duale)

(CCPD)

(CCPD

5) Verifica ammissibilità duale

Primo vincolo duale a posto per costruzione, secondo vincolo duale a posto per costruzione, terzo vincolo duale a posto per costruzione

6) Conclusioni

La è ammissibile duale ed è in scarti complementare con la data ottime

* Come si riconoscono sul tableau del simplesso le condizioni di illimitatezza per un problema di minimo? Giustificare la risposta.
  + Le condizioni di illimitatezza sono date dall’avere un’intera colonna che posto un costo ridotto positivo costituita di soli coefficienti negativi
* Si enunci e si giustifichi la regola adottata dal metodo del simplesso per la selezione della variabile uscente nelle operazioni di cambio base.
  + Per la selezione della variabile uscente, normalmente all’interno della matrice aumentata su cui eseguiamo le operazioni (nel nostro caso, il tableau del simplesso) si sceglie come riga la variabile che entra e si esegue il rapporto minimo tra ogni coefficiente e tutti i coefficienti
* Si discuta la complessità computazionale dell’algoritmo di Bellman-Ford.
  + L’algoritmo di Bellman-Ford ha una complessità ), in quanto viene fato un numero di iterazioni pari al numero di nodi con un ciclo che esamina le etichette duali su tutti gli archi . Qualora esistesse un ciclo negativo (abbiamo aggiornato almeno un’etichetta) oppure tutte le etichette sono stabili, l’algoritmo termina la sua esecuzione.
* Si discuta la complessità computazionale dell’algoritmo di Dijkstra per il problema del cammino minimo
  + Il prof riporta: “Si consiglia di riportare i passi dell’algoritmo di Dijkstra e, quindi, discuterne la complessità come fatto nelle dimostrazione della Proprietà 10 alla fine del paragrafo 6.1 delle dispense sugli algoritmi per problemi di cammino minimo. Quindi si può approfondire con qualche dettaglio sull’influenza della scelta di opportune strutture dati per migliorare l’efficienza, come dal paragrafo 6.2 delle stesse dispense.”
* Si discuta la complessità computazionale dell’algoritmo del simplesso.
  + Essendo un algoritmo iterativo che cambia base e vertice quando la base è degenere, tende a convergere in ; quando si ha a che fare con basi degeneri, non vi è nessuna garanzia di convergenza. Grazie anche alla regola di Bland, la complessità si assesta numericamente a quanto descritto, diventando tuttavia più efficiente, ottenendo complessità media lineare/sublineare in casi più fortunati.

Immagine che contiene testo

Descrizione generata automaticamente



Basi ottime:

* 🡪 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

Immagine che contiene tavolo

Descrizione generata automaticamente

a) La possibile soluzione di base è , dati gli 1 della matrice identità. Inoltre, non sappiamo al momento se sia ottima, in quanto esistono due costi ridotti non positivi; al momento, non si sa per certo se possa essere illimitata

b) I possibili cambi base sono 2: ed , in quanto hanno due coefficienti negativi

c) Seguendo la regola di Bland, il cambio di base sarà su e si andrà a fare il rapporto minimo tra e la colonna di , selezionando l’argomento minimo

d) Usando la regola di Bland, andiamo a selezionare ; di fatto, la funzione obiettivo non migliora, in quanto il minimo rapporto è , quindi la funzione obiettivo è

e) Anche se cambia la base corrente, non cambia il vertice del poliedro associato alla nuova base. Infatti, i nuovi valori delle variabili in base sono

Si passa quindi dalla soluzione di base degenere: (con e fuori base) alla soluzione di base sempre degenere (con entrante e uscente, visto il rapporto minimo precedente e sfruttando la regola di Bland), , che rappresenta lo stesso vertice del poliedro ammissibile.

Immagine che contiene tavolo

Descrizione generata automaticamente

a)

Immagine che contiene testo, diverso, orologio

Descrizione generata automaticamente



Per Bland, sceglieremo come variabile entrante . Poi, eseguiremo il pivot, sempre per Bland, sulle variabili che hanno rapporto minimo, quindi in questo caso specifico, su 75 e 13, in quanto hanno entrambe rapporto 0. Come tale, si evidenzia in tabella questa scelta.

b)

Di fatto, si ha un’iterazione degenere, in quanto il rapporto minimo è , e quindi il miglioramento atteso sarebbe dato da

c)

Anche se cambia la base corrente, non cambia il vertice del poliedro associato alla nuova base. Infatti, i nuovi valori delle variabili in base sono . Passiamo quindi dalla soluzione di base degenere (con e fuori base) alla soluzione di base sempre degenere (variabile questa che entra in base per la regola di Bland ma degenere come notato poco fa), (con e fuori base), che rappresenta lo stesso vertice del poliedro ammissibile.

* Enunciare e giustificare le condizioni di ottimalità nel metodo del simplesso
  + Il problema di PL deve essere in forma standard, quindi con funzione di minimo e una base ammissibile di partenza, cioè con tutti positivi
  + Deve essere in forma canonica rispetto alla base corrente B, quindi avere le colonne anche sparse con coefficienti dalla matrice identità
  + La soluzione è ottima nel momento in cui tutti i costi ridotti delle variabili fuori base sono positivi o nulli; sarà anche tale se non vi sono colonne di costo ridotto della f.o. con tutti coefficienti negativi, in tal caso sarà illimitata
  + Fatte tutte queste premesse, viene definita ottima in quanto il valore della f.o. migliora (o, in altri casi, non peggiora)