# Esercizio su cammino di costo minimo

Calcolare mediante il metodo di Djikstra un cammino da *s* a *t* di costo minimo nel seguente grafo orientato, in cui per ogni arco (*i*, *j*) è indicato il costo associato *qij*.



## Riferimento: M. Fischetti, “Lezioni di Ricerca Operativa” pag. 139-144, oppure

## R. Baldacci, M. Dell’Amico, “Fondamenti di Ricerca Operativa” pag. 129-132.

# Soluzione

Il problema è determinare un cammino di costo minimo da *s* a *t*, dove il costo di un cammino è dato dalla somma dei costi associati agli archi del cammino (ad esempio, il cammino *s* → *a* → *d* → *t*  ha costo 4 + 1 + 7 = 12). Il metodo di soluzione è basato sul seguente risultato:

#### Teorema. Sia W un insieme di vertici w per cui è noto il costo Lw di un cammino di costo minimo da s a w (in particolare, s ∈ W). Sia (w\*, v\*) = argmin { Lw + qwv : w ∈ W, v ∉ W}. Allora Lw\* + qw\*v\* è il costo di un cammino di costo minimo da s a v\* (cioè v\* può essere inserito in W).

Il metodo costruisce iterativamente tale insieme *W*, fino ad inserire *t* in *W*.

### Inizializzazione

*W* :*=* {*s*}. Costruisci il vettore *z*, che ha una componente per ogni vertice *v* del grafo, con *z*(*v*) uguale: a *qsv* se esiste l’arco *sv*, a ∞ altrimenti.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | *a* | *b* | *c* | *d* | *e* | *t* |
| **0** | 4 | 2 | ∞ | ∞ | ∞ | ∞ |

### Passo generico

• Scegli vertice *x* non in *W* tale che *z*(*x*) è minimo

• Inserisci *x* in *W*

• Aggiorna *z* ponendo *z*(*v*) := min {*z*(*v*), *z*(*x*) + *qxv*} per ogni vertice *v* non in *W*

### Passo 1

• Scegli *b*

• *W* :*= W* ∪ {*b*}

• Aggiorna *z* come di seguito

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | *a* | *b* | *c* | *d* | *e* | *t* |
| **0** | 4 | **2** | ∞ | 3 | 11 | ∞ |

### Passo 2

• Scegli *d*

• *W* :*= W* ∪ {*d*}

• Aggiorna *z* come di seguito

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | *a* | *b* | *c* | *d* | *e* | *t* |
| **0** | 4 | **2** | 6 | **3** | 11 | 10 |

### Passo 3

• Scegli *a*

• *W* :*= W* ∪ {*a*}

• Aggiorna *z* come di seguito

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | *a* | *b* | *c* | *d* | *e* | *t* |
| **0** | **4** | **2** | 6 | **3** | 11 | 10 |

### Passo 4

• Scegli *c*

• *W* :*= W* ∪ {*c*}

• Aggiorna *z* come di seguito

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | *a* | *b* | *c* | *d* | *e* | *t* |
| **0** | **4** | **2** | **6** | **3** | 11 | 8 |

### Passo 5

• Scegli *t*

• *W* :*= W* ∪ {*t*}

• Il metodo termina dato che *t* è stato inserito in *W*

Il numero *z*(*t*) = 8 indica il costo di un cammino di costo minimo da *s* a *t*. Per individuare tale cammino, si può procedere andando a ritroso da *t* verso *s* in questo modo:

*z*(*t*) è stato aggiornato l’ultima volta quando *c* è entrato in *W*;

*z*(*c*) è stato aggiornato l’ultima volta quando *d* è entrato in *W*;

*z*(*d*) è stato aggiornato l’ultima volta quando *b* è entrato in *W*;

*z*(*b*) è stato aggiornato l’ultima volta quando *s* è entrato in *W*.

Si ottiene il cammino: *s* → *b* → *d* → *c* → *t* (con costo = 8)

##### Esercizio su programmazione di progetti

## Calcolare mediante il metodo PERT il tempo minimo di completamento, le attività critiche e i massimi slittamenti ammessi (per ogni attività) nel seguente progetto:

## le attività sono 5: A1, …, A5; ogni attività Ai ha durata di, dove d = [5, 7, 2, 4, 3];

## le relazioni di precedenza sono: A1 < A4; A2 < A3; A3 < A4; A3 < A5.

*Riferimento*: M. Fischetti, “Lezioni di Ricerca Operativa” pag. 147-152.

# Soluzione

## • Costruisci un grafo orientato G associando ad ogni attività un arco e introducendo le precedenze con archi fittizi.

## • Costruisci un grafo G’ modificando il grafo G come segue: (i) elimina gli archi fittizi che sono obsoleti – tale operazione non deve né togliere né aggiungere precedenze; (ii) aggiungi un vertice s e gli archi fittizi sv per ogni vertice v di G in cui non entra alcun arco; (iii) aggiungi un vertice t e gli archi fittizi vt per ogni vertice v di G da cui non esce alcun arco; (iv) associa ad ogni attività Ai il valore di (per i = 1, . . ., 5); (v) associa ad ogni arco fittizio il valore 0 (non riportato in figura).

*Osservazione*: i valori *di* delle attività e i valori 0 degli archi fittizi possono essere visti come “costi” *qij* associati agli archi di *G’*; si osservi che, in tal modo, il costo di un cammino di costo massimo da *s* a *t* in *G’* è pari al tempo minimo di completamento del progetto.



• Ordina topologicamente gli archi di *G’*, cioè associa una numerazione progressiva ai nodi in modo che se il numero associato a un nodo *x* è minore di quello associato a un nodo *y* allora non esiste un cammino in *G’* da *y* a *x*.



## • Associa ad ogni vertice una doppia etichetta [Tmin , Tmax], cioè per ogni vertice h di G’ calcola due valori Tmin (h) e Tmax (h) come segue: (i) Tmin(1) = 0; (ii) per h = 2, …, 9, poni Tmin(h) = max {Tmin(i) + qih : esiste arco (i, h)}; (iii) Tmax(9) = Tmin(9); (iv) per h = 8, …, 1, poni Tmax(h) = min {Tmax(i) + qhi : esiste arco (h, i)}.



Il tempo minimo di completamento del progetto è dato da Tmin(9) = 13.

Le attività critiche (cioè gli archi (*i*, *j*) tali che: Tmin(*i*) = Tmax(*i*); Tmin(*j*) = Tmax(*j*); Tmax(*j*) - Tmax(*i*) = *qij*) sono *A*2, *A*3, *A*4. Tali attività sono quelle il cui eventuale ritardo di attuazione genera direttamente un ritardo sul tempo minimo di completamento del progetto. Esse individuano almeno un *cammino* *critico* (insieme con archi fittizi corrispondenti ad attività critiche fittizie) come evidenziato in figura sottostante.



### **Esercizio su massimo flusso**

Calcolare mediante l’algoritmo di Ford-Fulkerson un flusso da *s* a *t* di valore massimo nella seguente rete, in cui accanto ad ogni arco è indicata la corrispondente capacità.



*Riferimento*: R. Baldacci, M. Dell’Amico, “Fondamenti di Ricerca Operativa” pag. 89-93.

## **Soluzione**

Applichiamo l’algoritmo di Ford-Fulkerson, nelle pagine seguenti.

## Inizializzazione

• poni pari a 0 ogni componente del flusso su ogni arco



• costruisci rete incrementale rispetto al flusso



• individua cammino da *s* a *t* nella rete incrementale, e saturalo; un cammino è *s*, *a*, *c*, *t*; lo si satura “inviando” più flusso possibile, cioè 2 (l’arco *ac* è collo di bottiglia)

******

Iterazione 1

• aggiorna il flusso (in base al cammino aumentante trovato)



• costruisci la rete incrementale relativa al flusso



• individua cammino da *s* a *t* nella rete incrementale, e saturalo; un cammino è *s*, *b*, *d*, *t*; lo si satura “inviando” più flusso possibile, cioè 3 (l’arco *dt* è collo di bottiglia)



Iterazione 2

• aggiorna il flusso (in base al cammino aumentante trovato)



• costruisci la rete incrementale relativa al flusso



• individua cammino da *s* a *t* nella rete incrementale, e saturalo; un cammino è *s*, *b*, *d*, *c*, *t*; lo si satura “inviando” più flusso possibile, cioè 2 (l’arco *sb* è collo di bottiglia)

****

Iterazione 3

• aggiorna il flusso (in base al cammino aumentante trovato)



• costruisci la rete incrementale relativa al flusso



• individua cammino da *s* a *t* nella rete incrementale, e saturalo.

in questo caso non esiste alcun cammino da *s* a *t* nella rete incrementale; quindi il flusso corrente (cioè quello relativo all’ultimo aggiornamento – nella figura sotto) è ottimo; il suo valore è dato dalla somma delle componenti del flusso relative agli archi uscenti da *s*, cioè, 2 + 5 = 7.



##### Esercizio su programmazione della produzione

Un bene deve essere prodotto su un orizzonte temporale composto da 4 periodi. Per ogni periodo *t* = 1,…, 4, è nota la domanda *dt* da soddisfare nel periodo *t*. All’inizio del periodo 1 non ci sono giacenze in magazzino e non si vuole che ce ne siano alla fine del periodo 4.

Applicando il metodo di Wagner-Whitin, determinare per *t* = 1,…, 4, la quantità *xt* prodotte nel periodo *t* e le quantità *st* in giacenza nel periodo *t*, in modo da minimizzare i costi totali e da soddisfare le domande al momento in cui vengono effettuate, sapendo che la funzione costo di produzione è *Ct* = *Atw*(*xt*) + *ctxt*, dove: *w*(*xt*) = 1 se *xt* > 0, e *w*(*xt*) = 0 altrimenti; la funzione costo di stoccaggio è *Ht* = *htst*; inoltre in tabella sono riportati i dati relativi a quanto sopra.

Periodo Domanda (*d*) *A* *c* *h*

1 30 40 2 1

2 50 20 3 2

3 50 10 4 2

4 30 20 2

*Riferimento*: A. Sassano, “Modelli e algoritmi della Ricerca Operativa” pag. 327-340.

# Soluzione

# La situazione può essere schematizzata come nella figura di seguito, in cui ad ogni periodo *i* sono associate le quantità *di* di domanda, *xi* di produzione e *si* di stoccaggio (ciò che avanza a fine periodo). Quindi, note le quantità *di* e le funzioni costo, il problema è determinare le quantità *xi* per ogni periodo *i* = 1, …, 4 e le quantità *si* per ogni periodo *i* = 1, …, 3.



Fase 1 : calcola i valori *M*(*i*, *j*) per ogni coppia di periodi (*i*, *j*) con *i* < *j*

Il valore *M*(*i*, *j*) è il costo totale che si dovrebbe pagare per soddisfare le domande dei periodi *i*, *i+*1, …, *j*, producendo solo nel periodo *i* (partendo da giacenza nulla e lasciando giacenza nulla).

*M*(1,1) = 40 + 2 · 30 = 100

*M*(1,2) = 40 + 2 · (30+50) + 1 · 50 = 250

*M*(1,3) = 40 + 2 · (30+50+50) + 1 · (50+50) + 2 · 50 = 500

*M*(1,4) = 40 + 2 · (30+50+50+30) + 1 · (50+50+30) + 2 · (50+30) + 2 · 30 = 710

*M*(2,2) = 20 + 3 · 50 = 170

*M*(2,3) = 20 + 3 · (50+50) + 2 · 50 = 420

*M*(2,4) = 20 + 3 · (50+50+30) + 2 · (50+30) + 2 · 30 = 630

*M*(3,3) = 10 + 4 · 50 = 210

*M*(3,4) = 10 + 4 · (50+30) + 2 · 30 = 390

*M*(4,4) = 20 + 2 · 30 = 80

Fase 2 : calcola ricorsivamente *Fk* per *k =* 1, …, 4

Il valore *Fk* è il valore di una soluzione ottima del problema ristretto ai primi *k* periodi. Quindi *F*4 è il valore di una soluzione ottima cercata. Il calcolo avviene ricorsivamente, ponendo *F*0 = 0 e applicando la formula *Fk =* min 1< *j* < *k* {*Fj-*1 + *M*(*j*, *k*)}.

*F*0 = 0

*F*1 = min {*F*0 + *M*(1,1)} = 100

*F*2 = min {*F*0 + *M*(1,2), *F*1 + *M*(2,2)} = {250, 270} = 250

*F*3 = min {*F*0 + *M*(1,3), *F*1 + *M*(2,3), *F*2 + *M*(3,3)} = {500, 520, 460} = 460

*F*4 = min {*F*0 + *M*(1,4), *F*1 + *M*(2,4), *F*2 + *M*(3,4), *F*3 + *M*(4,4)} = {710, 730, 640, 540} = 540

Fase 3 : evidenzia la soluzione ottima trovata

Per fare ciò si parte da *F*4 andando a ritroso, sostituendo (in base ai calcoli appena svolti) iterativamente i valori *Fk* fino ad arrivare a *F*0, come di seguito:

*F*4 = *F*3 + *M*(4,4) = *F*2 + *M*(3,3) + *M*(4,4) = *F*0 + *M*(1,2) + *M*(3,3) + *M*(4,4).

Si ottiene che la soluzione ottima è individuata da *M*(1,2) + *M*(3,3) + *M*(4,4), cioè si produce: nel periodo 1 (soddisfacendo le domande dei periodi 1, 2), nel periodo 3 (soddisfacendo le domande del periodo 3), nel periodo 4 (soddisfacendo le domande del periodo 4). I valori ottimi di *x*1, …, *x*4 e *s*1, …, *s*3 possono quindi essere dedotti dal loro significato e dalle domande, come indicato in figura.



**Esercizio su localizzazione di impianti**

Bisogna attivare un certo numero di impianti, da scegliere fra *m* potenziali impianti, che dovranno rifornire *n* clienti. Attivare il potenziale impianto *j* costa *fj*. Ogni impianto si intende abbia capacità di rifornimento illimitata. Ogni cliente sarà collegato ad un unico impianto; in particolare collegare un cliente *i* ad un impianto *j* comporta un costo *cij*. Problema: scegliere quali impianti attivare in modo da minimizzare i costi complessivi di attivazione e di assegnazione ai clienti.

Calcolare una soluzione "greedy" di tale problema per la seguente istanza:

matrice *cij*

impianti

1 2 3 4 5

1 0 1 7 0 4

2 1 0 3 0 3

3 3 7 1 4 2

clienti 4 1 7 4 0 7

5 1 0 3 2 4

6 4 4 1 4 3

7 7 1 2 1 0

vettore *f* 7 9 5 4 7

*Riferimento*: A. Sassano, “Modelli e algoritmi della Ricerca Operativa” pag. 269-274.

#### **Soluzione**

Inizializzazione

*i*  = 1; *T*0 = ∅; *Z*(*T*0) = ∞.

Iterazione 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Z*({1}) | *Z*({2}) | *Z*({3}) | *Z*({4}) | *Z*({5}) |
| 24 | 29 | 26 | 15 | 30 |

Ad esempio *Z*({1}) = (0 + 1 + 3 + 1 + 1 + 4 + 7) + 7 = 24.

Il minimo è *Z*({4}) = 15;

15 < ∞ ⇒ *i* = 2, *T*1 = {4}.

Iterazione 2

|  |  |  |  |
| --- | --- | --- | --- |
| *Z*({4,1}) | *Z*({4,2}) | *Z*({4,3}) | *Z*({4,5}) |
| 20 | 22 | 14 | 18 |

Ad esempio *Z*({4,1}) = (0 + 0 + 3 + 0 + 1 + 4 + 1) + (7 + 4) = 20.

Il minimo è *Z*({4,3}) = 14;

14 < 15 ⇒ *i* = 3, *T*2 = {4,3}.

Iterazione 3

|  |  |  |
| --- | --- | --- |
| *Z*({4,3,1}) | *Z*({4,3,2}) | *Z*({4,3,5}) |
| 22 | 21 | 20 |

Ad esempio *Z*({4,3,1}) = (0 + 0 + 1 + 0 + 1 + 1 + 1) + (7 + 5 + 4) = 22.

Il minimo è *Z*({4,3,5}) = 20;

20 > 14 ⇒ stop

La soluzione greedy è *T*2 = {4,3}, con *Z*({4,3}) = 14.