Esercizi Capitolo 15 - Ricerca locale

Alberto Montresor

19 Agosto, 2014

Alcuni degli esercizi che seguono sono associati alle rispettive soluzioni. Se il vostro lettore PDF lo consente, è possibile saltare alle rispettive soluzioni tramite collegamenti ipertestuali. Altrimenti, fate riferimento ai titoli degli esercizi. Ovviamente, si consiglia di provare a risolvere gli esercizi personalmente, prima di guardare la soluzione.
Per molti di questi esercizi l'ispirazione è stata presa dal web. In alcuni casi non è possibile risalire alla fonte originale. Gli autori originali possono richiedere la rimozione di un esercizio o l'aggiunta di una nota di riconoscimento scrivendo ad `alberto.montresor@unitn.it`.

1 Problemi

1.1 Riempimento di matrice (Esercizio 15.4 del libro)

Dati $2n$ interi non negativi $c_1, c_2, \ldots, c_n, r_1, r_2, \ldots, r_n$, si vuole determinare una matrice $n \times n$ con elementi interi non negativi tali che la somma degli elementi della colonna i-esima sia uguale a c_i e la somma degli elementi della riga i-esima sia uguale ad r_i. Quali condizioni devono valere su $c_1, \ldots, c_n, r_1, \ldots, r_n$ affinché esista una soluzione? Si formuli il problema come un problema di flusso massimo. Si fornisca poi un algoritmo di complessità $O(n^2)$ senza usare il flusso massimo.

Soluzione: Sezione 2.1

1.2 Somma di matrici (Esercizio 15.5 del libro)

Siano dati una matrice M di dimensione $m \times n$ con elementi non negativi e quattro interi non negativi l_1, l_2, t_1, t_2. Si vogliono determinare due matrici M_1 e M_2 di dimensione $m \times n$ con elementi interi non negativi tali che $M = M_1 + M_2$, la somma degli elementi di ogni linea (cioè riga o colonna) di M_i non superi l_i, e la somma di tutti gli elementi di M_i non superi t_i, per $i = 1, 2$. Si formuli il problema come un problema di flusso massimo.

Soluzione: Sezione 2.2

1.3 Formulazioni flusso massimo (Esercizio 15.6 del libro)

Si dimostri che le seguenti varianti del problema del flusso massimo possono essere ricondotte alla formulazione originale del problema descritta nel presente capitolo:

- Il grafo ha più di una sorgente e più di un pozzo;
- I nodi, oltre agli archi, hanno capacità;
- Il grafo non è orientato.

Soluzione: Sezione 2.3

1.4 Aggiornamento del flusso massimo

Sia $G = (V, E, s, p, c)$ 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 $O(n + m)$.

1.5 Cammini indipendenti

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.

Soluzione: Sezione 2.5

1.6 Piano sanitario

Un grave terremoto ha colpito una regione. Ci sono n persone ferite dislocate nell'area colpita, e k ospedali sono ancora funzionanti. Ogni persona deve essere portata in un ospedale che non disti più di mezz'ora dal luogo di ferimento. Si vuole evitare di distribuire i pazienti in maniera diseguale, così ogni ospedale deve ricevere al più $\lceil n/k \rceil$ pazienti. Progettare un algoritmo che distribuisca i pazienti fra gli ospedali e discuterne la complessità.

Soluzione: Sezione 2.6

1.7 Torri di controllo

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.

Soluzione: Sezione 2.7

1.8 McDonald

Gestire un McDonald non è semplice. La giornata lavorativa è suddivisa in 3 turni da quattro ore, dalle 11 alle 23. Il management ha stabilito che per ognuno dei turni settimanali contenuti nell'insieme T ($7 \cdot 3 = 21$ turni totali), una certa quantità di personale è necessaria. Ad esempio, sabato dalle 19 alle 23 c'è la richiesta massima. Dato il turno $t_i \in T$, la richiesta di personale è p_i. 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 turno t_i deve essere coperto esattamente da p_i personale. 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 discutere la complessità dell'algoritmo risultante.

Soluzione: Sezione 2.8

1.9 Reti telematiche

Una rete telematica è formata da un'insieme di nodi di smistamento e un insieme di connessioni tra i nodi. Ogni connessione ha una capacità limitata (per esempio una connessione modem può essere limitata da 56Kb/sec). Il nostro obiettivo è inviare n filmati *distinti* $f_1 \ldots f_n$ da un computer "server" s a n computer "client" $c_1, \ldots, c_n$; tutti i filmati richiedono t bit/s per essere trasmessi. Si proponga un algoritmo per stabilire se la rete a nostra disposizione è in grado di trasmettere i filmati alle destinazioni.

Soluzione: Sezione 2.9

1.10 Evacuazione

Siete stati incaricati di redigere i piani di evacuazione di una regione minacciata da un vulcano. In questa regione, che può essere reppresentata da un insieme di nodi V e un insieme di archi (strade) E che connettono i nodi. Esiste un sottoinsieme $S \subset V$ di nodi detti villaggi; ed un insieme $P = V - S$ di luoghi "sicuri", dove la popolazione dei villaggi si può rifugiare in caso di eruzione.
Il vostro compito è scrivere un algoritmo per determinare se esiste un insieme di "cammini di evacuazione", tali per cui

- ogni cammino inizia da un nodo in S e termina in un nodo in P;
- ogni nodo v in S ha esattamente un cammino che parte da v;
- i cammini non condividono nessuno degli archi, ovvero tutti gli archi sono attraversati al massimo una volta da un cammino.

Discutere la complessità computazionale del vostro algoritmo.

Soluzione: Sezione 2.10

2 Soluzioni

2.1 Riempimento di matrice

Condizione necessaria e sufficiente affinché il problema abbia soluzione è la seguente:

$$\sum_{i=1}^{n} r_i = \sum_{j=1}^{n} c_j$$

ovvero, la sommatoria dei valori sulle righe deve essere uguale alla sommatoria dei valori sulle colonne.
Il problema può essere formulato come un problema di flusso massimo come in Figura 1; per semplicità, nella figura consideriamo una matrice 2×2.
Innanzitutto, definiamo un nodo per ognuna delle celle della matrice M (nodi grigi in figura). Poi definiamo un nodo per ognuna delle righe e delle colonne della matrice M. Infine, aggiungiamo una sorgente ed un pozzo. Il pozzo è collegato ad ogni riga i, con una capacità pari a r_i; ogni colonna j è collegata al pozzo, con una capacità pari a c_j. Ogni riga i ha un arco uscente verso le celle che la compongono, con una capacità pari a r_i; le celle che compongono una colonna j hanno archi uscenti verso la colonna j, con capacità c_j.
L'idea è la seguente: il valore assegnato ad una riga può essere distribuito fra le celle che la compongono; il valore ricevuto dalle celle può essere distribuito alle rispettive colonne.
Trovare il flusso massimo in questa rete consiste nel saturare i tagli che separano la sorgente dalle righe, e le colonne dal pozzo. Il flusso uscente dai nodi cella (nodi colorati in grigio) rappresenta il valore da assegnare alle celle corrispondenti
La complessità di questo algoritmo, secondo Ford-Fulkerson, è data da $O(C(|V|+|E|))$, dove $C = \sum_{i=1}^{n} c_i$ è la sommatoria delle colonne (o delle righe), mentre $|V| = n^2+2n+2$, mentre $|E|$ è pari a $2n^2+2n$; ovvero, il limite è dato da $O(Cn^2)$. Secondo Edmonds e Karp, il costo limitato superiormente da $O(|V| \cdot |E|^2)$, ovvero $O(n^6)$.

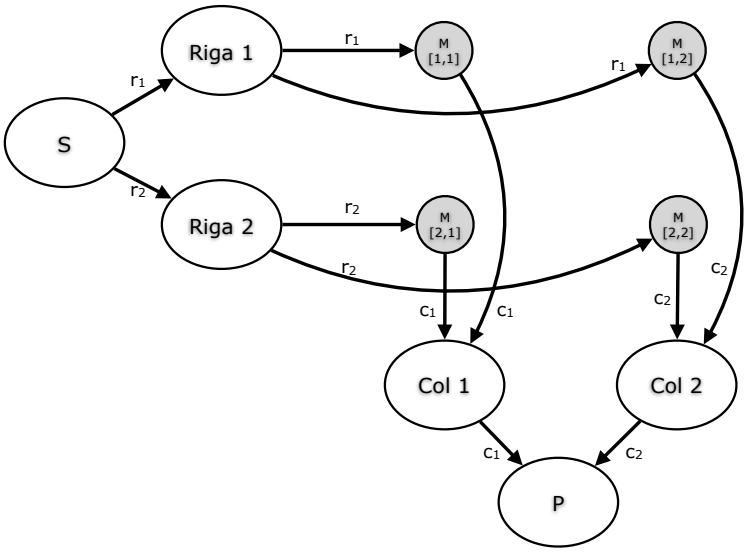

Figura 1: Rete di flusso associata ad una matrice 2×2 per il problema Riempimento matrice

Esiste un metodo più efficiente per calcolare M. Si consideri una qualunque matrice M' tale che:

$$\begin{aligned} \sum_{j=1}^{n} M'[i,j] &\leq r_i \\ \sum_{i=1}^{n} M'[i,j] &\leq c_j \end{aligned}$$

ovvero, una matrice in cui ogni riga i e ogni colonna j non eccedano i corrispondenti valori r_i e c_j. Possiamo calcolare una nuova coppia di vettori r' e c' nel modo seguente:

$$\begin{aligned} r'_i &= r_i - \sum_{j=1}^{n} M'[i,j] \\ c'_j &= c_j - \sum_{i=1}^{n} M'[i,j] \end{aligned}$$

Questi nuovi vettori costituiscono un nuovo problema di riempimento matrice, che risolto produce una matrice M". La matrice $M = M' + M$" è una soluzione del problema originale.
Forti di questa proprietà, cerchiamo di risolvere il problema una riga o una colonna per volta. Si consideri la cella in basso a destra $M[n,n]$; abbiamo due casi:

- $r_n < c_n$: assegniamo $M[n,n] = r_n$; poiché abbiamo saturato il valore della riga con un'unica casella, possiamo assegnare il valore $M[n,i] = 0$ per tutti gli i, $1 \leq i < n$. r_n diventa 0, mentre c_n diventa uguale a $c_n - r_n$. Abbiamo così cancellato l'ultima riga, e possiamo volgere la nostra attenzione alla casella $M[n-1,n]$.
- $r_n \geq c_n$: assegniamo $M[n,n] = c_n$; poiché abbiamo saturato il valore della colonna con un'unica casella, possiamo assegnare il valore $M[i,n] = 0$ per tutti gli i, $1 \leq i < n$. c_n diventa 0, mentre r_n diventa uguale a $r_n - c_n$. Abbiamo così cancellato l'ultima colonna, e possiamo volgere la nostra attenzione alla casella $M[n,n-1]$.

È facile vedere che l'algoritmo termina quando si raggiunge la casella $M[1,1]$, la si riempe con il valore residuo $c_1 = r_1$ e quindi si esce dalla matrice.
L'algoritmo illustrato di seguito inizializza tutte le celle a 0 e riempe quindi con valori positivi le celle considerate lungo il percorso verso $M[1,1]$. Il costo computazionale è $O(n^2)$, dato dall'inizializzazione.

```
integer[][] riempi(integer[] r, integer[] c, integer n)
    integer[][]M ← new integer[1...n, 1...n]
    for i ← 1 to n do
        for j ← 1 to n do
            M[i, j] ← 0
    i ← j ← n
    while i > 0 and j > 0 do
        if r[i] < c[j] then
            M[i, j] ← r[i]
            c[j] ← c[j] − r[i]
            i ← i − 1
        else
            M[i, j] ← c[j]
            r[i] ← r[i] − c[j]
            j ← j − 1
        return M
```

2.2 Somma di matrici

Il problema può essere formulato come un problema di flusso massimo come in Figura 2; per semplicità, nella figura consideriamo una matrice 2×2. Innanzitutto, definiamo un nodo per ognuna delle celle delle matrici M_1, M_2 e M, rappresentati dai nodi grigi in figura. Poi definiamo un nodo per ognuna delle righe e delle colonne delle matrici M_1 e M_2; poi un nodo per ognuna delle matrici M_1 ed M_2; infine, una sorgente e un pozzo.

Partiamo dai nodi "riga" e "colonna": in ognuno di essi entra un arco con capacità l_1 o l_2 (a seconda appartengano a M_1 o M_2), che possono liberamente ridistribuire ai nodi delle matrici M_1 e M_2 corrispondenti (ad esempio, il nodo "Riga 1-1" può distribuire il proprio valore l_1 ai nodi della riga 1, che sono $M_1[1,1]$ e $M_1[1,2]$). Questi archi hanno capacità l_1 o l_2.
Gli archi entranti nelle righe e nelle colonne provengono dai nodi matrice corrispondenti. La sorgente è collegata alla matrice M_i con archi con capacità $2t_i$; questo perchè vogliamo che il valore t_i sia distribuito due volte, atttraverso le righe e attraverso le colonne.
I nodi delle matrici M_1 e M_2 hanno archi uscenti verso i corrispondenti nodi della matrice M; questa parte della rete rappresenta la somma delle matrici. La loro capacità può essere infinita; il flusso lungo questi archi è uguale al doppio del valore da inserire nelle corrispondenti caselle delle due matrici. I nodi della matrice M hanno archi uscenti verso il pozzo, con capacità doppia rispetto al valore corrispondente della matrice; ancora una volta, questo rappresenta il fatto che i valori sono raddoppiati per via delle righe e delle colonne.

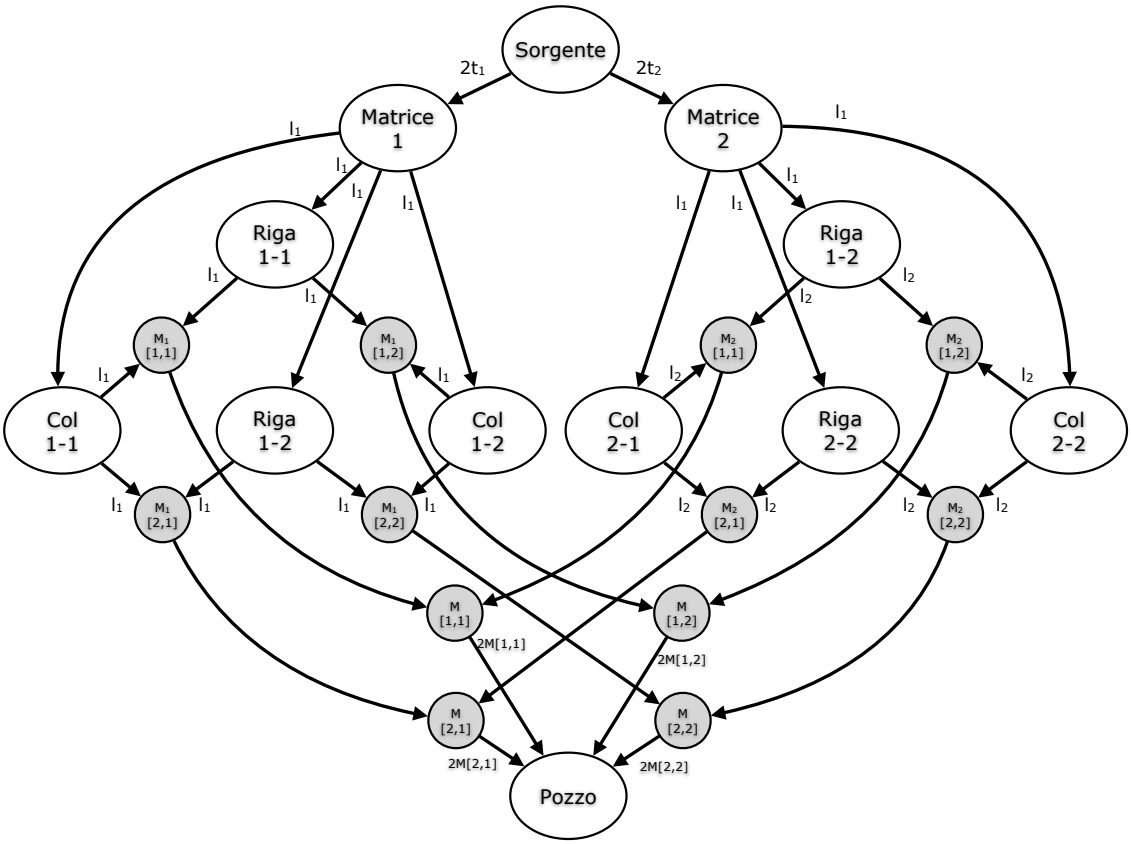

Figura 2: Rete di flusso associata ad una matrice 2×2 per il problema Somma di matrici

2.3 Formulazioni flusso massimo

- È possibile aggiungere un nodo super-sorgente con archi uscenti verso ognuno delle sorgenti ed un nodo super-pozzo con archi entranti da ognuno dei pozzi, ed assegnare capacità infinita ad ognuno di essi.
- È sufficiente creare un nuovo grafo $G' = (V', E')$ "sdoppiando" ogni nodo in una coppia di nodi e collegandoli fra loro con un arco con la capacità del nodo sdoppiato; più precisamente, se $c(u)$ è la

capacità del nodo u, si ha:

$$\begin{aligned} V' &= \{v_{in} : v \in V\} \cup \{v_{out} : v \in V\} \\ E' &= \{(u_{out}, v_{in}) : (u, v) \in E\} \cup \{(v_{in}, v_{out}) : v \in V\} \\ c'(u_{out}, v_{in}) &= c(u, v) \\ c'(v_{in}, v_{out}) &= c(v) \end{aligned}$$

- È sufficiente raddoppiare gli archi, assegnando la stessa capacità ad entrambe le direzioni.

2.4 Aggiornamento del flusso massimo

Calcoliamo la rete residua G_f di f; in questa rete residua, cerchiamo un cammino da s a u e un cammino da v a t; se tali cammini esistono, visto che gli archi con capacità nulla sono stati eliminati, tutti gli archi hanno almeno un'unità. Quindi è possibile aumentare il flusso lungo entrambi i cammini di un'unità.
Il calcolo del residuo può essere realizzato in tempo $O(n + m)$; le visite hanno lo stesso costo.

2.5 Cammini indipendenti

Si costruisce una funzione di capacità c tale per cui $c(x, y) = 1$ se $(x, y) \in E$, mentre $c(x, y) = 0$ 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.

2.6 Piano sanitario

È possibile trasformare questo problema in un una rete di flusso. Si aggiunge un nodo $p_1 \dots p_n$ per ognuna delle persone; un nodo $o_1 \dots o_k$ per ognuno degli ospedali; un nodo sorgente e un nodo pozzo. Ogni persona è unita alla sorgente con un arco di peso 1, ad indicare che ogni persona dovrebbe essere collocata al massimo in un ospedale. Ogni persona è collegata agli ospedali che può raggiungere in meno di mezz'ora con un arco di peso 1. Ogni ospedale è collegato al pozzo con un arco di peso $\lceil n/k \rceil$, ad indicare che ogni ospedale non dovrebbe accogliere un numero maggiore di pazienti.
Il flusso massimo non può superare n; se ha valore n, ogni paziente viene assegnato ad un ospedale. Il costo per l'esecuzione dell'algoritmo di Ford-Fulkerson è $O(|f^*|(|V| + |E|))$; sapendo che $|f^*| = O(n)$, $V = n + k + 2$, $E = O(kn + n + k)$, si ottiene un costo pari a $O(n(n + k + 2 + kn + k + n)) = O(n^2 k)$.

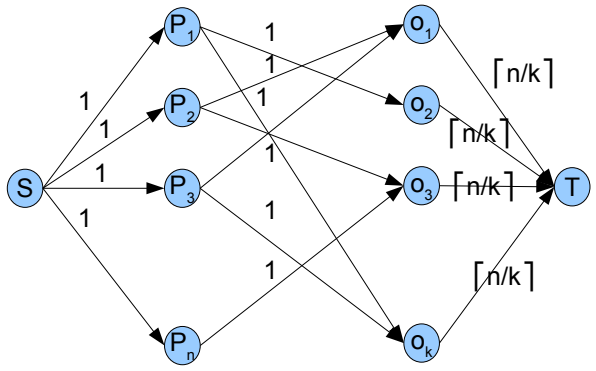

Figura 3: La rete di flusso definita nell'esercizio

2.7 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 $O(n^2m)$.

2.8 McDonald

Il problema può essere ridotto ad un problema di flusso massimo. A questo scopo, è necessario costruire un grafo a partire dai dati di input.
La Figura 4 contiene tale grafo; per ragioni di spazio, sono rappresentati solo due giorni della settimana.
Il grafo è così composto: per ogni dipendente d_j, aggiungiamo un nodo D_j. Per ogni turno t_i, aggiungiamo un nodo $T_{d.h}$, dove d è il giorno della settimana e h è l'ora di inizio turno. Per ogni coppia (dipendente D_j, giorno d) aggiungiamo un nodo $W_{j.d}$. Infine, aggiungiamo la sorgente S e il pozzo T.
Poiché ogni dipendente può lavorare al massimo 5 turni, mettiamo il peso 5 su tutti gli archi (S, D_j). Poiché ogni dipendente può lavorare al massimo 2 turni al giorno, mettiamo il peso 2 su tutti gli archi $(D_j, W_{j.d})$. Le preferenze dei dipendenti sono rappresentate dagli archi $(W_{j.d}, T_{d.h})$, tutti con peso 1. Infine, si collegano i turni al nodo pozzo, con peso pare a p_i.
Esiste un'assegnazione dei dipendenti sui turni se e solo se il flusso è esattamente pari alla somma dei pesi $P = \sum p_i$. Il costo computazionale dell'algoritmo è $O(P(|V| + |E|))$. Poiché ogni lavoratore può lavorare al massimo 5 turni, allora $P \leq 5|D|$. $|V|$ è pari a $8|D| + 25$ ed $|E|$ è $O(29|D| + 21)$, quindi il costo totale è $O(D^2)$.

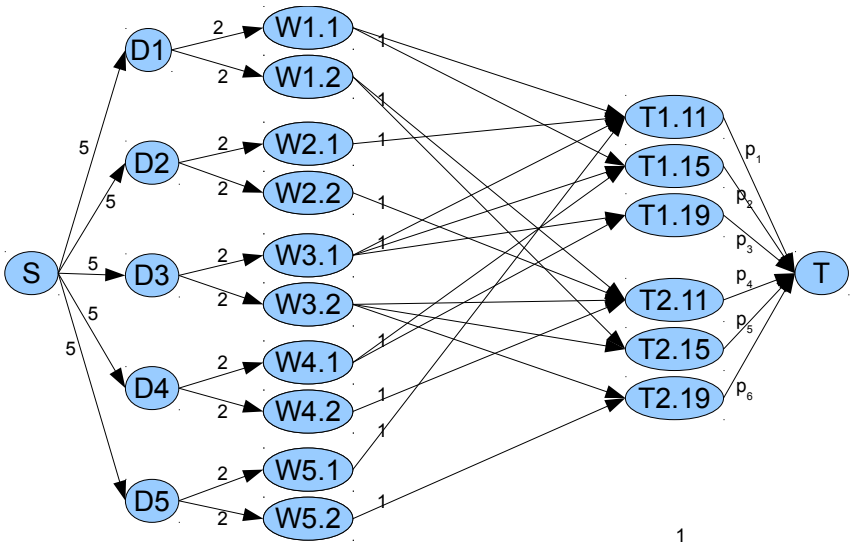

Figura 4: McDonald

2.9 Reti telematiche

Si utilizzano gli algoritmi di flusso massimo. La rete di connessione viene rappresentata da un grafo, con gli archi pesati dalla capacità limitata. Il nodo server costituisce la sorgente; si aggiunge un superpozzo, e un arco con peso k da ognuno dei nodi client al superpozzo. Si esegue un algoritmo di calcolo del flusso massimo, e se il flusso massimo è uguale a tn, la rete è in grado di trasmettere tutti i filmati.

2.10 Evacuazione

Il problema può essere visto come un problema di flusso. Si crea una supersorgente SS, connessa a tutti i nodi nell'insieme S con archi con peso 1. Si assegna peso 1 a tutti i nodi del grafo originario. Si crea un superpozzo SP connesso a tutti i nodi nell'insieme P con archi con peso $|P|$ (perché non abbiamo messo alcuna restrizione al numero di cammini che terminano in un nodo in P). Esiste un insieme di cammini di evacuazione se e solo se il flusso massimo ottenuto in questa rete è pari a $|P|$.
La complessità è pari a $O(m|f*|)$, dove $|f*| = O(|P|) = O(n)$. Il costo totale è quindi $O(mn)$.

3 Problemi aperti

3.1 Scheduling (Esercizio 15.2 del libro)

Dati n programmi con tempi di esecuzione $t_1, \ldots, t_n$, si vuole eseguirli su un processore in modo da minimizzarne il tempo medio di completamento $(1/n)\sum_i c_i$, dove il tempo di completamento c_i del programma i è la somma del suo tempo di esecuzione e dei tempi di esecuzione di tutti i programmi che lo precedono nell'ordinamento (schedule) finale. Si progetti un algoritmo di ricerca locale per risolvere il problema.

3.2 Caso pessimo algoritmo 3 indiani (Esercizio 15.7 del libro)

Si costruisca un grafo G, per una infinità di valori di n, per il quale l'algoritmo dei 3 indiani ha complessità $\Theta(n^3)$.