

Capitolo 7

Cammini minimi

Come abbiamo accennato nel capitolo precedente, il linguaggio dei grafi permette di rappresentare in modo semplice la struttura di molti problemi applicativi, consentendo, in molti casi di grande importanza, di costruire metodi razionali di soluzione dei problemi stessi.

Fra i problemi più importanti, più semplici e più antichi, per la cui soluzione sono utilizzate rappresentazioni basate su grafi, vi sono i problemi di ricerca di cammini minimi, di cui ci occupiamo in questo capitolo.

I grafi considerati in questo capitolo sono, salvo diversa specificazione, grafi orientati. Nel caso della ricerca di cammini infatti è sempre possibile ricondursi con poca fatica a questo caso. Per fare questo è sufficiente sostituire ogni arco non orientato (e quindi percorribile in entrambi i versi) con due archi diretti in direzione opposta. Se esiste un cammino con le caratteristiche richieste nel grafo originario, allora esiste anche nel grafo trasformato e viceversa.

Per brevità, infine, trattando solo di grafi orientati, *indichiamo con grafo aciclico un grafo che non contenga cicli orientati.*

7.1 Il problema del cammino minimo e alcuni esempi di applicazioni

Dato un grafo orientato $G = (V, E)$, associamo a ciascun arco $e = (u, v) \in E$ un peso $p(u, v) \in \mathbb{R}$. Per ogni cammino **orientato** $P = \{v_1, e_1, \dots, e_{p-1}, v_p\}$, definiamo peso $p(P)$ del cammino P la somma dei pesi degli archi che appartengono a P , e cioè: $p(P) = \sum_{(u,v) \in P} p(u, v)$. Il problema del cammino minimo può essere enunciato nel modo seguente:

dati due nodi $s \in V$ e $t \in V$, trovare un cammino orientato
 P^* in G da s a t che abbia peso minimo.

Notiamo che:

- Se non esiste un cammino orientato che vada da s a t in G , il problema non ha soluzioni ammissibili.
- Se esiste un ciclo orientato C in G , tale che $p(C) < 0$ (peso negativo), il problema è illimitato inferiormente.

In genere, dato un nodo s gli algoritmi per il calcolo di cammini minimi determinano non il (o uno dei) cammini minimi da s a un fissato nodo t , ma il cosiddetto *albero dei cammini minimi*, cioè un sottografo di G che è un albero i cui nodi includono s e tutti i nodi da esso raggiungibili con un cammino orientato e tale che l'unico (vedi teorema 6.1.1) cammino da s a ogni altro nodo t dell'albero sia un cammino minimo da s a t . A prima vista questa può sembrare una complicazione non necessaria, ma in effetti, per come

sono organizzati gli algoritmi, vedremo che non è così. Quindi, in pratica, gli algoritmi che studieremo risolvono il seguente problema:

dato un nodo $s \in V$, trovare un albero dei cammini minimi
da s a ogni nodo in V raggiungibile da s .

Nella letteratura esistono moltissimi algoritmi per il calcolo dei cammini minimi. Ognuno di essi ha le sue peculiarità e le sue limitazioni. Per esempio alcuni algoritmi funzionano solo su determinate classi di grafi (grafi aciclici, grafi con pesi non negativi, etc.), inoltre ci possono essere notevoli differenze nella efficienza degli algoritmi. In generale un algoritmo che funziona su una classe ampia di grafi sarà più complesso di un algoritmo studiato per una classe ristretta di grafi, tenendo conto delle peculiarità di quella classe.

In questo capitolo considereremo due algoritmi: il primo calcola i cammini minimi su grafi aciclici, il secondo calcola i cammini minimi su grafi con pesi non negativi.

Nella parte rimanente di questa sezione illustriamo alcuni semplici esempi di calcolo dei cammini minimi. Alla fine del capitolo considereremo due casi più complessi e interessanti.

7.1.1 Percorso di tempo minimo su una rete stradale

Dato un grafo pesato che rappresenta la rete stradale italiana in cui i pesi degli archi indicano il (valore atteso del) tempo di percorrenza dell'arco, il problema è quello di trovare il cammino che congiunge due particolari nodi del grafo (nodo di partenza e nodo di arrivo) con tempo di percorrenza minimo. Si noti che possono esistere più cammini con la caratteristica di essere minimi.

Questo esempio, apparentemente semplice, solleva complessi problemi di modellistica. Un primo problema è relativo al livello di dettaglio necessario nella rappresentazione. Se, per esempio, il percorso da effettuare parte da Milano e arriva a Brindisi, il grafo dovrà contenere solo le autostrade e le principali strade di collegamento fra città diverse e un'intera città potrà essere rappresentata con un nodo del grafo. Un eccesso di dettaglio appesantirebbe inutilmente la rappresentazione, rendendo ogni algoritmo di soluzione lento e inefficace. Se, d'altra parte, il percorso da effettuare parte da piazza San Pietro a Roma e arriva a un indirizzo di Frascati (a circa 20 km. da Roma), la rappresentazione dovrà essere completamente diversa, non solo perché il grafo sarà diverso, ma anche perché il peso degli archi, per essere significativo, dovrà prendere in considerazione i problemi del traffico urbano e quindi essere funzione dell'ora ed eventualmente del giorno. Infatti, effettuare tale percorso alle 10 di mattina di un giorno feriale non sarà ovviamente la stessa cosa che effettuarlo alle 4 del mattino (magari di un giorno festivo).

Un secondo problema è relativo al tipo di obiettivi che ci si propone. Infatti, una rappresentazione può essere o può non essere adeguata, a seconda del motivo per cui si vuole conoscere il cammino di tempo minimo. Per esempio, per alcune applicazioni critiche (autobus, vigili del fuoco, polizia), non basta l'informazione sul valore atteso del tempo di transito, ma serve anche valutare la varianza di tale tempo, ossia le possibili variazioni rispetto alla media. È meglio infatti utilizzare un percorso un po' più lungo ma con un tempo di percorrenza prevedibile con relativa certezza, piuttosto che un percorso mediamente più breve ma per cui vi sia il rischio di restare imbottigliati nel traffico.

Un terzo problema è relativo alla quantità di informazioni che è necessario inserire nell'elaboratore per affrontare il problema. Per calcolare il cammino da Milano a Brindisi è necessario inserire l'intera carta stradale italiana, o basta una porzione? Da un lato, più informazioni vengono inserite maggiore è il tempo necessario per inserirle e il costo dell'operazione; d'altro canto se vengono calcolati spesso percorsi di tempo minimo sulla rete stradale italiana forse conviene memorizzare tutto in modo organico una volta per tutte.

7.1.2 Costruzione di una autostrada

Il problema considerato è quello di costruire al costo minimo una autostrada fra le città A e B. Nel grafo riportato in figura 7.1 i nodi rappresentano i punti per cui l'autostrada può passare e gli archi i

possibili collegamenti fra punti. Il peso degli archi rappresenta il costo di costruzione della relativa tratta di autostrada.

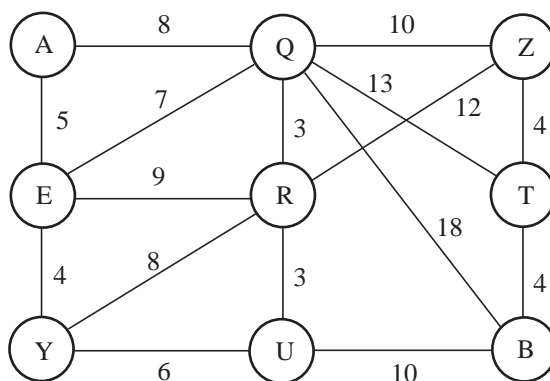


Figura 7.1: Cammino minimo tra A e B

Si noti che in questo caso il grafo è non orientato in quanto ogni arco può essere percorso in entrambi i sensi; per ricondursi al caso orientato basta sostituire ogni arco con due archi orientati in senso opposto, a ognuno dei quali viene attribuito un peso pari al peso dell'arco eliminato.

La scelta di una autostrada tra A e B con costo complessivo di costruzione minimo, corrisponde alla scelta del cammino di minimo costo dal nodo A al nodo B sul grafo. Nella figura 7.1 tale cammino è indicato in grassetto e il costo complessivo è pari a 24.

7.2 Cammini minimi e massimi su grafi aciclici

Una classe di grafi orientati di particolare interesse in campo applicativo (vedi gli esempi alla fine del capitolo) è la classe di grafi aciclici. La ricerca di cammini minimi o massimi su tali grafi è uno strumento di progetto di notevole importanza. Come si vedrà nel seguito, i due problemi di minimo e di massimo sono, in questo caso particolare, risolti da due algoritmi identici eccetto che per la sostituzione di un massimo a un minimo nella formula ricorsiva alla base del procedimento. Nel seguito verrà prima considerato il caso di un problema di cammino minimo. Per poter procedere con la descrizione dell'algoritmo, è necessario studiare prima una particolare tecnica di numerazione dei nodi di un grafo aciclico.

7.2.1 Numerazione topologica dei nodi di un grafo

Una caratteristica peculiare dei grafi aciclici (sia $G = (V, E)$ il grafo, con $|V| = n$ e $|E| = m$) consiste nella possibilità di numerare i nodi del grafo con i numeri $1, 2, 3, \dots, n-1, n$ in modo tale che:

$$\text{se esiste un arco dal nodo } i \text{ al nodo } j \text{ allora } j > i$$

Tale numerazione viene detta numerazione topologica dei nodi del grafo e non è in generale unica.

Non tutti i grafi possono essere numerati topologicamente. In effetti l'esistenza di una numerazione topologica dei nodi di un grafo caratterizza esattamente la classe dei grafi aciclici. Vale infatti il seguente teorema.

Teorema 7.2.1 *Un grafo è aciclico se e solo se esiste una numerazione topologica dei suoi nodi.*

Dimostrazione.

Sufficienza. Supponiamo che esista una numerazione topologica dei nodi e facciamo vedere che l'esistenza di un ciclo porterebbe ad una contraddizione. Possiamo assumere che i nodi siano numerati topologicamente, indicheremo l' i -esimo nodo di questa particolare numerazione, come v_i . Se esiste un ciclo (orientato) vuol dire che esiste una successione di nodi $(v_i, v_j, v_k, \dots, v_r, v_s)$ tali che

- esistono gli archi $(v_i, v_j), (v_j, v_k), \dots, (v_r, v_s)$;
- $v_i = v_s$.

Ma allora, da una parte, siccome la numerazione è topologica abbiamo $i < j < k < \dots < r < s$, cioè $i < s$, mentre dall'altra, poiché $v_i = v_s$ abbiamo $i = s$. Questa è una contraddizione e così il grafo deve essere aciclico.

Necessità. Supponiamo che il grafo sia aciclico e mostriamo che deve esistere almeno una numerazione topologica. La dimostrazione è costruttiva, faremo cioè vedere che esiste una numerazione topologica costruendone una.

Come primo passo osserviamo che se il grafo è aciclico, deve esistere almeno un nodo che non abbia archi entranti. Infatti, se ciò non fosse vero, potremmo ragionare nel modo seguente. Prendiamo un nodo qualunque, chiamiamolo v_1 . Siccome tutti i nodi hanno archi entranti esiste un nodo predecessore di v_1 , indichiamolo con v_2 ; notiamo che per come abbiamo scelto v_2 esiste l'arco (v_2, v_1) (attenzione, questa è una numerazione non topologica). Possiamo ripetere il ragionamento con v_2 e trovare un nodo v_3 tale che esista l'arco (v_3, v_2) . Siccome stiamo supponendo, per assurdo, che tutti gli archi abbiano degli archi entranti, possiamo ripetere il ragionamento quante volte vogliamo. Notiamo che ogni nodo generato deve essere diverso dai precedenti, altrimenti avremmo trovato un ciclo, contraddicendo l'aciclicità del grafo. D'altra parte, arrivati a v_n i nodi del grafo sono "finiti" e quindi il predecessore di v_n che stiamo supponendo esistente per assurdo, deve per forza essere uno dei nodi già esaminati. Così si viene a formare un ciclo.

Quindi dato un grafo aciclico deve per forza esistere almeno un nodo che non ha archi entranti. Prendiamo uno di questi nodi e numeriamolo con il numero 1. Eliminiamo dal grafo il nodo 1 e tutti gli archi uscenti da esso. Il nuovo grafo che otteniamo è ovviamente ancora un grafo aciclico. Quindi per lo stesso ragionamento fatto prima deve esistere almeno un nodo che non ha archi entranti. Prendiamo uno di questi nodi e numeriamolo con il numero 2. Notiamo che ovviamente se consideriamo il grafo originario il nodo 2 può avere archi entranti, ma solo provenienti dal nodo 1 e quindi la condizione $i < j$ è rispettata. Possiamo ora ripetere il procedimento n volte (quanti sono i nodi) ed ottenere così una numerazione topologica del grafo. \square

Il precedente teorema è importante, anche perché nella dimostrazione della necessità è sostanzialmente dato un algoritmo per numerare topologicamente i nodi di un grafo.

Riesponiamo qui l'algoritmo per chiarezza.

- Siccome il grafo è aciclico, deve esistere almeno un nodo con solo archi uscenti;
- individuiamo uno di questi nodi e attribuiamogli il numero 1;
- cancelliamo il nodo numerato e tutti gli archi adiacenti, nel nuovo grafo ridotto individuiamo un nodo con soli archi uscenti e attribuiamogli il numero 2, e così via fino ad aver numerato tutti i nodi

La correttezza e validità di questa procedura è stata provata nella dimostrazione della necessità del Teorema 7.2.1.

Notiamo che se ad un certo punto dell'applicazione della procedura non possiamo procedere, se succede cioè che ad un determinato passo non riusciamo a trovare un nodo senza archi entranti, questo vuol dire che il grafo considerato contiene un ciclo.

Quindi la procedura per la numerazione topologica di un grafo può anche essere utilizzata per determinare se un grafo è aciclico o meno.

Come esempio consideriamo il grafo di figura 7.2.

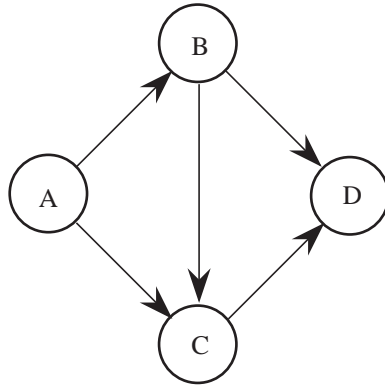


Figura 7.2: Numerazione dei nodi di un grafo

Si tratta di decidere se il grafo è aciclico e, in caso, numerare i nodi topologicamente. I vari passi dell'algoritmo sono riportati qui di seguito

Passo	Nodo senza archi entranti	Nodi non ancora numerati	Numerazione
1	A	B,C,D	A=1
2	B	C,D	B=2
3	C	D	C=3
4	D	\emptyset	D=4

Il procedimento è terminato con la numerazione di tutti i nodi. Il grafo è dunque aciclico e la numerazione trovata è topologica.

Supponiamo ora che nel precedente grafo l'arco (B, D) sia orientato da D a B . Ovviamente si verrebbe a creare un ciclo. Se proviamo ad applicare la procedura per la determinazione di una numerazione topologica possiamo iniziare numerando il nodo A come nodo 1. Ma dopo non possiamo più procedere perché eliminato il nodo A e gli archi da esso uscenti $((A,B)$ e $(A,C))$ non è più possibile individuare un nodo che non abbia archi entranti.

7.2.2 Un algoritmo per il cammino minimo su grafi aciclici

La numerazione dei nodi di un grafo aciclico descritta nella sezione precedente consente di costruire un algoritmo di soluzione per il problema di cammino minimo particolarmente semplice. Infatti, nella ricerca di un cammino tra una qualsiasi coppia di nodi i e j del grafo, a causa della numerazione attribuita ai nodi, si può affermare che:

se $j < i$ allora non esistono cammini da i a j ;

se $j > i$ allora gli unici nodi che è necessario considerare nella ricerca del cammino da i a j sono i nodi con indice k tale che $i < k < j$.

Infatti, se il cammino passasse per un nodo $h > j$, allora non potrebbe tornare su j , a causa della mancanza di archi che collegano nodi con indice maggiore a nodi con indice minore; se passasse per un nodo $h < i$, allora dovrebbe esistere un cammino da h a i , il che comporterebbe l'esistenza di archi che collegano nodi con indice maggiore a nodi con indice minore.

Sulla base di queste considerazioni, è possibile impostare un algoritmo per il calcolo dell'albero dei cammini minimi tra un nodo del grafo (per esempio il nodo 1) e tutti i nodi con indice superiore (per quelli con indice inferiore non esiste sicuramente un cammino; ovviamente, se il nodo di partenza è quello contrassegnato con l'indice 1, allora si tratta di calcolare l'albero dei cammini minimi tra il nodo 1 e tutti gli altri). L'algoritmo per il calcolo dell'albero dei cammini minimi dal nodo 1 a tutti gli altri nodi si basa sul calcolo in sequenza dei cammini minimi dal nodo 1 al nodo 2, dal nodo 1 al nodo 3, dal nodo 1 al nodo 4, e così via. Indichiamo con:

- $p(i, j)$ il peso dell'arco (i, j) che parte dal nodo i e arriva al nodo j ;
- $f(i)$ il valore del cammino minimo dal nodo 1 al nodo i ;
- $J(i)$ il nodo che precede i su tale cammino (nel caso il cammino minimo non sia unico, allora se ne sceglie uno qualsiasi fra quelli minimi).

Si noti che dai valori $J(i)$ è possibile ricostruire in modo immediato l'albero (o uno dei possibili alberi) dei cammini minimi.

Possiamo allora illustrare l'algoritmo per il calcolo dei percorsi minimi.

- $f(1) := 0; J(1) := 1$;
- per $j = 2, 3, 4, \dots, n - 1, n$ ripeti la seguente serie di operazioni

$$f(j) := \min_{(i,j) \in \omega^-(j)} \{f(i) + p(i, j)\};$$

$$J(j) := \text{valore di } i \text{ per cui si è verificato il minimo};$$

Si noti che una volta assegnato un peso $f(i)$ a un nodo (cioè un valore del cammino minimo dal nodo 1 al nodo considerato), tale peso non viene più modificato nel corso dell'algoritmo, ma indica in modo definitivo il valore del cammino. Questo è dovuto al fatto, già citato, che tutti i nodi successivi non devono essere considerati per il calcolo del percorso dal nodo 1 al nodo i .

Come esempio consideriamo il grafo di figura 7.3.

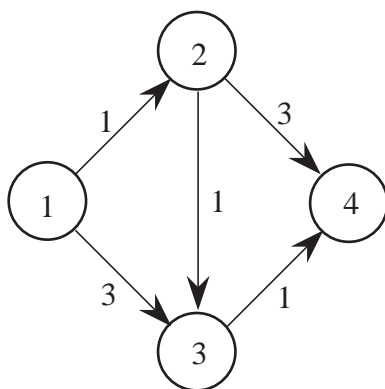


Figura 7.3: Percorso minimo su grafi aciclici

Si tratta di determinare l'albero dei cammini minimi tra il nodo 1 e tutti gli altri nodi. I vari passi dell'algoritmo (corrispondenti alla successione di nodi visitati, si osservi che in questo caso particolare l'indice del passo coincide con l'indice del nodo visitato) portano alla seguente successione di valori $f(i)$ e $J(i)$

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 1$	$J(2) = 1$
3	$f(3) = \min\{3, 1 + 1\} = 2$	$J(3) = 2$
4	$f(4) = \min\{1 + 3, 2 + 1\} = 3$	$J(4) = 3$

Il corrispondente albero dei cammini minimi è riportato in figura 7.4

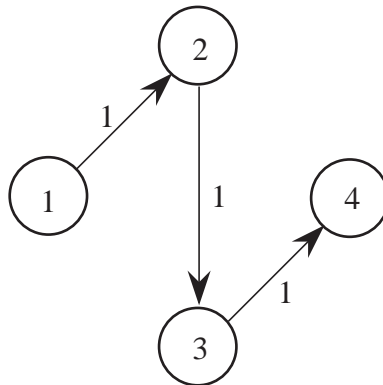


Figura 7.4: Albero dei cammini minimi

Esempio 7.2.2 Sia dato il grafo di Figura 7.5. Determinare l'albero dei cammini minimi, utilizzando

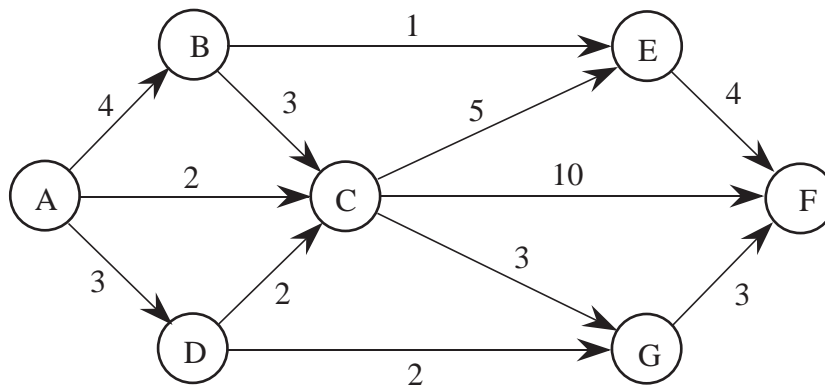


Figura 7.5: Grafo Esercizio 7.2.2

l'algoritmo per grafi aciclici.

Soluzione. Si deve prima numerare topologicamente il grafo. I passi sono riportati nella seguente tabella ed il grafo risultante in Figura 7.6

Passo	Nodo senza archi entranti	Nodi non ancora numerati	Numerazione
1	A	B,C,D,E,F,G	A=1
2	B o D	B,C,E,F,G	D =2
3	B	C,E,F,G	B=3
4	C	E,F,G	C=4
5	E o G	F,G	E=5
6	G	F	G=6
7	F	\emptyset	F=7

Osserviamo che la numerazione topologica in questo esempio non é unica; infatti ai passi 2 e 5 potevamo scegliere tra due nodi. Applichiamo ora l'algoritmo. I passi sono riportati nella seguente tabella

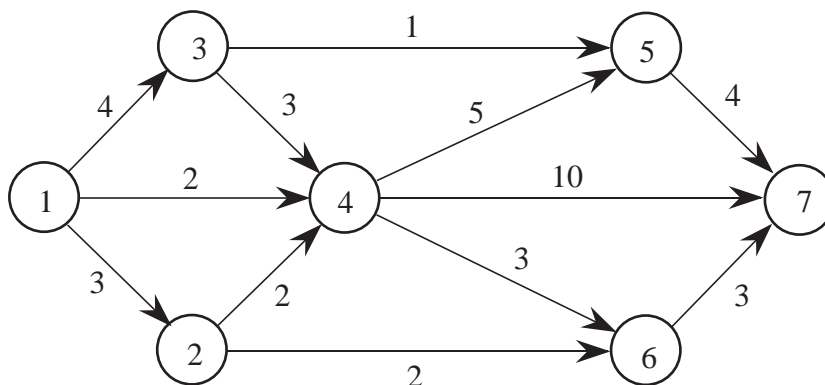


Figura 7.6: Numerazione topologica del grafo di Figura 7.5

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 3$	$J(2) = 1$
3	$f(3) = 4$	$J(3) = 1$
4	$f(4) = \min\{0 + 2, 3 + 2, 4 + 3\} = 2$	$J(4) = 1$
5	$f(5) = \min\{4 + 1, 2 + 5\} = 5$	$J(5) = 3$
4	$f(6) = \min\{3 + 2, 2 + 3\} = 5$	$J(6) = 4$
4	$f(7) = \min\{2 + 10, 5 + 4, 5 + 3\} = 8$	$J(7) = 6$

Il corrispondente albero dei cammini minimi è riportato in figura 7.7.

7.2.3 Un algoritmo per il cammino massimo su grafi aciclici

Se il problema di ottimo è quello della determinazione del cammino di peso massimo sul grafo, allora è facile convincersi che basta sostituire nella formula ricorsiva al min un max e tutte le considerazioni fatte continuano, in questo caso particolare di grafi aciclici, a essere valide. Oltre che da considerazioni dirette questo risultato può essere dedotto considerando che il problema di trovare un cammino di peso massimo su un grafo aciclico è equivalente a quello di trovare un cammino di peso minimo sullo stesso grafo dove i pesi, però, sono stati cambiati di segno.

L'algoritmo per il calcolo dei cammini massimi su grafi aciclici è allora il seguente

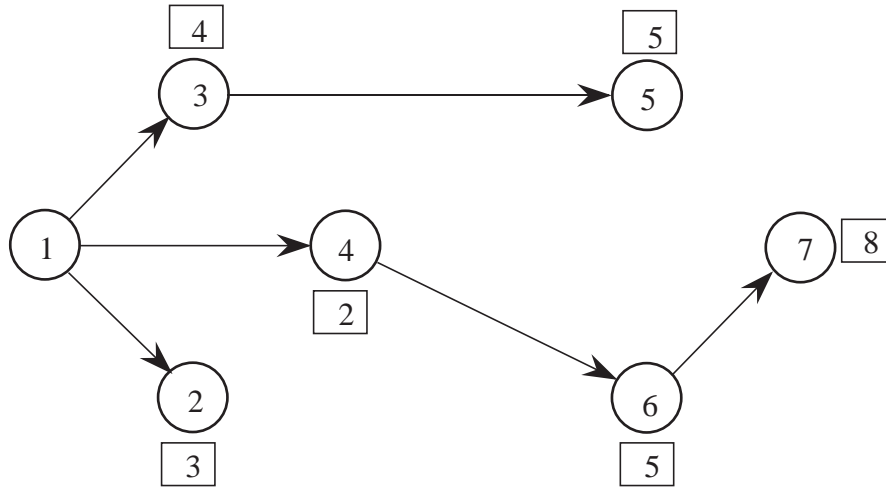


Figura 7.7: Albero dei cammini minimi

- $f(1) := 0; J(1) := 1;$
- per $j = 2, 3, 4, \dots, n - 1, n$ ripeti la seguente serie di operazioni

$$f(j) := \max_{(i,j) \in \omega^-(j)} \{f(i) + p(i,j)\};$$

$$J(i) := \text{valore di } i \text{ per cui si è verificato il massimo};$$

Come esempio consideriamo sempre il grafo di figura 7.3 Si tratta di determinare l'albero dei cammini massimi tra il nodo 1 e tutti gli altri nodi. I vari passi dell'algorithmo portano alla seguente successione di valori $f(i)$ e $J(i)$

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 1$	$J(2) = 1$
3	$f(3) = \max\{3, 1 + 1\} = 3$	$J(3) = 1$
4	$f(4) = \max\{1 + 3, 3 + 1\} = 4$	$J(4) = 2$ (oppure $J(4) = 3$)

7.3 Cammini minimi su grafi con pesi positivi: algoritmo di Dijkstra

L'algorithmo di Dijkstra permette di risolvere il problema del cammino minimo fra due nodi qualora tutti i pesi degli archi siano **non negativi**. Più precisamente, l'algorithmo calcola il peso del cammino minimo da un nodo s a tutti gli altri nodi del grafo, costruendo contemporaneamente l'albero dei cammini minimi. Siccome in questo caso la numerazione non gioca nessun ruolo, in questo paragrafo supponiamo, senza perdita di generalità, che s sia sempre uguale a 1.

Notiamo la differenza con il caso esaminato nel paragrafo precedente: nel caso precedente non c'era nessuna restrizione sui pesi, ma c'era una restrizione sulla *topologia* del grafo, che non doveva contenere cicli orientati. Nel caso esaminato in questo paragrafo, non c'è nessuna restrizione sulla topologia del grafo (che può essere qualunque e contenere, dunque, cicli orientati), ma c'è una restrizione sui pesi, che devono essere non negativi.

È evidente che il caso analizzato in questo paragrafo è di interesse in quanto, per esempio, in tutti i casi in cui la ricerca di cammini minimi corrisponde alla ricerca di un reale percorso in, per esempio, una città, esisteranno cicli, ma i pesi, che corrispondono a distanze fisiche, sono ovviamente positivi.

Un'altra differenza che vogliamo segnalare subito è che in questo caso, non è possibile dare una semplice variante dell'algoritmo che calcoli i cammini di peso massimo in quanto, se facciamo diventare il problema di minimo un problema di massimo cambiando i segni dei pesi, il grafo che otteniamo ha i pesi tutti *non positivi*, e quindi l'algoritmo non è più utilizzabile.

L'algoritmo per il calcolo dei cammini minimi su grafi aciclici si basa fortemente sul fatto che i nodi del grafo siano numerati topologicamente. Tenendo conto del fatto che il cammino minimo tra il nodo i e il nodo j (con $j > i$), se esiste, può passare solo per i nodi k , con k compreso tra i e j , abbiamo sviluppato una semplice procedura iterativa. Nel caso di grafi con pesi non negativi vogliamo, in qualche modo, ancora cercare di sviluppare un algoritmo che abbia le stesse caratteristiche. Ovviamente, non disponendo più di una numerazione topologica dobbiamo ragionare in maniera diversa. Vediamo su un esempio come possiamo ragionare.

Consideriamo il grafo in figura 7.8 e proponiamoci di trovare i cammini minimi dal nodo 1 a tutti i nodi da esso raggiungibili. Analogamente al caso dei grafi aciclici, poniamo $f(1) = 0$ e $J(1) = 1$. Vogliamo anche in questo caso arrivare ad associare ad ogni nodo i del grafo due etichette, $f(i)$ e $J(i)$ che diano rispettivamente, la distanza dal nodo 1 e il predecessore di i su un cammino minimo che va da 1 a i .

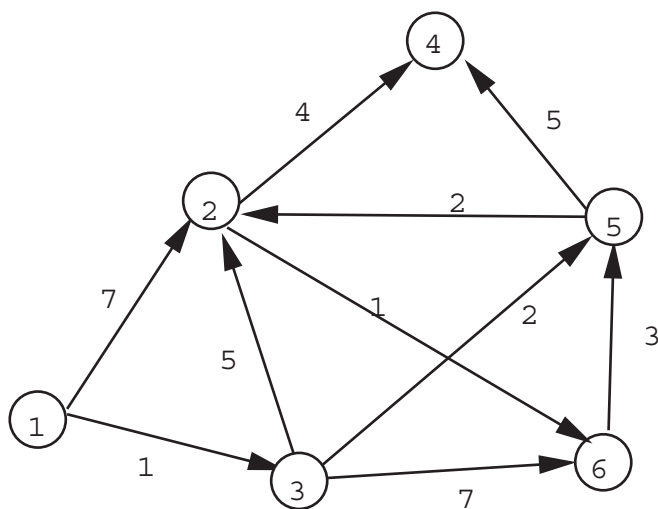


Figura 7.8: Grafo con pesi non negativi

Consideriamo ora i nodi raggiungibili da 1 con un solo arco: sono 2 e 3. Notiamo che il peso dell'arco $(1,3)$, che è uguale a 1, è minore del peso dell'arco $(1,2)$, che è uguale a 7. Possiamo allora porre $f(3) = 1$ e $J(3) = 1$. Infatti, supponiamo per assurdo che esista un cammino minimo per andare da 1 a 3 diverso da quello fornito dall'arco $(1,3)$ e con un peso più piccolo. Questo cammino dovrebbe prima "uscire" da 1, passando quindi per un arco di peso 1 o uno di peso 7, quindi dovrebbe "tornare" a 3. Ma siccome gli archi hanno tutti pesi non negativi, il peso di questa seconda parte di cammino si va ad aggiungere a quello dell'arco usato per "uscire" da 1. E' quindi ovvio che, essendo il peso dell'arco $(1,3)$ il più piccolo tra i pesi degli archi uscenti da 1, il peso di questo altro ipotetico cammino, deve essere almeno di 1. Quindi le etichette assegnate al nodo 3 sono corrette.

Possiamo ora ripetere questo ragionamento. Consideriamo i nodi raggiungibili da 1 e 3 (i nodi già

etichettati). Sono 2, 5, 6. Le distanze minime da 1, con la restrizione di passare solo per 1 e 3, sono:

- per il nodo 2: 6 (distanza data dal cammino che passa per i nodi 1, 3, 2);
- per il nodo 5: 3 (distanza data dal cammino che passa per i nodi 1, 3, 5);
- per il nodo 6: 8 (distanza data dal cammino che passa per i nodi 1,3,6).

Notiamo che per andare al 2 due passando solo per i nodi 1 e 3 (già etichettati) esiste anche dato dall'arco (1,2), ma questi ha un peso 7 superiore a quello che si ottiene passando per 3. Osserviamo anche che il nodo scelto è, *tra i nodi ancora non etichettati, quello che ha la distanza minima da 1 se ci limitiamo a considerare solo i cammini che passano per i nodi già etichettati*. A questo punto possiamo ragionare in modo simile a quello adottato prima, anche se la situazione è leggermente più complessa. Consideriamo il nodo 5 (quello che ha la distanza più piccola, tra quelli raggiungibili dai nodi 1 e 3), e poniamo $f(5) = 3$ e $J(5) = 3$. Queste sono etichette corrette. Supponiamo infatti che esista un altro cammino, C , da 1 a 5, diverso da quello trovato (e dato dai nodi 1,3 e 5) e con un peso minore di 3. Questo cammino C deve passare per almeno un nodo diverso da 1 e 3 e 5 per l'osservazione in corsivo fatta poche righe sopra. Ora, sia j il primo nodo diverso da 1, 3 e 5 nel cammino C . Il peso del cammino da 1 a j deve essere di almeno 3, sempre per l'osservazione in corsivo fatta prima. Siccome la parte del cammino che va da j a 5 ha un valore non negativo (per l'ipotesi che i pesi siano tutti non negativi) abbiamo di nuovo un assurdo.

A questo punto la tecnica da adottare dovrebbe essere chiara. A un generico passo dell'algoritmo possiamo supporre di avere un insieme di nodi, diciamo S , che hanno già le loro etichette f e J correttamente assegnate. Si sceglie il nuovo nodo da mettere in S come il nodo (o uno dei nodi) che ha la distanza minima da 1 con il vincolo di passare solo per nodi di S . Ad ogni passo aggiungiamo un nodo in S e quindi in un numero di passi uguale al numero dei nodi, l'algoritmo termina con le etichette correttamente assegnate.

Qui di seguito diamo una descrizione più dettagliata (e più vicina a una possibile implementazione sul calcolatore) dell'algoritmo delineato. In questa descrizione indichiamo con S i nodi a cui sono state assegnate le etichette corrette f e J , e con T tutti gli altri nodi. A differenza di quanto visto finora, però, noi diamo delle *etichette provvisorie*, che denominiamo sempre, per brevità, f e J , anche ai nodi in T . Se i appartiene a T , $f(i)$ rappresenta le distanze minima del nodo i dal nodo 1, con il vincolo di passare solo per nodi di S , e $J(i)$ è il corrispondente predecessore lungo il cammino minimo. Se non esiste nessun cammino da 1 a i che passa solo per nodi in S , poniamo $f(i) = +\infty$ e $J(i)$ è lasciato indefinito. Queste etichette hanno l'unico scopo di facilitare, ad ogni passo, la scelta del nodo in T da mettere in S . Infatti è chiaro che basterà scegliere, di volta in volta, il nodo in T con il valore di f più piccolo. Ovviamente, ad ogni iterazione, cambiando i nodi in S , le etichette provvisori dovranno essere aggiornate. Questo viene fatto nel punto (c) dell'algoritmo, la cui logica dovrebbe essere chiara, e che verrà ulteriormente chiarito dall'esempio che faremo subito dopo la descrizione dell'algoritmo. L'algoritmo termina quando o tutti i nodi del grafo sono in S o quando tutti i nodi in T hanno il valore di f uguale a $+\infty$, fatto che ovviamente indica che i nodi in T non sono raggiungibili da 1.

Algoritmo di Dijkstra

(a) *Inizializzazione.*

Poni $S \leftarrow \{1\}$, $T \leftarrow \{2, \dots, n\}$. $f(1) = 0$, $J(1) = 1$.

Poni $f(i) = p(1, i)$, $J(i) = 1$, per $(1, i) \in \omega^+(1)$.

Poni $f(i) = +\infty$, per $(1, i) \notin \omega^+(1)$.

(b) *Assegnazione etichetta permanente*

Trova $j \in T$ tale che $f(j) = \min_{i \in T} f(i)$.

Poni $T = T - \{j\}$, $S = S \cup \{j\}$.

Se $T = \emptyset$ o $f(i) = +\infty, \forall i \in T$ **STOP** (terminazione dell'algoritmo).

(c) *Assegnazione etichetta provvisoria*

Per ogni $(j, i) \in T \cap \omega^+(j)$ tale che $f(i) > f(j) + p(j, i)$ Poni:

c.1 $f(i) = f(j) + p(j, i)$

c.2 $J(i) = j$

Vai al passo (b).

Vediamo, nel caso del grafo di figura 7.8, come l'algoritmo proceda (per semplicità, l'evoluzione di T è omessa essendo $T = V - S$).

Iterazione 0 Inizializzazione.

(a) $S = \{1\}$. $d(1) = 0$. $d(2) = 7$. $d(3) = 1$. $d(4) = +\infty$. $d(5) = +\infty$. $d(6) = +\infty$.

$J(2) = 1$. $J(3) = 1$. $J(4) = 1$. $J(5) = 1$. $J(6) = 1$.

Iterazione 1.

(b) $j = 3$. $S = \{1, 3\}$.

(c.1) $(\omega^+(j) \cap T = \{(3, 2), (3, 5), (3, 6)\})$. È facile vedere che per ognuno dei nodi $\{2, 5, 6\}$ è verificata la condizione $d(i) > d(j) + p(j, i)$, e quindi le etichette vanno tutte aggiornate. $d(2) = d(3) + 1 = 6$. $d(5) = 2 + d(3) = 3$. $d(6) = 7 + d(3) = 8$.

(c.2) $J(2) = 3$. $J(5) = 3$. $J(6) = 3$.

Iterazione 2.

(b) $j = 5$. $S = \{1, 3, 5\}$.

(c.1) $(\omega^+(j) \cap T = \{(5, 2), (5, 4)\})$. $d(2) = d(5) + 2 = 5$. $d(4) = d(5) + 5 = 8$.

(c.2) $J(2) = 5$. $J(4) = 5$.

Iterazione 3

(b) $j = 2$. $S = \{1, 2, 3, 5\}$.

(c.1) $(\omega^+(j) \cap T = \{(2, 4), (2, 6)\})$. L'etichetta del nodo 4 non soddisfa la condizione al passo (c), per cui va' aggiornata solo l'etichetta (e il predecessore) del nodo 6. $d(6) = d(2) + 1 = 6$.

(c.2) $J(6) = 2$.

Iterazione 4.

(b) $j = 6$. $S = \{1, 2, 3, 5, 6\}$.

(c) $(\omega^+(j) \cap T = \emptyset)$.

Iterazione 5.

(b) $j = 4$. $S = \{1, 2, 3, 4, 5, 6\}$. **STOP**.

I pesi dei cammini minimi saranno quindi: $d(1) = 0$. $d(2) = 5$. $d(3) = 1$. $d(4) = 8$. $d(5) = 3$. $d(6) = 6$.

Una comoda rappresentazione dell'evolvere dell'algoritmo è la seguente forma tabellare ove le righe rappresentano iterazioni mentre le colonne rappresentano i nodi selezionati ad ogni iterazione. Per ciascun nodo j ci sono due colonne che riportano il valore della variabile $d(j)$ e della $J(j)$ all'iterazione i -esima. L'elemento selezionato all'iterazione i -esima è rappresentato in grassetto e, nelle iterazioni successive, il valore della variabile corrispondente non viene più riportato. La colonna corrispondente al nodo 1 è omessa.

	nodo									
	2		3		4		5		6	
	d	J	d	J	d	J	d	J	d	J
It. 0	7	1	1	1	$+\infty$	1(fitt.)	$+\infty$	1(fitt.)	$+\infty$	1(fitt.)
It. 1	6	3	1	1	$+\infty$	1(fitt.)	3	3	8	3
It. 2	5	5			8	5	3	3	8	3
It. 3	5	5			8	5			6	2
It. 4					8	5			6	2
It. 5					8	5				

7.4 Due esempi

In questo paragrafo consideriamo più in dettaglio due esempi di applicazioni non banali di quanto visto in questo capitolo.

7.4.1 Tecniche reticolari di programmazione delle attività

I progetti di grandi dimensioni sono costituiti da più attività, che devono essere tutte completate affinché il progetto di cui fanno parte sia completato, ma che possono essere iniziate e svolte indipendentemente l'una dall'altra, purché sia rispettata una data sequenza. Queste condizioni sono caratteristiche di molti progetti di sviluppo e produzione, ad esempio nel settore aeronautico ed aereo spaziale, o di costruzione, ad esempio nell'ingegneria civile, o di manutenzione di grossi sistemi; tutti progetti in cui il numero di attività costituenti può essere dell'ordine delle migliaia. La gestione di un progetto consiste nel coordinamento dell'esecuzione delle varie attività, unitamente al controllo dei tempi e dei costi di esecuzione. Poiché questo è evidentemente un problema di rilevante importanza economica, e a volte strategica, per esso sono state sviluppate, a partire dal 1958, alcune tecniche particolarmente efficaci, tra cui hanno assunto un ruolo importante il PERT (Program Evaluation and Review Technique) e il CPM (Critical Path Method). Il PERT è stato sviluppato inizialmente per pianificare le operazioni di ricerca e sviluppo connesse al progetto del missile Polaris, e l'applicazione di questa tecnica ha consentito di concludere il progetto con due anni di anticipo sui cinque anni inizialmente preventivati. Questo successo iniziale ha portato ad una sua rapida diffusione. Lo scopo principale del PERT è quello di pianificare e controllare i tempi di completamento delle attività di un progetto, e del progetto nel suo insieme, tenendo conto del fatto che i tempi di esecuzione delle varie attività non sono a priori noti con certezza, e possono variare in dipendenza di molteplici fattori aleatori (PERT-Time); successivamente sono state fatte estensioni alla pianificazione e controllo dei costi di esecuzione, soggetti anch'essi ad aleatorietà (PERT-Cost). PERT-Time e PERT-Cost costituiscono applicazioni software molto diffuse. Il CPM si applica invece quando al tempo di esecuzione di ogni attività può essere attribuito un valore certo, più o meno lungo a seconda di quanto si decide di spendere per l'esecuzione dell'attività stessa; è esperienza comune che se si riduce il tempo di esecuzione di un'attività il suo costo aumenta e viceversa. Il CPM ha come scopo principale quello di pianificare e controllare i tempi di esecuzione di un progetto, rendendo minima la spesa complessiva, e trova larga applicazione in programmi riguardanti la manutenzione periodica di grossi impianti industriali, e lavori di produzione e costruzione per cui esiste una consolidata esperienza, cosicché

si possono ritenere note con esattezza le relazioni costo-tempo di esecuzione. Base comune del PERT e del CPM è la rappresentazione del progetto mediante un grafo orientato, secondo opportune norme. Poiché in questo contesto, al grafo rappresentativo del progetto viene dato il nome di diagramma reticolare, queste tecniche vengono chiamate tecniche reticolari di programmazione delle attività. Dall'analisi del diagramma reticolare si possono ricavare molte informazioni significative sul progetto. Vogliamo qui illustrare alcune considerazioni, che possono essere fatte nell'analizzare i progetti, in cui giocano un ruolo importanti i cammini minimi.

Iniziamo la nostra analisi spiegando come sia possibile associare ad un progetto un *diagramma reticolare* (cioè un grafo orientato).

La costruzione del diagramma

Il diagramma reticolare rappresenta la successione temporale e la reciproca dipendenza delle varie attività che concorrono all'esecuzione del progetto, attività che devono essere completate prima che il progetto possa considerarsi eseguito. Il primo passo nella costruzione del diagramma reticolare consiste nell'individuazione e nell'elencazione di tutte le attività coinvolte nell'esecuzione del progetto, con un livello di disaggregazione tale per cui le si possa considerare ciascuna distinta da tutte le altre. Segue una fase di rappresentazione grafica, che dà luogo al disegno di un grafo orientato in cui ogni attività è rappresentata da un arco o ramo i cui nodi estremi rappresentano, secondo la direzione del ramo, l'inizio e il termine dell'attività in questione. Pertanto nei diagrammi reticolari un'attività A è rappresentata come in figura 7.9, ove i nodi *i* e *j* rappresentano rispettivamente l'inizio e il termine dell'attività.

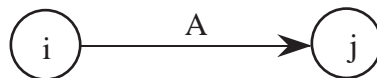


Figura 7.9: Rappresentazione grafica dell' attività A

Naturalmente tra le varie attività esistono delle precedenze, per cui, per ciascuna attività, esisteranno altre attività che devono essere completate prima che quella in questione possa avere inizio. Il caso più semplice di precedenza quello indicato in figura 7.10a, ove l'attività A precede l'attività B, e il nodo *j* rappresenta il termine dell'attività A e l'inizio dell'attività B. In figura 7.10b è rappresentato il caso in cui l'attività A precede l'attività B che a sua volta precede l'attività C. Può però anche avvenire che due attività, la A e la B precedano una terza, la C, senza che tra A e B esista una relazione di precedenza: questo caso, in cui le attività A e B possono essere svolte in parallelo, è rappresentato in figura 7.10c. In figura 7.10d abbiamo il caso in cui le due attività B e C, tra cui non sussistono precedenze, sono entrambe precedute dall'attività A. Per esprimere il fatto che l'attività A precede l'attività B utilizziamo la notazione $A < B$, per esprimere il fatto che l'attività B è preceduta dall'attività A, utilizziamo la notazione $B > A$.

Esempio 1. In figura 7.11b) è rappresentato il caso di un progetto il cui completamento richiede l'esecuzione di 9 attività, tra cui sussistono le relazioni di precedenza:

$$A < B,C; \quad B < D, E; \quad C < F; \quad D < G; \quad E,F < H; \quad G,H < I.$$

Il progetto rappresentato in 7.11b verrà più volte riutilizzato a scopo esemplificativo; ad esso faremo pertanto riferimento con il nome di progetto P1.

Nel disegnare il diagramma reticolare si utilizzano le seguenti regole fondamentali, alcune delle quali già implicitamente enunciate:

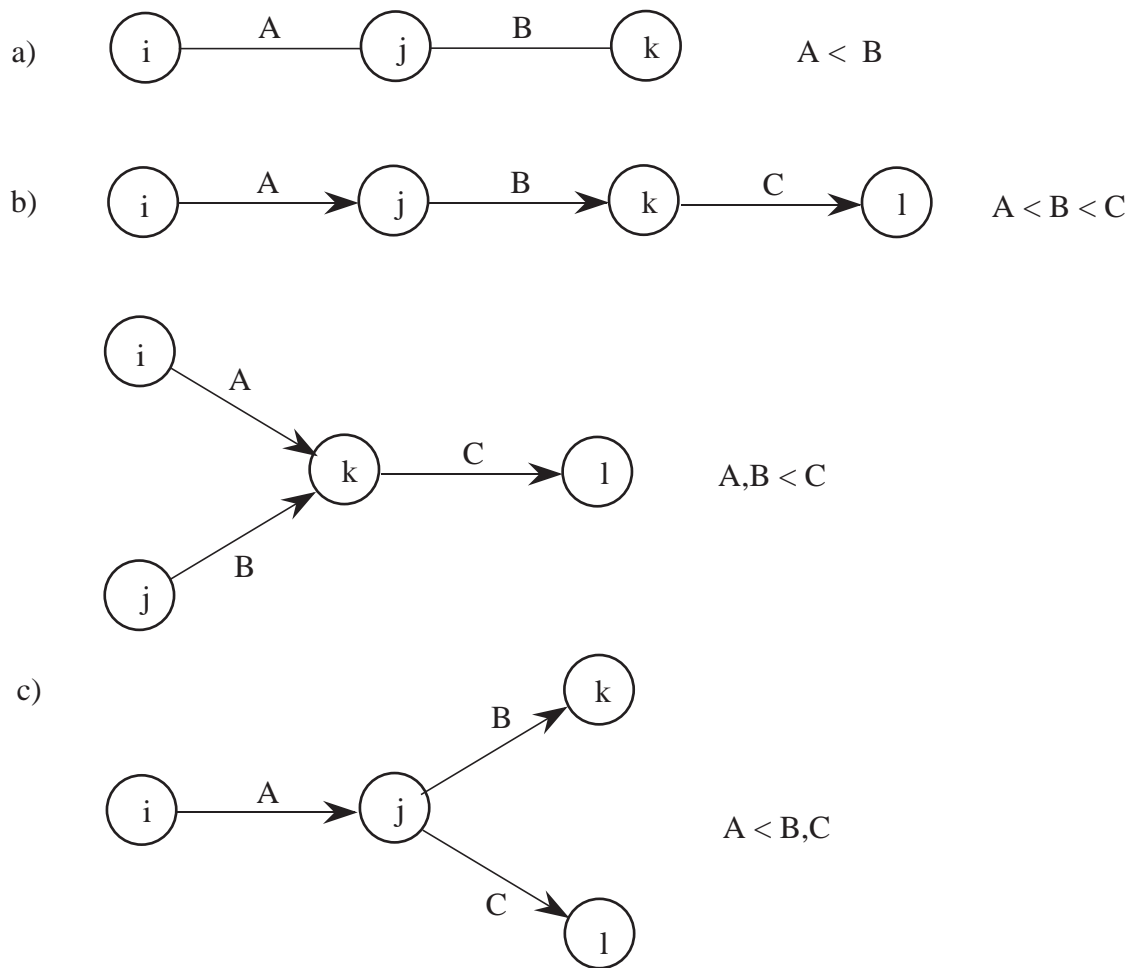


Figura 7.10: Rappresentazione grafica delle regole di precedenza

1. Le attività sono rappresentate dai rami del grafo.
2. L'inizio di un'attività è subordinato al completamento di tutte quelle che la precedono: in termini di diagramma reticolare ciò significa che rami diretti verso un nodo rappresentano attività da completare prima che abbiano inizio le attività rappresentate da rami aventi origine nel nodo stesso.
3. La lunghezza dei rami o la loro forma non hanno significato.
4. Due nodi non possono essere collegati da più di un ramo.
5. L'inizio del progetto è rappresentato da un nodo contrassegnato con zero.
6. Tutti i nodi sono numerati in modo che, se esiste un ramo diretto dal nodo i al nodo j , risulta $i < j$.
7. Il grafo può avere un solo nodo iniziale e un solo nodo finale.

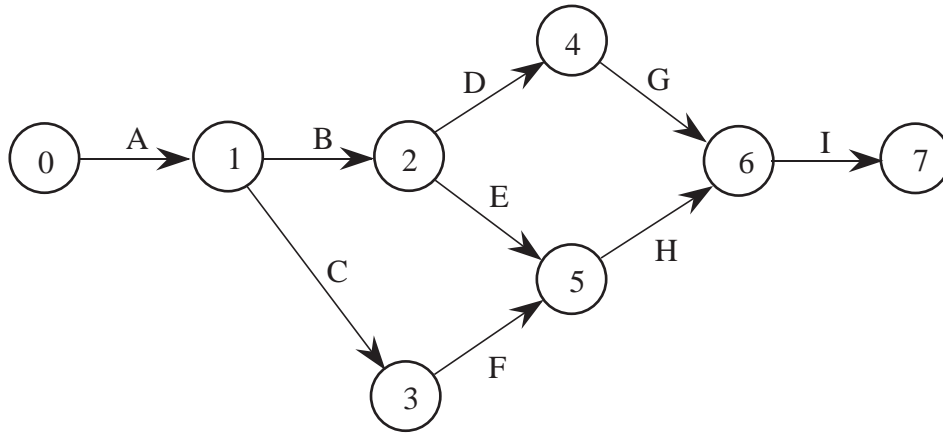


Figura 7.11: Diagramma reticolare del progetto P1

Delle suddette regole, le prime tre tengono conto della logica interna del grafo; le altre quattro sono richieste dai metodi di analisi del grafo, metodi che utilizzano tabulazioni ed uso di calcolatori.

Per quanto riguarda la regola 6, ricordiamo che una siffatta numerazione è detta numerazione topologica e che nel paragrafo precedente abbiamo visto che è possibile dare una numerazione topologica ai nodi di un grafo orientato se e solo se il grafo è aciclico. Bisogna quindi chiedersi se il diagramma reticolare di un progetto è un grafo aciclico. La risposta è ovviamente positiva. Se infatti esistesse un ciclo (orientato) nel diagramma reticolare di un progetto, questo vorrebbe dire, per come abbiamo costruito il diagramma reticolare stesso, che esistono delle attività che non possono iniziare prima di essere state concluse, e questo è ovviamente assurdo.

La regola 4, infine, ha lo scopo di rendere univoca la corrispondenza tra coppie di nodi ed attività, corrispondenza che potrebbe venire meno quando alcune attività possono essere svolte in parallelo, come accade nel seguente esempio.

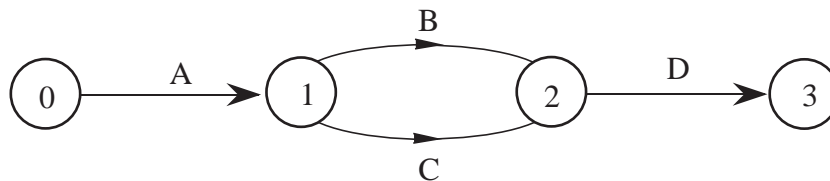


Figura 7.12: Una coppia di nodi che non individua univocamente un'attività

Esempio 2. Consideriamo le attività A, B, C, D, con le relazioni di precedenza $A < B, C$; $B, C < D$. Il grafo costruito ignorando la regola 4 è quello di 7.12, in cui alla coppia di nodi (1,2) non è associata in maniera univoca un'attività.

Quando la regola 4 non è esplicitamente soddisfatta dal progetto, come accade nell' Esempio 2, occorre ricorrere ad un artificio che consiste nell'introdurre un' *attività fittizia*, cui va associato un tempo di esecuzione nullo: nel caso dell'Esempio 2, introducendo l'attività fittizia X si ottiene il grafo della 7.13,

che rispetta la regola 4. Con l'introduzione di attività fittizie è quindi possibile individuare ogni attività

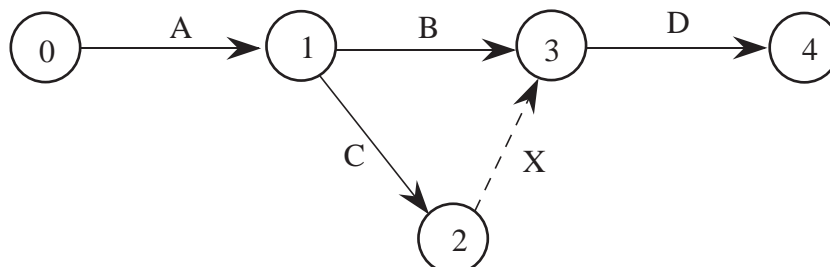


Figura 7.13: Introduzione di un'attività fittizia

mediante la coppia ordinata dei nodi estremi.

Esercizio 1. Un progetto comporta l'esecuzione delle 7 attività A, B, C, D, E, F, G, tra cui sussistono le relazioni di precedenza: $A < B, C$; $C < D, E$; $D, E < F$; $B, F < G$. Si costruisca il diagramma reticolare del progetto.

Il progetto considerato nell'esercizio precedente (il cui diagramma reticolare è dato in fondo al capitolo) sarà in seguito riutilizzato a scopo di esercizio; ad esso faremo riferimento con il nome di progetto P2.

Un altro caso in cui è richiesta l'introduzione di un'attività fittizia si verifica quando due attività precedono entrambe una terza attività, e una sola delle due ne precede una quarta. In questo caso è solo l'introduzione di un'attività fittizia che rende possibile la costruzione del grafo, come si vede nel prossimo esempio.

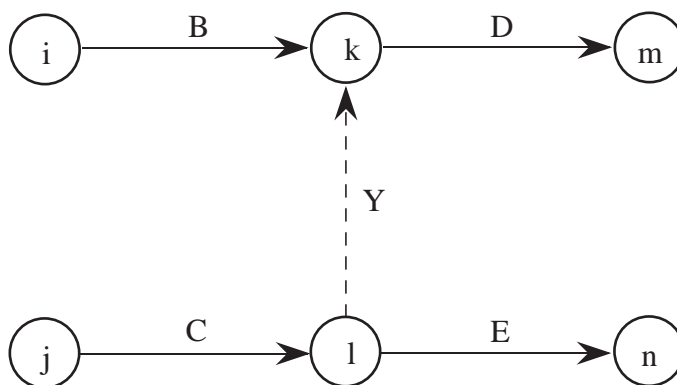


Figura 7.14: Introduzione dell'attività fittizia Y

Esempio 3. Un progetto prevede, tra la altre, le attività B, C, D, E, che devono essere svolte rispettando le precedenze: $B, C < D$; $C < E$. Dal diagramma di figura 7.14 si rileva come solo l'introduzione dell'attività fittizia Y rende possibile la rappresentazione di questa parte del progetto.

Nel diagramma reticolare ogni nodo (ad eccezione del primo e dell'ultimo) rappresenta il termine di alcune attività e l'inizio di altre. Pertanto, in questo contesto, i nodi vengono anche chiamati eventi.

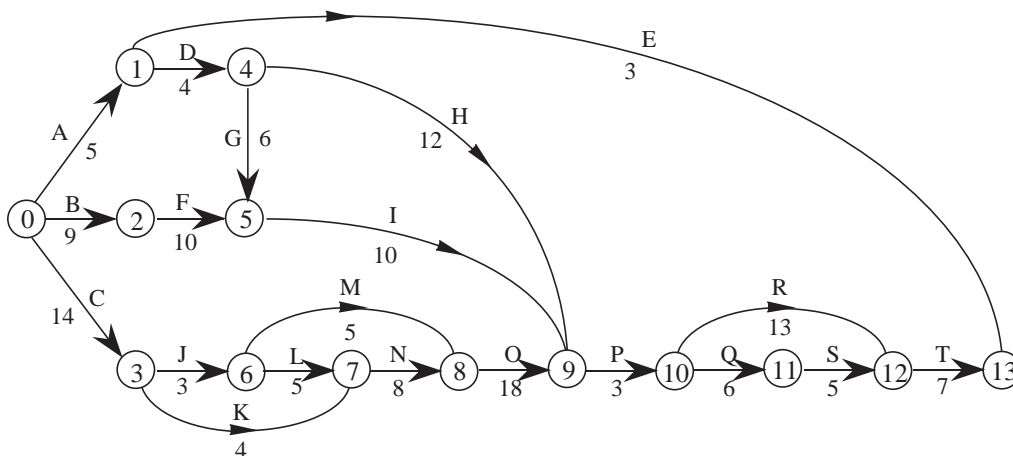


Figura 7.15: Diagramma reticolare del progetto P3

Esempio 4 Un progetto consiste nell'esecuzione di una serie di attività indicate con A, B, ..., T con le seguenti relazioni di precedenza: A, B, C possono iniziare immediatamente; D, E > A; F > B; G, H > D; I > F, G; J, K > C; M, L > J; N > K, L; O > M, N; P > H, I, O; R, Q > P; S > Q; T > R, S. Costruiamo il diagramma reticolare del progetto, numerando i nodi in modo che se il ramo (i, j) rappresenta un'attività, risulta $i < j$. Utilizzando le regole prima elencate, otteniamo il grafo di figura 7.15.

Anche il progetto considerato nell'Esempio 4 verrà riutilizzato nel seguito; ad esso faremo riferimento con il nome di progetto P3.

Domanda 1. Sai descrivere a cosa corrisponde il verificarsi dell'evento 9 nel diagramma reticolare del progetto P3 rappresentato in figura 7.15?

Domanda 2. Nel costruire il diagramma reticolare del progetto P3 è stato necessario introdurre attività fittizie ?

Esercizio 2. Supponiamo che alle relazioni di precedenza del progetto P3 debba essere aggiunta la $C < F$. Come si modifica il grafo di figura 7.15?

Esercizio 3. Supponi che il progetto P3 sia ampliato con l'introduzione dell'attività U, per cui sussistono le relazioni di precedenza $U > M, N$; $U < Q$. Come si modifica il grafo di figura 7.15?

Il percorso critico

Abbiamo finora visto come sia possibile costruire un grafo che rappresenti l'esecuzione di un progetto, dopo che il progetto stesso è stato decomposto in attività, o fasi, di cui si siano analizzate le relazioni di precedenza. Non abbiamo però finora tenuto conto del tempo richiesto per l'esecuzione delle varie attività che compongono il progetto, e che ovviamente condizionano il tempo di esecuzione complessivo. Poiché i metodi reticolari di programmazione hanno, tra gli altri, lo scopo di controllare i tempi di esecuzione delle attività al fine di ottenere il rispetto del tempo di completamento del progetto, occorre aggiungere alla analisi qualitativa delle precedenze già effettuata anche un'analisi quantitativa che determini i valori

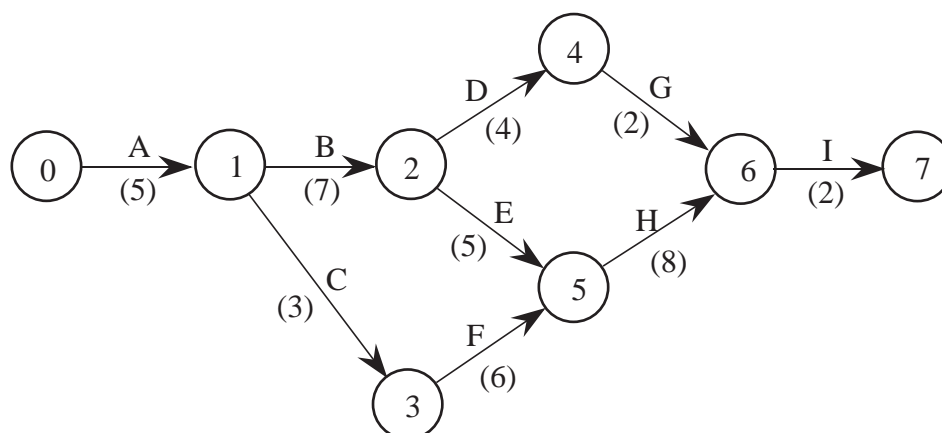


Figura 7.16: Tempi di esecuzione delle attività del progetto P1

temporali corrispondenti agli eventi descritti dal grafo, e individui i limiti entro cui tali valori temporali possono variare senza pregiudicare il valore del tempo complessivo di completamento. Per effettuare quest'analisi associamo ad ogni attività (i, j) un tempo di esecuzione t_{ij} . Il tempo di esecuzione t_{ij} può essere assunto come variabile certa, il che avviene nel CPM, o come variabile aleatoria, il che avviene nel PERT. In entrambi i casi il metodo di analisi è fondamentalmente lo stesso, per cui in questo paragrafo facciamo riferimento alla situazione in cui i tempi di esecuzione sono noti con certezza. In ogni progetto esiste un certo insieme di attività che sono di particolare importanza ai fini della determinazione del tempo di completamento dell'intero progetto, nel senso che se si verifica un ritardo nel completamento di una di queste attività, si verifica un ritardo anche nel completamento del progetto. Altre attività invece sono meno importanti, nel senso che possono anche subire un ritardo, entro certi limiti, senza che l'intero progetto ne risenta. È evidente l'importanza di distinguere tra questi due tipi di attività, così come quella di determinare il tempo minimo entro il quale certe attività intermedie possono essere completate. Quanto esposto in questo paragrafo serve proprio a consentire questa analisi. Supponiamo dunque che ad ogni attività (i, j) sia associato il tempo di esecuzione t_{ij} ; per le attività fittizie il tempo di esecuzione è ovviamente nullo.

Definizione 7.4.1 Si definisce tempo minimo di raggiungimento del nodo i , e si indica con t_i , il minimo tempo entro cui possono essere terminate tutte le attività afferenti al nodo i .

Data la definizione precedente viene del tutto naturale definire il tempo di completamento minimo del progetto nel seguente modo.

Definizione 7.4.2 Si definisce tempo minimo di completamento dell'intero progetto, e si indica con T , il tempo minimo di raggiungimento del nodo finale $T = t_f$.

In base alle regole di costruzione del diagramma reticolare di un progetto è facile convincersi che il tempo minimo di raggiungimento di un nodo i coincide con il peso del cammino massimo dal nodo iniziale al nodo i , dove i pesi degli archi sono dati dalle durate t_{ij} delle attività che essi rappresentano. Poiché il diagramma reticolare è aciclico e i nodi sono già numerati in modo topologico, è immediato applicare l'algoritmo per i cammini massimi su grafi aciclici al fine di calcolare i tempi di raggiungimento minimi.

Esempio 3. Consideriamo il diagramma reticolare del progetto P1, e associamo alle attività A, B, ..., I i seguenti tempi di esecuzione, espressi in giorni lavorativi :

tempo di esecuzione di A: $t_{01} = 5$
tempo di esecuzione di B: $t_{12} = 7$
tempo di esecuzione di C: $t_{13} = 3$
tempo di esecuzione di D: $t_{24} = 4$
tempo di esecuzione di E: $t_{25} = 5$
tempo di esecuzione di F: $t_{25} = 6$
tempo di esecuzione di G: $t_{46} = 2$
tempo di esecuzione di H: $t_{56} = 8$
tempo di esecuzione di I: $t_{67} = 2$.

Nella figura 7.4.1 i tempi di esecuzione delle attività sono stati associati ai rami del diagramma reticolare del progetto. Posto $t_0 = 0$, possiamo calcolare per i successivi nodi i tempi minimi di raggiungimento, espressi in giorni, utilizzando l' algoritmo dei cammini massimi.

per il nodo 1, $t_1 = t_0 + t_{01} = 0 + 5 = 5$
per il nodo 2, $t_2 = t_1 + t_{12} = 5 + 7 = 12$
per il nodo 3, $t_3 = t_1 + t_{13} = 5 + 3 = 8$
per il nodo 4, $t_4 = t_2 + t_{24} = 12 + 4 = 16$
per il nodo 5, $t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5, 8 + 6) = 17$
per il nodo 6, $t_6 = \max(t_4 + t_{46}, t_5 + t_{56}) = \max(16 + 2, 17 + 8) = 25$
per il nodo 7, $t_7 = t_6 + t_{67} = 25 + 2 = 27$;

avremo inoltre per il tempo minimo di completamento del progetto, $T = t_7 = 27$ giorni.

Esercizio 4. Consideriamo nuovamente il progetto P2 e associamo alle attività i seguenti tempi di esecuzione, espressi in settimane: A, 3 settimane; B, 2; C, 1; D, 4 ; E, 1; F, 2; G, . Determinare il tempo minimo di completamento del progetto.

Domanda 3 Con riferimento al progetto P1, supponiamo che il tempo di esecuzione dell'attività H passi da 8 a 10 giorni. Come varia il tempo minimo di completamento del progetto?

Domanda 4 Sempre con riferimento al progetto P1, supponiamo ora che il tempo di esecuzione dell'attività G passi da 2 a 6 giorni. Come varia il tempo minimo di completamento del progetto?

Oltre al tempo minimo di completamento dell'intero progetto, è utile introdurre anche la nozione di tempo minimo di completamento per ogni attività. Ovviamente un'attività (i, j) può avere inizio, al più presto, dopo un tempo t_i dall'inizio dell'esecuzione del progetto, in quanto perché l'attività possa avere inizio deve essere stato raggiunto il nodo i ; di conseguenza se l'attività richiede un tempo di esecuzione pari a t_{ij} , non potrà essere completata prima di un tempo pari a $t_i + t_{ij}$. Possiamo quindi dare la definizione seguente:

Definizione 7.4.3 Si definisce tempo minimo di completamento dell'attività (i, j) , e si indica con C_{ij} , il valore $C_{ij} = t_i + t_{ij}$.

Esempio 6 Consideriamo ancora il progetto P1. Per le attività del progetto, tenendo conto dei tempi di raggiungimento dei nodi calcolati nell'Esempio 5, abbiamo i seguenti tempi minimi di completamento espressi in giorni:

per l'attività A: $C_{01} = t_0 + t_{01} = 5$
per l'attività B: $C_{12} = t_1 + t_{12} = 5 + 7 = 12$
per l'attività C: $C_{13} = t_1 + t_{13} = 5 + 3 = 8$
per l'attività D: $C_{24} = t_2 + t_{24} = 12 + 4 = 16$
per l'attività E: $C_{25} = t_2 + t_{25} = 12 + 5 = 17$
per l'attività F: $C_{35} = t_3 + t_{35} = 8 + 6 = 14$

per l'attività G: $C_{46} = t_4 + t_{46} = 16 + 2 = 18$

per l'attività H: $C_{56} = t_5 + t_{56} = 17 + 8 = 25$

per l'attività I: $C_{67} = t_6 + t_{67} = 25 + 2 = 27$.

Nelle tecniche reticolari di programmazione sono di fondamentale importanza le cosiddette attività critica di cui diamo la definizione.

Definizione 7.4.4 *Sia T il tempo minimo di completamento di un progetto, corrispondente a un insieme $\{t_{ij}\}$ di valori prefissati dei tempi di esecuzione delle singole attività. Un'attività (h, k) viene detta attività critica se un variazione positiva comunque piccola ma non nulla del suo tempo di esecuzione comporta una variazione della stessa entità nel tempo minimo di completamento del progetto; e cioè, un'attività (h, k) è critica se, sostituito t_{hk} con $t_{hk} + \Delta t$, con $\Delta t \neq 0$ il tempo minimo di completamento del progetto diventa $T + \Delta t$, per qualunque valore positivo di Δt .*

Esempio 7 Nel progetto P1 l'attività E è critica: infatti se si pone $t_{25} = 5 + \Delta t$, si ottiene per i tempi di raggiungimento dei nodi 5, 6, 7 :

per il nodo 5, $t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5 + \Delta t, 8 + 6) = 17 + \Delta t$

per il nodo 6, $t_6 = \max(t_4 + t_{46}, t_5 + t_{56}) = \max(16 + 2, 17 + \Delta t + 8) = 25 + \Delta t$

per il nodo 7, $t_7 = t_6 + t_{67} = 25 + \Delta t + 2 = 27 + \Delta t$;

invece l'attività F non critica; infatti posto $t_{35} = 6 + \Delta t$, si ottiene per il tempo di raggiungimento del nodo 5 :

$$t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5, 8 + 6 + \Delta t) = 17$$

e cioè lo stesso valore di prima, almeno fintanto che Δt non supera i 3 giorni; e ovviamente se t_5 non varia, non variano neanche i tempi minimi di raggiungimento dei nodi successivi.

La determinazione dei percorsi critici è evidentemente di fondamentale importanza nelle tecniche reticolari di programmazione. Infatti le attività critiche sono quelle su cui più stretto deve essere il controllo di chi gestisce l'esecuzione del progetto, nei casi in cui un ritardo dell'esecuzione complessiva comporta una penalità, che può essere sia economica, sia di immagine. Da quanto abbiamo visto finora, l'individuazione delle attività critiche è molto semplice: è evidente che le attività critiche sono tutte e sole le attività che appartengono ad almeno un cammino massimo dal nodo iniziale al nodo finale.

Per quanto riguarda le attività non critiche è anche possibile stimare di quanto esse possano essere ritardate senza aumentare il tempo di completamento del progetto. Un'attività non critica, per esempio l'attività F nel progetto P1, non appartiene a un cammino massimo dal nodo iniziale al nodo finale, ma appartiene comunque a cammini che vanno dal nodo iniziale al nodo finale (nel caso specifico considerato uno solo, in generale più di uno). Sia allora L_{\max} la lunghezza massima di un percorso dal nodo iniziale al nodo finale che passa per F. È chiaro che l'attività F è l'unica a subire un ritardo, il ritardo massimo che è possibile tollerare senza che aumenti la durata del progetto è $T - L_{\max}$. Questo tempo è chiamato tempo di slittamento dell'attività. In generale:

Definizione 7.4.5 *Si definisce tempo di slittamento, o margine di tempo dell'attività (i, j) il valore che indica di quanto tempo può essere ritardato il completamento dell'attività (i, j) senza che si determini un aumento del tempo minimo di completamento dell'intero progetto.*

Ovviamente le attività critiche hanno tempo di slittamento nullo, le attività non critiche hanno un tempo di slittamento positivo. Segnaliamo che è possibile determinare in maniera molto semplice e efficiente i tempi di slittamento di tutte le attività di un progetto, noi non approfondiamo qui ulteriormente la questione per mancanza di tempo.

7.4.2 Gestione delle scorte.

Il problema di cui ci occuperemo in questo capitolo è il cosiddetto problema di Gestione delle Scorte. Si tratta di un problema di programmazione della produzione industriale che consiste nel decidere, dato un

particolare prodotto, le quantità da produrre o da immagazzinare in modo da soddisfare una domanda presente o futura. Tipicamente per la pianificazione si considera un numero ristretto di periodi di tempo, ad esempio i prossimi dodici mesi o le prossime 20 settimane. In generale indicheremo con $\{1, \dots, T\}$ l'orizzonte temporale, che è composto di T periodi detti *periodi di controllo*. Per ogni periodo è nota una domanda del bene d_t . Ad esempio, se il bene consiste in tondini metallici, per ogni giorno (o settimana o mese) del nostro orizzonte di pianificazione sono note le tonnellate richieste dal mercato. Ora, la domanda giornaliera di tondini deve essere integralmente soddisfatta. Io posso scegliere se produrre quotidianamente la quantità richiesta, oppure produrre solo in alcuni giorni (ad esempio una volta a settimana) una quantità sufficiente a soddisfare la domanda di più giorni successivi, mettendo in magazzino ciò che invierò al mercato nei giorni futuri. Ovviamente, sia produrre che immagazzinare costa, e il costo può variare da periodo a periodo. Di seguito indicheremo con c_t il costo unitario di produzione nel periodo t , mentre indicheremo con h_t il costo unitario di immagazzinamento, per $t = 1 \dots T$. Infine, iniziare un lotto di produzione presenta i cosiddetti costi di *setup*, ovvero costi amministrativi e di avviamento. Questi costi sono fissi, non dipendono cioè dalle quantità prodotte, e vengono effettivamente sostenuti solo se si produce qualcosa. Chiameremo con f_t i costi fissi di set-up relativi al periodo t , per $t = 1 \dots, T$.

Il problema è quello di scegliere quando (in che giorni) e quanto produrre in modo da minimizzare i costi di produzione (fissi e variabili) e i costi di immagazzinamento.

Una formulazione di PL01.

Introduciamo innanzitutto una variabile reale non-negativa p_t che indica il livello di produzione e una variabile reale non-negativa s_t che indica il livello delle scorte di magazzino per ogni periodo $t = 1, \dots, T$. Nel primo periodo il magazzino è vuoto (per ipotesi): quindi la domanda d_1 deve essere soddisfatta dalla produzione p_1 . Se $p_1 > d_1$, cioè produciamo più di quanto domandato, la parte residua finirà nel magazzino per servire la domanda dei periodi successivi. In particolare, avendo indicato con s_1 il livello delle scorte accumulate nel primo periodo, si avrà:

$$p_1 = d_1 + s_1 \quad (7.1)$$

Per i periodi successivi al primo, la quantità prodotta p_t nel periodo t si va ad aggiungere alle scorte accumulate nel periodo precedente s_{t-1} . Quindi, nel periodo t avremo a disposizione una quantità di bene pari a $p_t + s_{t-1}$. Una parte di questa quantità servirà a soddisfare la domanda d_t nel periodo t , mentre il resto rimarrà in magazzino e cioè sarà il livello di scorte s_t nel periodo t (utilizzabili nel periodo successivo). Quindi possiamo scrivere

$$p_t + s_{t-1} = d_t + s_t \quad t = 1, \dots, T \quad (7.2)$$

Per tenere conto del costo fisso di produzione f_t che va pagato solo nei periodi per cui si abbia $p_t > 0$, dobbiamo aggiungere una variabile booleana x_t per $t = 1, \dots, T$ con il seguente significato

$$x_t = \begin{cases} 1 & \text{se si produce nel periodo } t \text{ (} p_t > 0 \text{)} \\ 0 & \text{altrimenti.} \end{cases}$$

Per esprimere con un vincolo che $x_t = 1$ quando $p_t > 0$, o, equivalentemente, che possiamo produrre nel periodo t solo se $x_t = 1$ introduciamo il seguente vincolo:

$$p_t \leq M \cdot x_t \quad t = 1, \dots, T \quad (7.3)$$

dove M è una costante sufficientemente grande. Il ruolo del vincolo 7.3 è il seguente: se $x_t = 0$, allora non è possibile produrre ($p_t \leq 0 \rightarrow p_t = 0$). Se $x_t = 1$ allora $p_t \leq M$ e noi possiamo produrre una quantità compresa fra 0 e M . Come si è detto M deve essere scelto opportunamente in modo da rendere il vincolo sempre soddisfatto quando $x_t = 1$. Ad esempio, poichè in ogni periodo non ha senso produrre più di quanto viene richiesto durante tutto l'orizzonte temporale, un valore ammissibile per M può essere $M = \sum_{t=1}^T d_t$.

Siamo ora in grado di scrivere la funzione obiettivo, che si compone di tre termini:

- Il costo variabile di produzione, pari a $\sum_{t=1}^T c_t p_t$.
- Il costo fisso di produzione, pari a $\sum_{t=1}^T f_t x_t$.
- Il costo (variabile) di immagazzinamento, pari a $\sum_{t=1}^T h_t s_t$.

La funzione obiettivo risulta quindi essere:

$$\min \sum_{t=1}^T c_t p_t + f_t x_t + h_t s_t \quad (7.4)$$

Uno sguardo alla “struttura” del problema

Nelle righe precedenti abbiamo formulato il problema di gestione delle scorte come problema di programmazione mista. Per la soluzione del modello potremmo utilizzare algoritmi generali quali il branch-and-bound. Tuttavia, studiando più a fondo la struttura del problema, faremo vedere come sia possibile applicare algoritmi molto più efficienti per identificare una soluzione ottima.

Una prima osservazione riguarda la struttura delle soluzioni ottime. In particolare, si può facilmente dimostrare che esiste sempre una soluzione ottima per cui sia $p_t \cdot s_{t-1} = 0$ per $t = 1, \dots, T$. In altri termini, se decidiamo nel periodo t di soddisfare la domanda d_t con scorte provenienti dal periodo precedente non produrremo niente; di converso, se produciamo qualcosa ($p_t > 0$) allora il livello delle scorte precedenti s_{t-1} deve essere 0 e tutta la domanda sarà soddisfatta dalla produzione attuale. Quindi, possiamo distinguere i periodi di controllo in periodi in cui si produce (per cui $p_t > 0$ e $s_{t-1} = 0$) e periodi in cui non si produce (per cui $s_{t-1} > 0$ e $p_t = 0$). La domanda di ciascun periodo produttivo è soddisfatta dalla produzione, mentre nei periodi non produttivi la domanda è soddisfatta dalle scorte. Se i è un periodo produttivo e j è il prossimo periodo produttivo, la domanda relativa ai periodi compresi fra i e $j-1$ è servita dalla produzione realizzata in i . Quindi, in i io dovrò produrre $d_i + d_{i+1} + d_{i+2} \dots + d_{j-1} = \sum_{t=i}^{j-1} d_t$. Calcoliamo ora il costo complessivo $C(i, j)$ di produrre in i per soddisfare la domanda fino a j . Poiché la produzione in i sarà $p_i = \sum_{t=i}^{j-1} d_t$, il suo costo ammonterà a $c_i * \sum_{t=i}^{j-1} d_t$; a questo costo va' aggiunto il costo fisso di produzione del periodo i , f_i ; infine, bisogna calcolare periodo per periodo il livello delle scorte. Ricordiamo che le scorte nel periodo k sono determinate dall'equazione $s_k = p_k + s_{k-1} - d_k$ (e cioè le scorte che accumulo in un certo periodo sono pari alle scorte del periodo precedente, più la produzione nel periodo, meno ciò che se ne va per soddisfare la domanda del periodo). Quindi, nel periodo i le scorte saranno pari a $s_i = p_i + s_{i-1} - d_i$; essendo $p_i = d_i + d_{i+1} + \dots + d_{j-1} = \sum_{t=i}^{j-1} d_t$, ed avendo inoltre $s_{i-1} = 0$, sarà: $s_i = \sum_{t=i}^{j-1} d_t - d_i = \sum_{t=i+1}^{j-1} d_t$ e il suo costo sarà pari quindi $h_i * \sum_{t=i+1}^{j-1} d_t$. Nel periodo successivo, $i+1$ (che supponiamo momentaneamente non essere un periodo produttivo), il livello delle scorte sarà $s_{i+1} = p_{i+1} + s_i - d_{i+1}$, ed essendo $p_{i+1} = 0$, $s_i = \sum_{t=i+1}^{j-1} d_t$ avremo $s_{i+1} = \sum_{t=i+1}^{j-1} d_t - d_{i+1} = \sum_{t=i+2}^{j-1} d_t$, ed il costo sarà $h_{i+1} * \sum_{t=i+2}^{j-1} d_t$. Quindi, il costo complessivo delle scorte mantenute dal periodo i al periodo $j-1$ sarà pari a $h_i * \sum_{t=i+1}^{j-1} d_t + h_{i+1} * \sum_{t=i+2}^{j-1} d_t + \dots + h_{j-2} * d_{j-1}$. Riassumendo, il costo complessivo di produrre in i per soddisfare la domanda fino a $j-1$ sarà pari a $C(i, j) = c_i * \sum_{t=i}^{j-1} d_t + f_i + h_i * \sum_{t=i+1}^{j-1} d_t + h_{i+1} * \sum_{t=i+2}^{j-1} d_t + \dots + h_{j-2} * d_{j-1}$.

Ora, una soluzione (livello di produzione e livello di scorte in ogni periodo) è completamente determinata una volta stabilito l'insieme dei periodi produttivi (periodi in cui la produzione è positiva). Infatti, se $I = \{1, i, j, \dots, r, q\}$ è l'insieme ordinato dei periodi produttivi, allora vuol dire che si produce in 1 per soddisfare la domanda fino a $i-1$, quindi si produce in i per soddisfare la domanda fino a $j-1$, etc. etc., quindi si produce in r per soddisfare la domanda fino a $q-1$ e si produce in q per soddisfare la domanda fino a T . Il costo della soluzione sarà $C(I) = C(1, i) + C(i, j) + C(j, k) + C(k, \cdot) + \dots + C(\cdot, r) + C(r, q)$.

Quindi, il problema di gestione delle scorte può essere riformulato come il problema di trovare l'insieme di periodi produttivi I^* che minimizzi il costo $C(I^*)$.

Rappresentazione del problema di gestione delle scorte come problema di cammino minimo su grafo

Vediamo ora come il problema di trovare un insieme di istanti produttivi ottimale può essere agevolmente ricondotto a un problema di cammino minimo su grafi aciclici. A tal scopo definisco un grafo orientato $G = (V, E)$. L'insieme dei nodi è l'insieme dei periodi di controllo più un nodo fittizio che denominiamo $T + 1$, e cioè $V = \{1, \dots, T, T + 1\}$. L'insieme degli archi è l'insieme di tutti gli archi "in avanti", e cioè $E = \{(i, j) : i \in V, j \in V, i < j\}$. È facile che con questa definizione degli archi, l'insieme dei nodi risulta ordinato topologicamente, e cioè gli archi vanno sempre da nodi con indice più piccolo a nodi con indice più grande. Quindi, il grafo G è privo di cicli orientati (vedi il teorema 7.2.1). Associamo inoltre a ogni arco (i, j) il peso $p(i, j) = C(i, j)$.

Ora, a ogni cammino $P = \{1, i, j, k, \dots, r, q, T + 1\}$ dal nodo 1 al nodo $T + 1$ associamo l'insieme di periodi produttivi $I(P) = \{1, i, j, k, \dots, r, q\}$. Per quanto visto sopra, il costo $C(I) = C(1, i) + C(i, j) + C(j, k) + C(k, \dots) + \dots + C(\dots, r) + C(r, q)$; ma, per come sono stati definiti i pesi sugli archi di G , è facile vedere che $C(I) = p(P)$. Quindi, il peso del cammino minimo da 1 a $T + 1$ in G corrisponde al costo di un insieme di istanti produttivi di costo minimo.

Esempio

Si consideri l'esempio riportato in tabella.

Periodo	Domanda	f	c	h
1	20	30	3	2
2	30	40	3	2
3	40	30	4	1
4	30	50	4	1

Per prima cosa calcoliamo costruiamo il grafo associato all'istanza del problema. Si tratta di un nodo con 5 nodi, uno in più dei periodi di controllo: quindi $V = \{1, 2, 3, 4, 5\}$ e tutti gli archi "in avanti".

Per prima cosa calcoliamo i pesi degli archi. L'arco $(1, 2)$ ha peso pari a $C(1, 2)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $2 - 1 = 1$. Poiché la domanda è pari a $d_1 = 20$, il costo variabile di produzione sarà $c_1 d_1 = 3 \times 20 = 60$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Non ci sono scorte residue e quindi il costo complessivo $C(1, 2) = 60 + 30 = 90$. Calcoliamo ora il costo $C(1, 3)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $3 - 1 = 2$. La domanda complessiva sarà $d_1 + d_2 = 50$. Quindi il costo variabile di produzione sarà pari a $c_1(d_1 + d_2) = 3 \times 50 = 150$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Inoltre, una parte della produzione (d_2) dovrà essere immagazzinata per un periodo (da 1 fino a 2), e quindi pagheremo un costo d'immagazzinamento pari a $s_1 * d_2 = 2 \times 30 = 60$. Quindi, il costo complessivo $C(1, 3) = 150 + 30 + 60 = 230$. Calcoliamo ora il costo $C(1, 4)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $4 - 1 = 3$. La domanda complessiva sarà $d_1 + d_2 + d_3 = 90$. Quindi il costo variabile di produzione sarà pari a $c_1(d_1 + d_2 + d_3) = 3 \times 90 = 270$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Inoltre, una parte della produzione ($d_2 + d_3$) dovrà essere immagazzinata nel periodo 1 fino al periodo 2, e un'altra parte d_3 dovrà essere immagazzinata nel periodo 2 fino al periodo 3 e quindi pagheremo un costo d'immagazzinamento pari a $s_1(d_2 + d_3) + s_2 d_3 = 2 \times 70 + 2 \times 40 = 220$. Quindi, il costo complessivo $C(1, 4) = 270 + 30 + 220 = 520$. A titolo esemplificativo, calcoliamo un ultimo peso, il peso dell'arco $(3, 5)$ corrispondente a $C(3, 5)$.

A questo punto, applichiamo l'algoritmo per il calcolo di un cammino minimo su grafi aciclici per calcolare un cammino di peso minimo dal nodo 1 al nodo 5.

Figura 7.17: Il grafo dei periodi

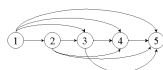


Figura 7.18: Calcolo dei pesi sugli archi



- $f(1) = 0$;
 $J(1) = 1$;
- $f(2) = f(1) + p(1, 2) = 90$;
 $J[2] = 1$.
- $f(3) = \min\{f(1) + p(1, 3), f(2) + p(2, 3)\} = \min\{240, 220\} = 220$;
 $J(3) = 2$.
- $f(4) = \min\{f(1) + p(1, 4), f(2) + p(2, 4), f(3) + p(3, 4)\} = \min\{520, 420, 410\} = 410$;
 $J(4) = 4$.
- $f(5) = \min\{f(1) + p(1, 5), f(2) + p(2, 5), f(3) + p(3, 5), f(4) + p(4, 5)\} = \min\{760, 600, 560, 580\} = 560$;
 $J(5) = 3$.

A questo punto è possibile ricostruire il cammino minimo da 1 a 5 utilizzando il vettore dei predecessori. Infatti, il predecessore di 5 è il nodo 3, il predecessore di 3 è il nodo 2 e il predecessore di 2 è il nodo 1.

Figura 7.19: Cammino minimo da 1 a 5

