otterremo una soluzione per il problema $\mathcal{CM}_{a2a}$. Come vedremo, comunque, i costi computazionali di questo algoritmo per i problemi presentati saranno dell'ordine di $O(n^2)$ e $O(n^3)$.

1.2 L'algoritmo di Dijkstra

Nel 1959 Dijkstra [DK] propose un algoritmo iterativo per risolvere il problema del cammino (di lunghezza) minima. Questo algoritmo è basato sulla seguente proprietà fondamentale del problema considerato:

Principio di ottimalità: dato il cammino minimo $C(x, y)$, consideriamo due nodi $p, q \in C$ e chiamiamo $C(p, q)$ il sottocammino di C che va da p a q. Questo sottocammino è il più breve tra quelli che partono da p ed arrivano in q. Infatti, se per assurdo esistesse un altro cammino $C'(p, q)$ con una lunghezza inferiore a quella di $C(p, q)$, sostituire $C(p, q)$ con $C'(p, q)$ all'interno di C indurrebbe un nuovo cammino C' tra x e y di lunghezza inferiore, che è contro l'ipotesi che C sia il cammino minimo.

Vediamo ora come funziona l'algoritmo. Dopo la sua inizializzazione, ed alla fine di ogni iterazione, l'algoritmo opera una partizione dell'insieme dei nodi del grafo in tre insiemi (che verranno aggiornati durante ciascuna iterazione):

- l'insieme K dei nodi ***assegnati***, ovvero tutti quei nodi di cui conosciamo il percorso minimo a partire dalla sorgente s;
- l'insieme F dei nodi ***frontiera***, ovvero già visitati in quanto adiacenti ad almeno un nodo in K e candidati ad entrarvi;
- l'insieme U dei nodi rimanenti o ***sconosciuti***.

otterremo una soluzione per il problema $\mathcal{CM}_{a2a}$. Come vedremo, comunque, i costi computazionali di questo algoritmo per i problemi presentati saranno dell'ordine di $O(n^2)$ e $O(n^3)$.

1.2 L'algoritmo di Dijkstra

Nel 1959 Dijkstra [DK] propose un algoritmo iterativo per risolvere il problema del cammino (di lunghezza) minima. Questo algoritmo è basato sulla seguente proprietà fondamentale del problema considerato:

Principio di ottimalità: dato il cammino minimo $C(x,y)$, consideriamo due nodi $p,q \in C$ e chiamiamo $C(p,q)$ il sottocammino di C che va da p a q. Questo sottocammino è il più breve tra quelli che partono da p ed arrivano in q. Infatti, se per assurdo esistesse un altro cammino $C'(p,q)$ con una lunghezza inferiore a quella di $C(p,q)$, sostituire $C(p,q)$ con $C'(p,q)$ all'interno di C indurrebbe un nuovo cammino C' tra x e y di lunghezza inferiore, che è contro l'ipotesi che C sia il cammino minimo.

Vediamo ora come funziona l'algoritmo. Dopo la sua inizializzazione, ed alla fine di ogni iterazione, l'algoritmo opera una partizione dell'insieme dei nodi del grafo in tre insiemi (che verranno aggiornati durante ciascuna iterazione):

- l'insieme K dei nodi ***assegnati***, ovvero tutti quei nodi di cui conosciamo il percorso minimo a partire dalla sorgente s;
- l'insieme F dei nodi ***frontiera***, ovvero già visitati in quanto adiacenti ad almeno un nodo in K e candidati ad entrarvi;
- l'insieme U dei nodi rimanenti o ***sconosciuti***.

Inoltre per ogni nodo x in K od in F, l'algoritmo considera un certo cammino $C(s,x)$ che potrà cambiare da iterazione a iterazione, se $x \in F$. La lunghezza di $C(s,x)$ sarà una funzione non crescente al crescere del numero delle iterazioni. Come vedremo, se però $x \in K$, $C(s,x)$ non cambierà più e sarà di lunghezza minima. Per semplicità di notazione, nei simboli K, F, U e $C(s,x)$ ometteremo la dipendenza dal numero di iterazioni effettuate. Schematicamente l'algoritmo, descritto nel dettaglio qui di seguito, trasporta ad ogni iterazione un punto di F in K, assegnandogli così il cammino minimo. Per il problema $\mathcal{CM}_{o2o}$ l'algoritmo prosegue finché il vertice t non viene immesso in K. Proseguendo invece l'algoritmo finché tutti gli $n-1$ nodi sono in K, risolviamo $\mathcal{CM}_{o2a}$.

Inizializziamo l'algoritmo mettendo il nodo sorgente s in K, assegnandogli il cammino banale $C(s,s)$, mentre tutti gli altri nodi saranno in U.

La generica iterazione $i+1, i = 0, 1, \ldots, l \leq n-2$ può essere suddivisa in 3 fasi consecutive:

1. Prolunghiamo il cammino $C(s,v_i)$, dove v_i è il nodo appena assegnato durante l'iterazione i all'insieme K, tramite ciascuno degli archi uscenti da esso. Chiamiamo j il nodo di arrivo del generico arco.
2. Se $j \in U$, spostiamo j in F e gli associamo il cammino $C(s,j) = C(s,v_i),j$. Se $j \in F$, rimpiazziamo il cammino corrente $C(s,j)$ con $C(s,v_i),j$ se la lunghezza del secondo è minore di quella del primo. Se invece $j \in K$, ovviamente non consideriamo questo prolungamento.
3. Determiniamo il vertice $\bar{x}$ che minimizza $W(C(s,x))$ tra gli $x \in F$, spostandolo poi da F in K in modo che $v_{i+1} = \bar{x}$.

Ripetiamo questo procedimento finché il nodo destinazione t non viene visitato,

ed abbiamo la soluzione per $\mathcal{CM}_{o2o}$. Come già osservato, se eseguiamo l'algoritmo fino a quando l'insieme U diventerà vuoto, avremo ottenuto il *minimo sotto-albero ricoprente* radicato in S, cioé la soluzione di $\mathcal{CM}_{o2a}$. Ripetendo l'algoritmo n volte indipendentemente risolveremo $\mathcal{CM}_{a2a}$.

Introduciamo ora il seguente teorema, che prova l'efficacia dell'algoritmo di Dijkstra nel trovare la soluzione del problema del cammino minimo. Nella pagine seguenti verrà mostrato che il suo costo computazionale per risolvere $\mathcal{CM}_{o2a}$,e quindi a maggior ragione per $\mathcal{CM}_{o2o}$ è dell'ordine $O(n^2)$, e per risolvere $\mathcal{CM}_{a2a}$ il costo è dell'ordine di $O(n^3)$

Teorema 1.2.1. *L'algoritmo di Dijkstra risolve il problema del cammino minimo* $\mathcal{CM}_{o2a}$.

Dimostrazione.

L'algoritmo trova la soluzione del problema considerato se, per ogni nodo x in K, il cammino $C(s,x)$ è quello di lunghezza minima. Dimostriamo questo per induzione. Il primo passo è banalmente verificato: l'algoritmo inizia con $K = \{s\}$ e certamente il percorso $C(s,s)$ è di lunghezza minima (cioé zero). Supponiamo ora che valga l'ipotesi induttiva all'iterazione i, ovvero che alla fine di essa per ogni nodo $v_j \in K$, $j = 0, \ldots, i$ il percorso minimo da s a v_j sia $C(s,v_j)$. All'iterazione $i+1$, F contiene, alla fine della fase 2, tutti i nodi adiacenti ad almeno un nodo di K; tra i nodi di F, l'algoritmo sceglie nella fase 3 di trasferire in K

$$v_{i+1} = \arg\min_{x \in F} W(C(s,x)).$$

Vogliamo mostrare che il cammino minimo da s a v_{i+1} è proprio $C = C(s, v_{i+1})$. Dimostriamolo per assurdo: supponiamo che esista un altro cammino $\bar{C} \neq C$ da s a v_{i+1} di lunghezza inferiore. Vogliamo mostrare che esiste un sottocammino

C^* di $\bar{C}$ che termina in un nodo di F. Per come è strutturato, l'algoritmo genera cammini, a partire da s, attraverso prolungamenti di cammini minimi per nodi di K. Se il secondo estremo del primo arco di $\bar{C}$ appartiene ad F, abbiamo trovato il sottocammino cercato. Nel caso contrario, esso apparterrà a K. Consideriamo allora il sottocammino di $\bar{C}$ formato da i primi 2 archi: se questo sottocammino termina in un nodo di F, allora abbiamo concluso. Altrimenti il nodo in cui termina questo sottocammino sarà in K, e considereremo allora i primi 3 archi di $\bar{C}$, e così via. Questo ragionamento, poiché non può essere iterato più di i volte, porta alla conclusione che esiste un sottocammino C^* di $\bar{C}$ che termina in un nodo di F. Poiché la lunghezza di un cammino è inferiore a quella di ciascuna sua estensione, avremo che

$$W(C^*) < W(\bar{C}) < W(C).$$

Questo è contro l'ipotesi secondo la quale C è il percorso di lunghezza minima tra quelli che terminano in nodi di F alla fine dell'iterazione $i+1$. Ciò conclude la dimostrazione.

□

Vediamo ora con un esempio come si comporta il nostro algoritmo. Consideriamo il grafo pesato in figura 1.1. Osserviamo che non sono presenti archi orientati perché per questo esempio ammettiamo che un arco possa essere percorso in entrambi i sensi.

Supponiamo di voler raggiungere il nodo e partendo dalla sorgente a: pertanto inizializziamo l'algoritmo mettendo $a \equiv v_0$ in K ed assegnandogli il cammino banale $C(a,a)$, di peso nullo.

Nella fase 1 della prima iterazione generiamo i cammini uscenti da a che ter-