Cammini e cicli

Sandro Bosio

Dottorando in Ingegneria Matematica, XVIII^o ciclo
Corso di Matematica Discreta
e-mail: sandro.bosio@polimi.it

Indice

Considerazioni preliminari

Quasi tutte le definizioni e gli algoritmi sono descritti relativamente ai digrafi, tuttavia laddove non esplicitato sono estendibili ai grafi.

Nel seguito si considerano nodo e vertice sinomini del tutto intercambiabili. La dicitura corretta riguardo agli archi in caso di grafi non orientati è "spigolo", tuttavia ho deciso di utilizzare sempre il termine arco in quanto il contesto chiarisce il tipo di grafo a cui si fa riferimento.

Nello pseudocodice, con `dist[]` $:= \alpha$ si intende che tutti gli elmenti del vettore sono inizializzati al valore α. Le operazioni a`++` e a`--` incrementano e decrementano di una unità la quantità a. Si intende inoltre l'indentazione come elemento raggruppatore in blocchi delle istruzioni, senza dover esplicitamente indicare l'inizio e la fine di ogni blocco.

1 Cammini

Definizione 1.1.
Dato un digrafo $D = (V, A)$, per ogni nodo $q \in V$ si definiscono:

- $d^-(q) = |\{(p, q) \in A\}|$ grado entrante di q
- $d^+(q) = |\{(q, p) \in A\}|$ grado uscente di q

Risulta ovvio che $\sum_{q \in V} d^-(q) = \sum_{q \in V} d^+(q) = |A|$.
In modo simile su un grafo si definisce il grado di un nodo.

Definizione 1.2.
Dato un digrafo $D = (V, A)$, sia $G(D) = (V, E)$ il grafo ottenuto togliendo l'orientazione agli archi e collassando gli archi paralleli, cioè dove $E = \{\{p, q\} \mid \forall q, p \in V, \quad (p, q) \in A \ \vee \ (q, p) \in A\}$.

Definizione 1.3.
Dato un digrafo $D = (V, A)$ e due nodi $s, t \in V$ qualsiasi, si definisce *cammino* da s a t una sequenza di archi

$$(a_1 = (v_1, v_2), \ldots, a_{k-1} = (v_{k-1}, v_k))$$

tale per cui $v_1 = s$, $v_k = t$, e $a_j \in A \quad \forall j \in \{1, \ldots, k-1\}$. Spesso un cammino è indicato con la corrispondente sequenza di vertici $(v_1, \ldots, v_k)$.

Si dice che un cammino *tocca* o *visita* un nodo o un arco se contiene nella sequenza tale nodo o arco almeno una volta.

Allo stesso modo su un grafo $G = (V, E)$ si definisce una *catena*.

Definizione 1.4.

La *lunghezza* di un cammino è il numero di archi (non distinti) che lo compone ed è pari al numero di vertici (non distinti) meno uno.

Definizione 1.5.

Su un digrafo si definisce *circuito* un cammino chiuso, ovvero un cammino in cui il primo e l'ultimo nodo coincidono.

Su un grafo si definisce *ciclo* una catena chiusa.

In alcuni testi si applica un'estensione della definizione di ciclo ai digrafi, dove viene definito ciclo su D una sequenza di vertici che forma un ciclo sul corrispondente grafo $G(D)$.

Nella maggior parte dei testi invece il termine ciclo è ormai diventato un sinonimo di circuito, ed è questa convenzione quella adottata nel seguito.

Definizione 1.6.

Un grafo si dice *connesso* se presi comunque due nodi s e t esiste un cammino da s a t.

Un digrafo si dice *connesso* se il grafo indotto dal digrafo risulta connesso, ovvero se presa una qualsiasi partizione di V in due insiemi non vuoti A, B, esiste un cammino da uno dei due insiemi verso l'altro.

Un digrafo si dice *fortemente connesso* se presi comunque due nodi s e t esiste un cammino da s a t.

Osservazione 1.1.

Un digrafo fortemente connesso contiene almeno un ciclo che tocca tutti i nodi.

2 Cammini con proprietà particolari

2.1 Cammini semplici e cammini elementari

Definizione 2.1.

Un cammino $(v_1, \ldots, v_k)$ si definisce *elementare* se i suoi vertici sono tutti distinti tra loro (tranne eventualmente v_1 e v_k), ovvero se non visita due volte uno stesso nodo.

Definizione 2.2.
Un cammino si definisce *semplice* se gli archi che lo compongono sono tutti distinti tra loro, ovvero se non visita due volte uno stesso arco.

Osservazione 2.1.
Un cammino elementare è anche semplice, mentre un cammino semplice può non essere elementare.

Definizione 2.3.
Un ciclo $(v_1, \ldots, v_k = v_1)$ si definisce *elementare* se i suoi vertici sono tutti distinti tra loro, tranne ovviamente v_1 e v_k.
Alternativamente un ciclo è elementare se e solo se non è suddivisibile in più cicli, ovvero se in ogni nodo visita un solo arco entrante ed un solo arco uscente.

Definizione 2.4.
Un ciclo si definisce *semplice* se i suoi archi sono tutti distinti.

Osservazione 2.2.
Dato un cammino da s a t si può sempre ricavare un cammino elementare da s a t. Lo stesso vale per i cicli.

Osservazione 2.3.
Un cammino semplice si può sempre decomporre in un cammino elementare più un insieme, eventualmente vuoto, di cicli semplici.

Osservazione 2.4.
Un ciclo semplice si può sempre decomporre in cicli elementari.

Osservazione 2.5.
Se un digrafo non possiede cicli tutti i cammini al suo interno sono elementari.

Nota: non tutti i testi sono conformi a questa nomenclatura; in alcuni testi semplice è un sinonimo di elementare e non viene data nessuna definizione per descrivere cammini in cui gli archi non siano ripetuti, mentre in altri testi tali cammini vengono definiti *semi-semplici*.

2.2 Cammini hamiltoniani

Definizione 2.5.
Un cammino hamiltoniano è un cammino elementare che visita tutti i nodi, mentre un ciclo hamiltoniano è un ciclo elementare che visita tutti i nodi.

Osservazione 2.6.
Se un digrafo ha un ciclo hamiltoniano, allora ha un cammino hamiltoniano, ma in generale il viceversa non è vero.

Il problema HC di trovare, se esiste, un ciclo hamiltoniano in un digrafo è stato formulato nel 1859 dal matematico Sir William Hamilton ed è $\mathcal{NP}$-completo, così come lo è il problema HP di trovare un cammino hamiltoniano per una coppia di nodi.

Teorema 2.1.
HC è $\mathcal{NP}$-completo se e solo se HP lo è.

Dimostrazione:
Se avessimo un algoritmo polimoniale per risolvere HP su un digrafo, potremmo cercare un cammino hamiltoniano per ogni coppia di nodi s, t per cui esiste l'arco (t, s). Se il digrafo possiede un ciclo hamiltoniano tale coppia di nodi deve esistere.

Viceversa se avessimo un algoritmo polinomiale che trova un ciclo hamiltoniano su un digrafo potremmo, rimuovendo tutti gli archi entranti in s e quelli uscenti da t e aggiungendo l'arco (t, s), verificare se esiste un ciclo hamiltoniano, in cui avremmo necessariamente s e t adiacenti.

Da notare che se trattiamo con i grafi non si può fare questa operazione, ma si deve generare un numero polinomiale di grafi in cui si obbliga il passaggio per l'arco (t, s) lasciando un solo altro arco sia a t che a s, risolvendo il problema HC su tutti questi grafi separatamente.

Nonostante non esitano algoritmi efficienti per trovare cicli o cammini hamiltoniani, tranne che per particolari categorie di grafi (come chiaramente i grafi completi), sono state trovate un certo numero di condizioni necessarie e non sufficienti perché un digrafo contenga un ciclo o un cammino hamiltoniano.

2.3 Cammini euleriani

Definizione 2.6.

Un cammino *euleriano* è un cammino semplice che tocca tutti gli archi del digrafo, mentre un ciclo *euleriano* è un ciclo semplice che tocca tutti gli archi del digrafo.

Osservazione 2.7.

Dalla definizione di cammini, o cicli, euleriani segue che se un digrafo $D = (V, A)$ ha un cammino (o un ciclo) euleriano $(v_1, \ldots, v_k)$ allora per ogni arco $(p, q) \in A$ esiste unico l'indice $i \in \{1, \ldots, k\}$ tale per cui $v_i = p \ \wedge \ v_{i+1} = q$.

Il problema di determinare se un digrafo (o un grafo) ha un cammino o un ciclo euleriano è stato formulato e risolto da Eulero nel 1736, con il problema dei ponti di Konigsberg.

Teorema 2.2.

In un digrafo $D = (V, A)$, se $d^-(q) = d^+(q) > 0 \quad \forall q \in V$ allora da ogni nodo parte un ciclo semplice di lunghezza positiva.

Dimostrazione:

Iniziamo un cammino da un nodo qualsiasi q, e ad ogni passo eliminiamo da D gli archi visitati. Ad ogni nodo p del cammino in cui entriamo, poiché nel digrafo originale $d^-(p) = d^+(p)$, deve esistere un nodo verso cui muoversi. Poiché il numero di archi è finito prima o poi si deve tornare in q.

Teorema 2.3.

Sia un digrafo $D = (V, A)$ connesso. D possiede un ciclo euleriano se e solo se $d^-(q) = d^+(q) \quad \forall q \in V$

Dimostrazione:

(solo se:) Se contiamo nei gradi dei nodi solo gli archi toccati da un qualsiasi ciclo, per tali nodi vale $d^-(q) = d^+(q)$. Poiché un ciclo euleriano tocca tutti gli archi, tale conteggio equivale agli effettivi gradi dei nodi.

Dimostrazione:

(se:) Prendiamo un nodo qualsiasi q, e sia π un ciclo semplice che parte da q di lunghezza positiva, la cui esistenza è garantita dal teorema (2.2).

Se π non è un ciclo euleriano, allora sia $\tilde{D} = (\tilde{V}, \tilde{A})$ il digrafo che si ottiene togliendo da D tutti gli archi del ciclo, e tutti i nodi senza più archi.

Poiché in ogni nodo di un ciclo si usano un uguale numero di archi entranti ed uscenti, vale ancora $d^-(p) = d^+(p) \quad \forall p \in \tilde{V}$.

Inoltre poiché D era connesso, l'intersezione tra i nodi del ciclo e $\tilde{V}$ non può essere vuota, altrimenti i nodi del ciclo formerebbero una componente non connessa al resto del digrafo, ovvero a nessuno dei nodi di $\tilde{V} \neq \emptyset$.

Sia allora z un nodo del ciclo contenuto in $\tilde{V}$, e sia π_z un ciclo semplice di lunghezza positiva che parte da z sul grafo $\tilde{D}$.

Si può costruire un ciclo semplice più lungo di π inserendo π_z in un punto qualsiasi in cui z viene visitato in π.

A questo punto si ha un nuovo ciclo semplice su D, più lungo del precedente; o è un ciclo euleriano, oppure si ripete la procedura. Poiché ad ogni passo il ciclo diventa più lungo, e poiché il numero di archi è finito, in un numero finito di passi otterremo un ciclo euleriano.

Teorema 2.4.

Sia un digrafo $D = (V, A)$ connesso. D possiede un cammino euleriano da s a t, con $s, t \in V, s \neq t$, se e solo se

- *(1)* $d^-(s) = d^+(s) - 1$
- *(2)* $d^-(t) = d^+(t) + 1$
- *(3)* $d^-(q) = d^+(q) \quad \forall q \in V \setminus \{s, t\}$

Dimostrazione:

Si costruisca il digrafo $\tilde{D}$ aggiungendo un nodo z e i due archi $(t, z), (z, s)$. Su questo nuovo digrafo vale $d^-(q) = d^+(q) \quad \forall q \in \tilde{V}$. A questo punto la dimostrazione segue dal teorema (2.3). Ottenuto il ciclo euleriano, rimuovendo dal ciclo il nodo z si ottiene il cammino euleriano desiderato.

Osservazione 2.8.

Non è possibile aggiungere direttamente l'arco (t, s) perché un digrafo potrebbe già possederlo e comunque avere un cammino euleriano da s a t.

Osservazione 2.9.

Dai gradi entranti e uscenti del teorema (2.4) si deduce che in un digrafo possono eventualmente esistere diversi cammini euleriani distinti, ma tutti avranno in comune nodo iniziale e nodo finale.

Si noti inoltre che è possibile estendere i teoremi di cui sopra al caso di digrafi in cui ci sia una componente connessa e dei vertici isolati. In tal caso basta rimuovere tali vertici e applicare direttamente i teoremi definiti su digrafi connessi.

Esistono versioni di questi teoremi anche per grafi non diretti, le cui dimostrazioni sono simili alle precedenti e perciò omesse.

Teorema 2.5.

Un grafo connesso contiene un ciclo euleriano se e solo se tutti i nodi hanno grado pari.

Teorema 2.6.

Un grafo connesso contiene un cammino euleriano da s a t, con $s \neq t$, se e solo se tutti i nodi tranne s e t hanno grado pari.

2.4 Relazioni tra cammini euleriani e hamiltoniani

Osservazione 2.10.

Un cammino hamiltoniano che tocca tutti gli archi di un digrafo è anche un cammino euleriano.

Definizione 2.7.

Consideriamo grafi non orientati. Il grafo *commutato* (line graph) di un grafo $G = (V, E)$ è il grafo $G^* = (V^*, E^*)$ in cui c'e' un nodo v_e per ogni spigolo in E e uno spigolo (v_{e_1}, v_{e_2}) per ogni coppia di spigoli e_1, e_2 adiacenti in G (ovvero incidenti su uno stesso nodo).

Teorema 2.7.

Se un grafo G ha un ciclo euleriano, allora G^* ha un ciclo hamiltoniano.

Dimostrazione:

Sia $c = (a_1, \ldots, a_{|E|})$ un ciclo euleriano su G.

Consideriamo la sequenza di nodi di G^* data da $\pi = (v_{a_1}, \ldots, v_{a_{|E|}}, v_{a_1})$. Poiché ogni a_i è adiacente a a_{i+1} e a_{i-1} in G (modulo $|E|$) ne risulta che π è un ciclo su G^*; poiché gli a_i sono distinti tra loro in G ne risulta che π è

un cammino elementare; infine poiché tale ciclo tocca tutti i vertici di G^* ne risulta che π è un ciclo hamiltoniano.

Teorema 2.8.
Se un grafo G ha un ciclo euleriano, allora G^* ha un ciclo euleriano.

Dimostrazione:
Per il teorema (2.5) in ogni nodo di G incide un numero pari di archi, pertanto ogni arco di G che non sia un autoanello è adiacente a un numero dispari di archi distinti in ognuno dei suoi estremi, per un totale di archi in numero pari.

Un autoanello in G deve essere incidente ad un numero pari di archi, poiché nel grado conta due volte.

Pertanto G^* è connesso poiché G, e ogni suo vertice ha grado pari. Quindi esiste un cammino euleriano su G^*.

Teorema 2.9.
Se un grafo G ha un ciclo hamiltoniano, allora anche G^* ha un ciclo hamiltoniano.

Dimostrazione:
Sia $c = (a_1, \ldots, a_{|V|})$ un ciclo hamiltoniano su G. Ora, se consideriamo la sequenza $\pi = (v_{a_1}, \ldots, v_{a_{|V|}}, v_{a_1})$ otteniamo un ciclo elementare, ma che non tocca necessariamente tutti i nodi di G^*, poiché nel ciclo hamiltoniano non erano usati necessariamente tutti gli archi.

Prendiamo un nodo q di G in cui incidano archi non ancora toccati da π nel grafo commutato. Ci saranno due archi a_i, a_{i+1} incidenti a q in G scelti nel ciclo hamiltoniano, e quindi ci sarà in π ad un certo punto $\ldots, v_{a_i}, v_{a_{i+1}}, \ldots$. Tuttavia poiché tutti gli archi incidenti su q sono nodi tra loro adiacenti nel grafo G^* (cioè formano un sottografo completo), è possibile trovare un cammino elementare in G^* da v_{a_i} a $v_{a_{i+1}}$ che tocca tutti e soli tali nodi. Si può quindi inserire questo cammino in π, ottenendo un ciclo elementare più lungo.

Questa operazione può essere ripetuta finché tutti i vertici di G^* non siano stati inseriti nel ciclo, fino ad ottenere un ciclo hamiltoniano.

Osservazione 2.11.
Il grafo commutato di G^* non è G, quindi non si può invertire i teoremi (2.7), (2.8), (2.9).

3 Cammini ottimi

Nel problema del cammino ottimo si chiede di trovare il cammino, tra tutti i cammini aventi certe proprietà, che minimizza o massimizza una certa funzione obiettivo.

Definizione 3.1.
Dato un digrafo $D = (V, A)$ e una funzione di peso $w : A \to R$, il peso di un cammino è definito come la somma dei pesi dei suoi archi, ovvero

$$w((v_1, \ldots, v_k)) = \sum_{i=1}^{k-1} w((v_i, v_i + 1))$$

Da notare che il cammino (v_1, v_2) di lunghezza 1 è identificabile con l'arco (v_1, v_2), quindi la definizione non introduce incoerenze.

3.1 Cammini minimi

Definizione 3.2.
Sia $\mathcal{C}$ l'insieme di tutti i cammini di un digrafo.

Definizione 3.3.
Un cammino $c = (v_1, \ldots, v_k)$ si dice *minimo* rispetto a una funzione w se non esiste un altro cammino da v_1 a v_k di peso minore, ovvero se vale $w(c) \leqslant w(h) \quad \forall h \in \mathcal{C}$.

Osservazione 3.1.
Data una funzione peso w e un cammino $c = (v_1, \ldots, v_k)$, allora per ogni $1 < i < k$ vale $w(c) = w(v_1, \ldots, v_i) + w(v_i, \ldots, v_k)$

Allo stesso modo dati due cammini qualsiasi $c_1 = (v_1, \ldots, v_i)$ e $c_2 = (v_i, \ldots, v_k)$ si può costruire il cammino $c = (v_1, \ldots, v_i, \ldots, v_k)$ che ha peso $w(c) = w(c_1) + w(c_2)$

Le due osservazioni sono dirette conseguenze di come è stato definito il peso di un cammino.

Teorema 3.1.

Sia $c = (v_1, \ldots, v_k)$ *un cammino minimo da* v_1 *a* v_k. *Allora per ogni* $1 < i < k$ *i cammini* $(v_1, \ldots, v_i)$ *e* $(v_i, \ldots, v_k)$ *sono minimi.*

Dimostrazione:

Se anche solo uno dei due cammini non fosse minimo, come diretta conseguenza dell'osservazione (3.1) potremmo costruire un cammino da v_1 a v_k di peso minore di c, e questo va contro all'ipotesi che c sia minimo.

Definizione 3.4.

Dato un peso w, si definisce la distanza $d(s,t)$ come il peso del cammino minimo da s a t rispetto a w.

Se due nodi non sono connessi si definisce $d(s,t) = \infty$.

Osservazione 3.2.

Dato un digrafo $D = (V, A)$, per ogni $s, q, t \in V$ vale $d(s,q) + d(q,t) \geqslant d(s,t)$.

Osservazione 3.3.

Dato un digrafo $D = (V, A)$, per ogni $s, q, t \in V \mid (q,t) \in A$ vale $d(s,q) + w(q,t) \geqslant d(s,q) + d(q,t) \geqslant d(s,t)$.

3.2 Cammino minimo con archi di peso non negativo

Dati un digrafo $D = (V, A)$, $s, t \in V$ e una funzione di peso $w : A \to R^+$ si chiede di trovare, se esiste, il cammino di peso minimo da s a t.

Algoritmo di Dijkstra

L'algoritmo di Dijkstra risolve il problema di trovare il cammino minimo da un nodo s a tutti gli altri nodi.

L'input dell'algoritmo sono il digrafo $D = (V, A)$, la funzione di peso w e il nodo sorgente s. L'output sono il vettore `dist` delle distanze di ogni nodo da s, e il vettore di precedenza `prev`, in cui `prev`$[i]$ indica il nodo che precede i nel cammino ottimo da s a i. Quindi il cammino

$$\tilde{c}_t = (s = \texttt{prev}[v_1], v_1 = \texttt{prev}[v_2], \ldots, v_{k-1} = \texttt{prev}[v_k], v_k = t)$$

è un cammino ottimo da s a t.

```
function DIJKSTRA(V, A, s, w)
    // Inizializzazione
    dist[] := ∞
    prev[] := -1;
    dist[s] := 0
    V̄ := V
    // Ciclo principale
    while ( V̄ ≠ ∅ )
        q := arg min_{i∈V̄} dist[i]
        if ( dist[q]=∞ )
            break
        V̄ := V̄ \ {q}
        for ⟨ k ∈ V | (q,k) ∈ A ⟩
            if ( dist[k]>dist[q]+w(q,k) )
                dist[k] := dist[q]+w(q,k)
                prev[k] := q
```

Ad ogni passo l'algoritmo cerca tra i nodi attivi il nodo di distanza minima, lo imposta come non piu' attivo e cerca di aggiornare le etichette dei nodi da lui raggiungibili.

L'algoritmo termina se $\overline{V} = \emptyset$, ovvero se ha fissato tutti i nodi, oppure se viene scelto un nodo q con distanza da s infinita, e in questo caso significa che tutti i nodi rimasti in $\overline{V}$ non sono raggiungibili da s.

Teorema 3.2.
Ad ogni passo $\texttt{dist}[i] \geq d(s,i) \quad \forall i \in V$

Dimostrazione:

Dimostriamo per induzione sulle iterazioni dell'algoritmo. La base d'induzione è banale, poiché all'inizio $\texttt{dist}[i] = \infty \geqslant d(s,i)$ per $i \neq s$ e $\texttt{dist}[s] = 0 = d(s,s)$.

Consideriamolo vero al passo $t-$esimo. Sia q il nodo prescelto, e consideriamo un nodo k tale che $(q,k) \in A$ e $\texttt{dist}[k] > \texttt{dist}[q] + w(q,k)$ (ovvero un nodo per cui l'etichetta decresca). Abbiamo quindi $\texttt{dist}[k] = \texttt{dist}[q] + w(q,k) \geqslant d(s,q) + w(q,k) \geqslant d(s,k)$.

Teorema 3.3.
Quando il nodo q viene scelto dall'algoritmo vale $\texttt{dist}[q] = d(s,q)$.

Dimostrazione:

La dimostrazione viene fatta per induzione sui nodi fissati: si suppone che sia vero per tutti i nodi fissati precedentemente a q, e si dimostra per assurdo che è vero per il nodo q. La base di induzione è il primo nodo fissato, s, per cui ovviamente vale $\texttt{dist}[s] = d(s, s) = 0$.

Supponiamo per assurdo che $d(s, q) < \texttt{dist}[q]$, ovvero che esista un cammino da s a q di peso minore di $\texttt{dist}[q]$. Sia $(p_1 = s, \ldots, p_k = q)$ tale cammino (supponiamo anche che sia minimo). Oltre al nodo q in questo cammino devono esistere altri nodi non ancora fissati, altrimenti avremmo $d(s, q) = d(s, p_{k-1}) + w(p_{k-1}, q) = \texttt{dist}[p_{k-1}] + w(p_{k-1}, q) \geqslant \texttt{dist}[q]$.

Sia p_r il primo vertice non fissato del cammino (quindi $1 < r < k$), e si consideri il cammino $(p_1 = s, \ldots, p_r)$. Tale cammino è minimo per il teorema (3.1). Ma poiché tutti i nodi prima di p_r sono stati fissati, significa che $\texttt{dist}[p_r] \leqslant \texttt{dist}[p_{r-1}] + w(p_{r-1}, p_r) = d(s, p_{r-1}) + w(p_{r-1}, p_r) = d(s, p_r) \leqslant d(s, q) < \texttt{dist}[q]$, ovvero in definitiva che $\texttt{dist}[p_r] < \texttt{dist}[q]$. Ma questo è un assurdo per come è stato scelto q.

3.3 Cammino minimo con archi di peso qualsiasi

Dati un digrafo $D = (V, A)$, $s, t \in V$ e una funzione di peso $w : A \to R$ si chiede di trovare, se esiste, il cammino di peso minimo da s a t.

Se esiste un cammino da s a t che passa per un vertice appartenente ad un ciclo di peso negativo, si definisce $d(s, t) = -\infty$. Infatti ripetendo i passaggi per tale ciclo un numero arbitrario di volte si può ottenere cammini da s a t di peso sempre minore.

Anche nel caso che non esista un ciclo di peso negativo, l'algoritmo di Dijkstra non funziona perché suppone che un etichetta possa decrescere ma mai diventare più piccola di quella del nodo da cui avviene l'aggiornamento; in un caso simile infatti non si può garantire che per i nodi gia' fissati valga $\texttt{dist}[q] = d(s, q)$.

Esistono algoritmi che risolvono il problema solo se non ci sono cicli di peso negativo, e algoritmi che invece riescono anche a individuare eventuali cicli di peso negativo. Tutti questi algoritmi, nelle loro differenti implementazioni, hanno complessità differenti e a seconda dell'applicazione può risultare conveniente applicarne uno piuttosto che un altro.

Algoritmo di Floyd.

L'algoritmo di Floyd risolve il problema di trovare il cammino minimo da ogni nodo a ogni altro nodo (se esiste) ed è in grado di individuare cicli di peso negativo.

Durante le varie iterazioni k aggiorna una matrice di etichette `dist`[][] e alla fine si ha che $\texttt{dist}[i][j] = d(i,j) \quad \forall i,j \in V$

```
function FLOYD(V, A, w)
    // Inizializzazione
    dist[][] := ∞
    for ⟨ q ∈ V ⟩
        dist[q][q] := 0
        for ⟨ p ∈ V | (q,p) ∈ A ⟩
            dist[q][p] := w(q,p)
    // Ciclo principale
    for ⟨ k ∈ V ⟩
        for ⟨ i ∈ V ⟩
            for ⟨ j ∈ V ⟩
                if ( dist[i][k]+dist[k][j]<dist[i][j] )
                    dist[i][j] := dist[i][k]+dist[k][j]
```

Siano $\texttt{dist}^k[p][q]$ le etichette alla fine del passo k.

Teorema 3.4.

Le etichette $\texttt{dist}^k[p][q]$ *sono il valore del cammino minimo che usa come vertici intermedi solo i nodi* $1, \ldots, k$.

Dimostrazione:

Dimostrazione per induzione.

Per $k = 0$ l'ipotesi è verificata per come sono state inizializzate le etichette: $\texttt{dist}^0[p][q]$ è il valore del cammino minimo che non usa vertici intermedi, quindi il peso dell'arco (p,q) dove questo esiste, e ∞ dove i due vertici non sono adiacenti.

Supponiamo che sia vero al passo $k-1$ e mostriamo che è vero al passo k. Consideriamo due nodi qualsiasi p e q, e sia π il cammino ottimo che usa come vertici intermedi $1, \ldots, k$. Allora o π non contiene k, e in tal caso $\texttt{dist}^k[i][j] = \texttt{dist}^{k-1}[i][j]$, oppure lo contiene.

Se π contiene k, possiamo dividere π nel punto in cui lo tocca in due cammini ottimi π_1 da p a k e π_2 da k a q, in cui vengono usati come nodi intermedi solo i nodi $1, \ldots, k-1$.

Abbiamo quindi $w(\pi) = w(\pi_1) + w(\pi_2) = \texttt{dist}^{k-1}[p][k] + \texttt{dist}^{k-1}[k][q] = \texttt{dist}^{k}[p][q]$.

Nel caso in cui esistano cicli di peso negativo, per ogni nodo i del ciclo ad una certa iterazione comparirà un termine $\texttt{dist}[i][i]$ negativo.

L'algoritmo può essere modificato in modo da poter ricostruire ogni cammino.

3.4 Cammino minimo elementare

Se la funzione di peso è non negativa, ogni cammino minimo è elementare. Se invece la funzione di peso è qualsiasi, allora il problema di trovare un cammino minimo elementare tra una coppia di nodi diventa $\mathcal{NP}$-completo.

Si consideri il problema $\mathcal{NP}$-completo di verificare se un grafo $G = (V, E)$ possiede un cammino hamiltoniano da un nodo s a un nodo t.

Si prenda come funzione peso la funzione costante $w(e) = -1 \quad \forall e \in E$. Se e solo se il cammino minimo elementare da s a t rispetto a w ha peso $1 - |V|$ allora il digrafo possiede un cammino hamiltoniano da s a t.

3.5 Problema del commesso viaggiatore

Il problema del commesso viaggiatore (o Travelling Salesman Problem, TSP) consiste nel trovare un ciclo hamiltoniano minimo.

La formulazione da cui prende il nome consiste, dato un insieme di città e un distanza tra ogni coppia di città, nel trovare la sequenza che permette di visitarle tutte una e una sola volta, e ritornare nella città di partenza, compiendo meno strada possibile.

Trovare un ciclo hamiltoniano in un digrafo è un problema $\mathcal{NP}$-completo, e a maggior ragione lo è il trovare un ciclo hamiltoniano minimo. Allo stesso modo anche il problema di trovare un cammino hamiltoniano minimo è $\mathcal{NP}$-completo.

Il TSP è uno dei problemi più studiati in ottimizzazione combinatoria, ed esistono diversi risultati che permettono di trovare soluzioni quasi ottime.

3.6 Problema del postino cinese

Il problema del postino cinese (o Chinese Postman Problem, CPP) consiste nel trovare il ciclo di peso minimo che visita tutti gli archi almeno una volta.

La formulazione da cui prende il nome consiste, dato un insieme di vie di una città e la loro lunghezza, nel trovare la sequenza che permette di visitarle tutte almeno una volta, tornando al punto di partenza e compiendo meno strada possibile.

Se il digrafo contiene un ciclo euleriano allora tale ciclo è ovviamente la soluzione del problema, poiché qualsiasi altro ciclo che visita tutti gli archi almeno una volta ha un peso maggiore (il peso è positivo). Se tale ciclo non esiste, allora è necessario visitare qualche arco più di una volta.

Il CPP è molto simile al TSP nella sua formulazione, così come cicli euleriani e cicli hamiltoniani sono tra loro legati. Tuttavia mentre il TSP è un problema difficile, il CPP, così come il problema di trovare un ciclo euleriano, è risolvibile in tempo polinomiale, tramite una riduzione al problema di trovare un matching di cardinalità massima su un appropriato grafo di supporto.

3.7 Cammino minimo su digrafi aciclici

Il caso di grafi diretti aciclici (DAG) è particolarmente interessante, perché in questi casi è possibile ottenere algoritmi di complessità lineare, qualunque sia la funzione di peso w.

Teorema 3.5.
In un digrafo aciclico esiste almeno un nodo con grado uscente 0.

Dimostrazione:

Supponiamo per assurdo che tutti i nodi abbiano grado uscente maggiore o uguale a 1. Consideriamo il cammino che parte da un qualsiasi nodo e si muove su uno qualsiasi degli archi uscenti. Poiché il digrafo è aciclico tale cammino deve essere elementare e poiché ogni nodo in cui ha grado uscente maggiore o uguale a 1 si può sempre prolungare tale cammino, il che porta all'assurdo che il numero di nodi distinti è infinito.

Teorema 3.6.

In un digrafo aciclico esiste almeno un nodo con grado entrante 0.

Dimostrazione:

Invertiamo la direzione di tutti gli archi. Poiché il digrafo risultante è ancora aciclico (un ciclo rimane tale invertendo la direzione di tutti gli archi) esiste un nodo con grado uscente 0, e quindi esiste nel digrafo originale un nodo con grado entrante 0.

Teorema 3.7.

Un digrafo $D = (V, A)$ è aciclico se e solo se si può dare una funzione di ordinamento totale "$<$" dei suoi vertici tale per cui $\forall (i, j) \in E, i < j$.

Dimostrazione:

Supponiamo che esista un tale ordinamento. Se per assurdo esistesse un ciclo $(v_1, \ldots, v_n)$ avremmo $v_1 < v_2 < \cdots < v_n < v_1 < v_2 < \ldots$, cioè per la transitività della relazione d'ordine avremmo $v_1 < v_2$ e $v_2 < v_1$.

Dimostriamo il "solo se" dimostrando che la relazione d'ordine fornita dal seguente algoritmo è una relazione d'ordine valida.

Nell'algoritmo l'eliminazione di un nodo q dal digrafo $D = (V, A)$ comporta, oltre all'eliminazione del nodo da V, l'eliminazione dei suoi archi incidenti da E e quindi l'aggiornamento dei gradi dei nodi adiacenti a q.

```
function NUMBERING(D = (V, A))
    i := 1
    while ( V ≠ ∅ )
        q := arg min_{k∈V} d⁻(k) // PRENDE IL NODO CON GRADO MINORE
        if ( d⁻(q) > 0 )
            return ERROR // IL DIGRAFO CONTIENE UN CICLO
        remove q from D
        ord[q] := i
        i++
    return SUCCESS // IL DIGRAFO È ACICLICO
```

Teorema 3.8.

Se D è aciclico l'algoritmo `NUMBERING` *termina correttamente.*

Dimostrazione:

Se si elimina un nodo da un digrafo aciclico si ottiene ancora un digrafo aciclico. Pertanto ad ogni passo esiste almeno un nodo con grado entrante 0, e ad ogni passo tale nodo viene eliminato. Quindi in esattamente $|V|$ passi l'algoritmo termina.

Teorema 3.9.
Se D contiene un ciclo l'algoritmo `NUMBERING` *produce un errore.*

Dimostrazione:
Supponiamo per assurdo che D contenga un ciclo e che l'algoritmo non fornisca nessun errore. Siano $(v_1, \ldots, v_k = v_1)$ i nodi del ciclo. Sia v_i il primo tra i nodi del ciclo ad essere eliminato. Al momento della sua estrazione avremo $d^-(v_i) = 0$, il che genera ovviamente un assurdo poiché v_{i-1} (modulo k) è ancora nel digrafo.

Teorema 3.10.
Se l'algoritmo `NUMBERING` *termina senza errori allora fornisce un ordinamento valido, ovvero vale* $\texttt{ord}[i] < \texttt{ord}[j] \quad \forall (i,j) \in A$.

Dimostrazione:
Partiamo con la considerazione che `ord` contiene solo valori interi distinti, e che ogni etichetta indica l'istante temporale in cui un nodo è stato eliminato. Supponiamo per assurdo che per un certo arco $(p,q) \in A$ la relazione non valga; allora deve valere $\texttt{ord}[q] < \texttt{ord}[p]$. Questo conduce direttamente ad un assurdo, poiché significherebbe che il nodo q è stato eliminato prima del nodo p, quando cioè aveva grado uscente almeno 1.

Così come è presentato, l'algoritmo non sembra essere lineare; tuttavia, considerando che all'algoritmo serve sempre un nodo con grado entrante nullo e che per i nostri scopi la rimozione consiste nel solo aggiornamento dei gradi entranti, sono sufficienti alcuni accorgimenti per ottenerne una versione lineare. Sia Q una coda, con le operazioni a tempo costante `POP`, `PUSH`, e `ISEMPTY`.

```
function L_NUMBERING(D = (V, A))
    for ⟨ ∀q ∈ V ⟩
        if ( d⁻(q) = 0 )
```

```
            Q.PUSH(q)
    for ⟨ i := 1...|V| ⟩
        if ( Q.ISEMPTY() )
            return ERROR // Il digrafo contiene un ciclo
        q := Q.POP()
        ord[q] := i
        for ⟨ k ∈ V | (q,k) ∈ A ⟩
            d⁻(k)--
            if ( d⁻(k) = 0 )
                Q.PUSH(k)
    return SUCCESS // Il digrafo è aciclico
```

Ottenuto l'ordinamento totale `ord`, si definisca $\texttt{nod}[i] = q \mid \texttt{ord}[q] = i$, per $1 \leqslant i \leqslant |V|$. Per risolvere il problema di cammino minimo da un nodo s a tutti i nodi da lui raggiungibili si può utilizzare il seguente semplice algoritmo:

```
function DAG_MIN(V, A, s, w)
    // Inizializzazione
    dist[] := ∞
    prev[] := -1;
    dist[s] := 0
    i := ord[s]
    // Ciclo principale
    for ⟨ i := ord[s]...|V| ⟩
        q := nod[i]
        for ⟨ k ∈ V | (q,k) ∈ A ⟩
            if ( dist[k]>dist[q]+w(q,k) )
                dist[k] := dist[q]+w(q,k)
                prev[k] := q
```

Teorema 3.11.

Può esistere un cammino da s a t se e solo se $s < t$.

Dimostrazione:

Se il cammino esiste, allora data la transitività della relazione d'ordine avremmo $s < t$. Viceversa, sia $s > t$; se esistesse un cammino da s a t avremmo l'assurdo $s < t \ \wedge \ s > t$.

Da notare che è possibile che si abbia $s < t$ e che tuttavia non esista nessun cammino da s a t.

Teorema 3.12.

L'algoritmo DAG_MIN è corretto.

Dimostrazione:

Per il nodo sorgente vale $\texttt{dist}[s] = d(s,s) = 0$, e per tutti i nodi per cui vale $p < s$ (cioè che non sono raggiungibili da s) vale $\texttt{dist}[p] = d(s,p) = \infty$.

Supponiamo quindi per induzione che $\texttt{dist}[\hat{q}] = d(s,\hat{q})$ per tutti i nodi $\hat{q} < q$, e dimostriamo che è vero per q.

Quando l'algoritmo valuta il nodo q ha già fissato l'etichetta di tutti i nodi p per cui vale $p < q$.

Se per assurdo esistesse un cammino c minore di $\texttt{dist}[q]$, tale cammino dovrebbe contenere almeno un nodo p non fissato, ovvero per cui vale $q < p$. Ma questo conduce direttamente all'assurdo $p < q$ e $q < p$.

Esiste una versione alternativa dell'algoritmo `NUMBERING` che deriva da una ricerca DFS:

```
function DFS_NUMBERING(D = (V, A))
    i := |V|
    V̄ := V
    while ( V̄ ≠ ∅ )
        select q ∈ V̄ // UN NODO QUALSIASI
        DFS(q)

function DFS(q)
    V̄ := V̄ \ {q}
    for ⟨ k ∈ V̄ | (q, k) ∈ A ⟩
        DFS(k)
    ord[q] = i
    i--
```

Come per il precedente algoritmo con alcuni semplici accorgimenti implementativi si può ottenere un algoritmo lineare. Allo stesso modo è facile modificarlo per fare in modo che si accorga della presenza di cicli senza cambiarne la complessità.

La correttezza di questo algoritmo deriva dal fatto che nel momento in cui etichetta un nodo, ha già dato un etichetta maggiore a tutti i nodi da lui raggiungibili.

Osservazione 3.4.

In caso di digrafi aciclici ogni cammino è elementare, quindi il problema di trovare un cammino minimo elementare su digrafi aciclici è in $\mathcal{P}$.

3.8 Numero di cammini su digrafi aciclici

Dato un digrafo $D = (V, A)$ aciclico e un nodo s, si chiede di contare tutti i possibili cammini da s ad ogni altro nodo. Tale problema è risolvibile in tempo lineare in modo simile al problema di trovare il cammino minimo. Una volta ottenuto l'ordinento topografico sul digrafo si esegue il seguente algoritmo.

```
function COUNT_PATHS(V, A, s)
    cnt[] := 0
    cnt[s] := 1
    for ⟨ i:=ord[s]...|V| ⟩
        q := nod[i]
        for ⟨ k ∈ V | (k,q) ∈ A ⟩
            cnt[q] := cnt[q] + cnt[k]
    cnt[s] := 0
```

Alla fine dell'algoritmo `cnt`$[i]$ conterà il numero di tutti i possibili da s a i.

3.9 Cammini massimi

I problemi di cammino massimo si possono riformulare come problemi di cammino minimo cambiando la funzione peso.

Teorema 3.13.
Dato un digrafo $D = (V, A)$ e una funzione di peso w, un cammino c^* è minimo rispetto a w se e solo se lo stesso cammino è massimo rispetto a $-w$.

Dimostrazione:
Sia c^* un cammino minimo rispetto a w. Allora $w(c^*) \leqslant w(c) \quad \forall c$, quindi $-w(c^*) \geqslant -w(c) \quad \forall c$, ovvero c è massimo rispetto a $-w$. Allo stesso modo si dimostra il viceversa.

Riferimenti bibliografici

[1] T.B. Boffey "Graph Theory in Operations Research", Macmillan Computer Science Series, 1982

[2] A.V. Aho, J.E.Hopcroft, J.D.Ullman "Data Structures and Algorithms", Addison Wesley series in Computer Science and Information Processing, 1983

[3] F. Maffioli "Elementi di programmazione matematica", Masson, 1990

[4] Nora Hartsfield, Gerhard Ringel "Pearls in Graph Theory: A Comprehensive Introduction", $2^n d$ edition, Academic Press, 1994

[5] Nicos Christofides "Graph Theory, An Algorithmic Approach", Computer Science and Applied Mathematics, Academic Press, 1975