R.O. teoria

Riccardo Zaupa

January 2024

Indice

1 Modelli di PL

2 Simplesso

3 Dualità

4 Cammino minimo

4.1 Scelta algoritmo (Bellman-Ford)

Scelta dell'algoritmo: anche se tutti i costi sono positivi, posso applicare solo l'algoritmo di Bellman-Ford che è l'unico che dia la possibilità di calcolare i cammini minimi con il massimo numero di archi. Infatti, è possibile dimostrare che, all'iterazione k dell'algoritmo, le etichette corrispondono ai cammini minimi che utilizzano al più k archi. Applicheremo quindi Bellman-Ford fermandoci alla quarta iterazione, dopo l'inizializzazione.

4.2 Applicazione algoritmo (Bellman-Ford)

Applicazione dell'algoritmo: si utilizza una tabella che riporta una riga per ogni iterazione dell'algoritmo. Ogni colonna della tabella è dedicata ad un nodo e riporta, iterazione dopo iterazione, l'evoluzione delle rispettive etichette. L'ultima colonna riporta i nodi aggiornati nel corso dell'iterazione: all'iterazione successiva è sufficiente controllare solo gli archi uscenti da questi nodi.

iter.	n. 1	nodo 2	nodo 3	nodo 4	nodo 5	nodo 6	nodo 7	nodo 8	`Aggior.`
init.	$0_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	1
$h=1$	$0_{(\wedge)}$	$+\infty_{(\wedge)}/1_{(1)}$	$+\infty_{(\wedge)}/5_{(1)}$	$+\infty_{(\wedge)}/2_{(1)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$+\infty_{(\wedge)}$	$2,3,4$
$h=2$	$0_{(\wedge)}$	$1_{(1)}$	$5_{(1)}/3_{(4)}$	$2_{(1)}$	$+\infty_{(\wedge)}/7_{(3)}$	$+\infty_{(\wedge)}/7_{(4)}$	$+\infty_{(\wedge)}/10_{(2)}$	$+\infty_{(\wedge)}/9_{(3)}$	$7,5,8,3,6$
$h=3$	$0_{(\wedge)}$	$1_{(1)}$	$3_{(4)}$	$2_{(1)}$	$7_{(3)}/5_{(3)}$	$7_{(4)}$	$10_{(2)}$	$9_{(3)}/8_{(5)}/7_{(3)}$	$8,5$
$h=4$	$0_{(\wedge)}$	$1_{(1)}$	$3_{(4)}$	$2_{(1)}$	$5_{(3)}$	$7_{(4)}/6_{(5)}$	$10_{(2)}/9_{(8)}$	$7_{(3)}/6_{(5)}$	$7,6,8$

Le etichette di una riga sono ottenute controllando i vincoli duali su tutti gli archi uscenti dai nodi "aggiornati" della riga (iterazione) precedente secondo la regola

if $\pi_j > \pi'_i + c_{ij}$ **then** $\pi_j = \pi'_i + c_{ij}$ **and** $p(j) = i$

dove (i, j) è uno degli archi uscenti da un nodo i aggiornato all'iterazione precedente, π_j è l'etichetta corrente (sulla riga corrente) del nodo j, π'_i è l'etichetta del nodo i all'iterazione (riga) *precedente* e c_{ij} è il costo dell'arco (i, j). Nella tabella indico in rosso (anziché sbarrate perché non riesco a farlo in LaTeX, ndr) le etichette che sono migliorate durante la stessa iterazione [1].

Si fa notare che, grazie al fatto di utilizzare l'etichetta del nodo i all'iterazione precedente negli aggiornamenti, all'iterazione h garantiamo di considerare lunghezze di cammini con al più h archi. Ad esempio, per ottenere le etichette all'iterazione con $h = 4$, partendo dal nodo 8, la cui etichetta valeva 7 all'iterazione precedente, si aggiorna il nodo 7 al valore 9 ($10 > 7 + 2 = 9$) e non al valore $6 + 2 = 8$ (il valore 6 è relativo all'iterazione corrente e non a quella precedente e, infatti, l'etichetta 8 per il nodo 7 è ottenibile con 5 archi e non con 4).

Un cammino minimo con al massimo 4 archi da 1 a 7: seguo la catena dei predecessori a partire dal nodo 7 sulla riga con $h = 4$ e, ad ogni passo, considero la riga precedente (con h diminuito di 1). Ottengo $7 \leftarrow 8 \leftarrow 3 \leftarrow 4 \leftarrow 1$, cioè il cammino di 4 archi $1 \rightarrow 4 \rightarrow 3 \rightarrow 8 \rightarrow 7$ il cui costo è effettivamente 9.

4.3 Complessità (Dijkstra)

Proprietï¿½ 10 *Dato un grafo pesato $G = (N, A)$ e un nodo origine $s \in N$, l'algoritmo di Dijkstra risolve il problema del cammino minimo dall'origine s verso tutti gli altri nodi in tempo $O(|N|^2)$.*

Dimostrazione: L'algoritmo parte con $S = \emptyset$ e trasferisce un nodo da $\bar{S}$ in S ad ogni iterazione. Dopo $|N|$ iterazioni quindi, $\bar{S} = \emptyset$ e l'algoritmo termina. All'ultima iterazione, tutti i nodi sono in S e, pertanto, π_i rappresenta il costo del cammino minimo da s verso i, per tutti i nodi $i \in N$ (e $p(i)$ il relativo predecessore). Inoltre, il passo `0)` comprende $O(N)$ operazioni di inizializzazione effettuabili in tempo costante. Il passo `1)` consiste nella ricerca del minimo in un insieme di al piï¿½ $|N|$ elementi. La ricerca del minimo si puï¿½ effettuare scandendo i nodi in $\bar{S}$ ed effettuando $|\bar{S}|$ confronti, quindi in tempo $O(|N|)$. Per quanto riguarda il passo `2)`, consideriamo la sua complessitï¿½ *ammortizzata*, ossia la complessitï¿½ cumulativa di tutte le iterazioni. Ad ogni iterazione si considerano gli archi uscenti da un solo nodo. Nel corso delle iterazioni si considerano tutti i nodi e, di conseguenza, si verificano i vincoli duali su tutti gli archi (per alcuni, eventualmente, in modo implicito, se la testa dell'arco non ï¿½ un nodo in $\bar{S}$). L'operazione di verifica ed eventuale aggiornamento di un etichetta e di un puntatore prende tempo costante e pertanto, la complessitï¿½ ammortizzata del passo `3)` ï¿½ $O(|A|)$. In totale, l'algoritmo converge alla soluzione ottima in tempo $O(|N| + |N|^2 + |A|) = O(|N|^2)$. ■

Si fa notare che, nella dimostrazione di correttezza, abbiamo utilizzato il Lemma 4, che usa l'ipotesi che *tutti i costi* siano non negativi: se esiste anche solo un costo negativo, non possiamo applicare l'algoritmo di Dijkstra, anche se siamo sicuri che non esistono cicli negativi. **Insomma, si ribadisce che, per la corretta applicazione dell'algoritmo di Dijkstra, ï¿½ necessario che *tutti i costi* siano** ≥ 0.

5 Branch and Bound

	min $[LB;\ SA]$	max $[SA;\ UB]$
min o max	LB sempre crescente	UB sempre decrescente
Chiusura nodi	LB $\geq$ SA più bassa	UB $\leq$ SA più alta
Intervallo ottimo	$[\text{min } LB\ figli;\ \text{min } SA^*\ generale]$	$[\text{max } SA^*\ generale;\ \text{max } UB\ figli]$
Nuovo nodo	LB = SA	SA = UB
Valori ottimi	LB padre $\leq$ SA $\leq$ LB min**	UB max** $\leq$ SA $\leq$ UB padre