Aggiornamento del flusso massimo

Problema

Sia $G = (V, E, s, t, c)$ una rete di flusso. Si supponga di conoscere un flusso massimo in G, e si supponga che la capacità di un singolo arco $(u, v) \in E$ sia aumentata di una unità. Progettare un algoritmo per aggiornare il flusso massimo in tempo $O(n + m)$.

Cammini indipendenti

Problema

Dato un grafo orientato $G = (V, E)$ e due vertici u, v contenuti in V, trovare il numero totale di cammini "edge-independent", ovvero in cui un arco può comparire al massimo in un cammino.

Torri di controllo

Problema

Si consideri un insieme di aerei $A = \{a_1, \ldots, a_n\}$ e un insieme di torri di controllo $T = \{t_1, \ldots, t_m\}$.

In ogni istante, ogni aereo e ogni torre è dotato di coordinate geografiche $(a_i.x, a_i.y)$ o $/t_j.x, t_j.y)$. Ogni torre può gestire al più L aerei, e ovviamente questi devono essere a portata radio r dalla torre.

Scrivere un algoritmo che assegni ogni aereo ad una torre, rispettando i vincoli sulla distanza e sul carico.

Discutere correttezza e complessità.

MC Donald

Gestire un McDonald non è semplice.

- La giornata lavorativa è suddivisa in 3 turni da 4 ore, dalle 11 alle 23.
- Durante il turno $t_i \in T$ (dove T l'insieme di 21 turni), è necessario che siano presenti p_i unità di personale
- Avete a disposizione un insieme D di dipendenti; ogni dipendente $d_j \in D$ dichiara un insieme di turni $n_j \subseteq T$ in cui non può lavorare.
- Ad esempio, d_j non può lavorare nei turni $\{t_1, t_7, t_{21}\}$.
- Per contratto aziendale, ogni lavoratore non può lavorare per più di 5 turni.
- Ogni giorno, un dipendendente non può lavorare per più di due turni (qualsiasi, anche non consecutivi).

Progettare un algoritmo che produca uno scheduling che illustri, per ogni turno, il personale associato e discuterne la complessità.

Spoiler alert!

Aggiornamento del flusso massimo

---

maxFlow(**integer**[ ][ ] c, **integer** n, **integer** s, **integer** p, **integer**[ ][ ] f, **integer** u, **integer** v)

---

integer[ ][ ] $r \leftarrow$ **integer**$[1 \ldots n][1 \ldots n]$ % Rete residua

integer[ ][ ] $g \leftarrow$ **integer**$[1 \ldots n][1 \ldots n]$ % Cammino aumentante

$c[u, v] \leftarrow c[u, v] + 1$

$r \leftarrow c - f$

$g \leftarrow$ cammino-aumentante(r, n, s, p)

$f \leftarrow f + g$

Nota: Operazioni come $f \leftarrow f + g$ si intendono come somma, elemento per elemento, delle matrici e hanno costo $O(n^2)$.

Cammini indipendenti

Si costruisce una funzione di capacità c tale per cui

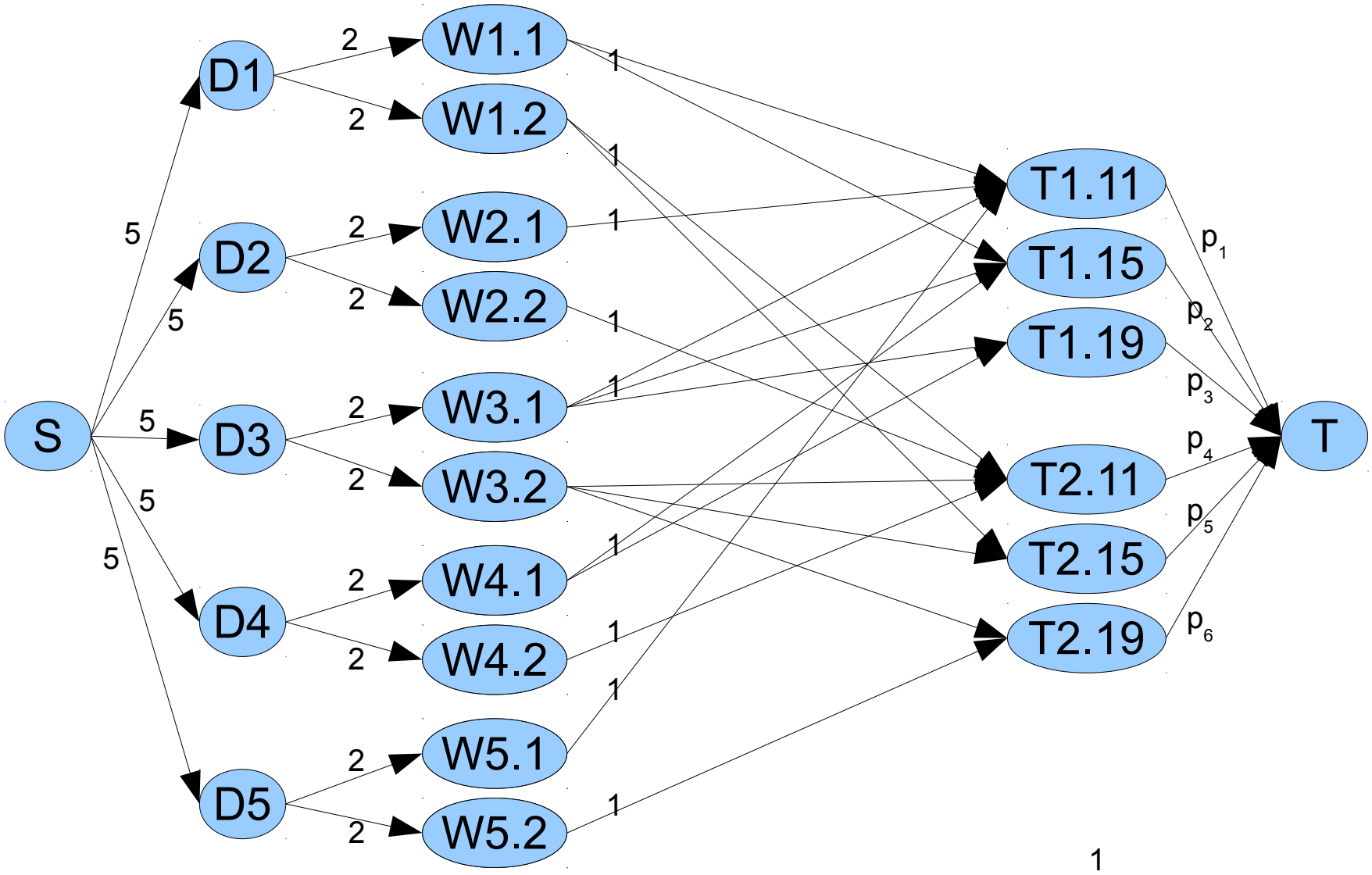
- $c(x, y) = 1$ se $(x, y) \in E$
- $c(x, y) = 0$ se $(x, y) \notin E$.

Si consideri il massimo flusso fra u e v; questo valore corrisponde al numero totale di cammini indipendenti, in quanto essendo la capacità di tutti gli archi pari ad 1, ogni cammino aumentante avrà valore di flusso 1 e tutti i suoi archi saranno distinti dagli altri cammini aumentanti.

Torri di controllo

- È possibile associare ad ogni aereo ed ad ogni torre i vertici di un grafo bipartito.
- Ogni aereo è connesso ad una torre se è entro la sua portata radio con un arco di capacità 1.
- Si aggiunge poi un nodo sorgente s, connesso a tutti gli aerei con archi di capacità 1; e un nodo pozzo p, connesso a tutte le torri con archi di capacità L.
- Il flusso massimo è ovviamente n. Utilizzando il limite derivante da Ford e Fulkerson, il costo dell'algoritmo risultante è quindi $O(n(|V| + |E|))$, dove $|V| = n + m + 2$ e $|E| = O(nm) + n + m$; il costo totale è quindi $O(n^2 m)$.

MCDonald


S
D1
D2
D3
D4
D5
W1.1
W1.2
W2.1
W2.2
W3.1
W3.2
W4.1
W4.2
W5.1
W5.2
T1.11
T1.15
T1.19
T2.11
T2.15
T2.19
T
5
5
5
5
5
2
2
2
2
2
2
2
2
2
2
1
1
1
1
1
1
1
1
1
1
p1
p2
p3
p4
p5
p6
1