Reti di Markov 11/05/2023

Possiamo classificare le reti in tre tipologie:

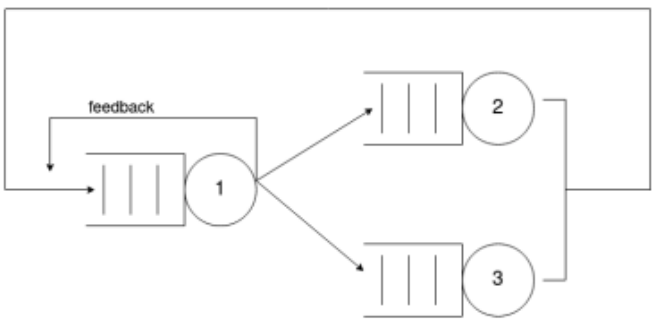
- **Aperte**, quelle maggiormente studiate, in cui c'è un tasso λ, e prima o poi ogni job lascia il sistema.
- **Chiuse**, in cui la popolazione circola per sempre nella rete. Da non confondere con il sistema feedback, perchè anche se in tale sistema un job può rientrare in coda, non rimarrà per sempre nel sistema.
- **Miste**, dove avviene una classificazione degli utenti in classi, e quindi per alcune classi il sistema si comporta come fosse aperto, per altre classi come se fosse chiuso.

Noi siamo particolarmente interessati al modello **Markoviano**, il quale si muove a stati esponenziali. Quando abbiamo visto Erlang, esso proveniva da un processo Markoviano. La rete di code è una visione ad alto livello della rete di Markov. Se definisco il comportamento della rete di code, sto definendo Markov.

Posso usare un processo di Markov se vengono rispettate due condizioni:

- L'insieme di stati deve essere **mutuamente esclusivo**, ovvero per ogni istante di tempo posso trovarmi solo in un unico stato, e **collettivamente esaustivo**, ovvero non deve esistere uno stato non modellabile dal sistema di stati scelto.
- Deve valere la **memoryless**, ovvero, il passaggio dallo stato s_i allo stato futuro s_{i+1} è influenzato unicamente dallo stato s_i, e non da come sono arrivato allo stato s_i.

Esempio con applicazione:



In figura è rappresentata una rete chiusa con M = 3 centri. Poichè la rete è chiusa, nessun job può uscire da questo centro, può solo spostarsi al suo interno. Assumiamo che al suo interno vi siano N = 2 job. I serventi sono a coda singola, FIFO, disciplina astratta. Introduciamo la **matrice di routing/probabilità**, la quale esprime la possibilità di passare da un sotto-centro ad un altro. Questa matrice è MxM.

$$P = \begin{pmatrix} 0,2 & 0,4 & 0,4 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Ad esempio, la componente di riga 3 e colonna 1, ovvero $p_{3,1}$, ci dice che tutti i job uscendi dal centro 3 andranno nel centro 1.

Possiamo notare come, prendendo ogni singola riga, la somma delle componenti sia 1, trattandosi di probabilità. NOTA: per ogni istante di tempo, consideriamo *un solo* job che cambia stato, non di più! Siamo interessati a capire come possano distribuirsi questi due job nel centro. In generale, con tempi di servizio esponenziali, possiamo scrivere: $\bar{s} = (n_1, ..., n_i, ..., n_M)$ ovvero, nel centro i ci sono n job, tra quelli in coda e quelli in servizio. Introduciamo il concetto di **Spazio degli stati**: $E = \{s | n_i \geq 0, \sum_{i=1}^{M} n_i = N\}$ con N popolazione totale, la cardinalità di E è: $|E| = \begin{pmatrix} N+M-1 \\ M-1 \end{pmatrix}$

Vogliamo trovare i possibili stati di questo sistema, il metodo preferibile è **l'algoritmo lessico grafico inverso**. L'idea è di partire da un centro pieno (nel nostro caso il centro 1), togliere un job a questo centro e distribuirlo agli altri centri, poi toglierne un altro e così via. Ovvero, parto da $(N, 0, 0)$, e passo a $(N-1, 1, 0)$ e $(N-1, 0, 1)$.

Poi continuo: $(N-2, 2, 0)$ e $(N-2, 0, 2)$ etc.. vediamolo applicato. $s_1 = (2, 0, 0)$

$s_2 = (1, 1, 0)$ $s_3 = (1, 0, 1)$ $s_4 = (0, 2, 0)$ $s_5 = (0, 1, 1)$ $s_6 = (0, 0, 2)$

Osservazione: quando arriviamo in $s4$ è come se l'algoritmo ripartisse, poichè nel centro 2 ho tutti gli N job, e quindi da lui inizio nuovamente a distribuire i job *guardando avanti*, infatti non riassegno alcun job al centro 1, ma solo al centro 3. Ciò che sto facendo è descrivere il processo di Markov che modella questa rete di code. Adesso possiamo associare ad ogni stato un *tempo di vita*, ovvero il tempo passato in un certo stato, e lo indichiamo con t_{vi}. Vediamo un esempio:

Partiamo da $s2 = (1, 1, 0)$, da lui possiamo andare verso $s4 = (0, 2, 0)$ ed $s_5 = (0, 1, 1)$. Il tempo di vita in s_2 è l'inverso del tasso di uscita in quello stato (è come dire che ci sto un tempo inverso a quanto velocemente me ne vado). Si può dimostrare, ed è una condizione *sufficiente ma non necessaria*, che se i tassi di uscita sono esponenziali, allora anche i tempi di vita nello stato lo sono. In s_2 il centro 1 ed il centro 2 presentano dei job,esco da tale stato se termina uno dei due job. Se il tasso di uscita dai due centri è rispettivamente μ_1 e μ_2, allora il tempo di vita nello stato 2 è : $E(t_{v2}) = \frac{1}{\mu_1 + \mu_2}$. Se $s_1 = (2, 0, 0)$ allora $E(t_{v1}) = \frac{1}{\mu_1}$ (ne esce solo uno alla volta!).

Transizione tra gli stati con probabilità

Vogliamo adesso analizzare le transizioni tra gli stati. Concentriamoci su le frecce *entranti* nello stato $(1,1,0)$. Prendiamo la prima componente dello stato, ovvero il suo primo '1'. Quali sono gli stati, che per effetto di una transazione, ci permettono di entrare in $(1,1,0)$?

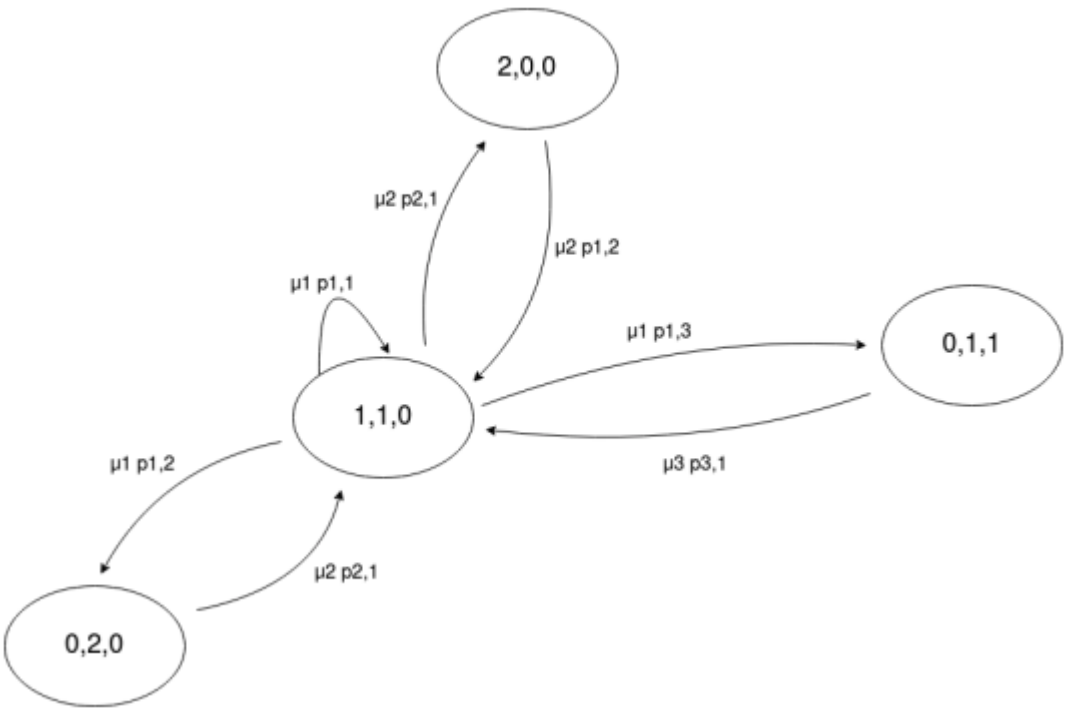
L'idea è questa: per avere '1' come prima componente, questo '1' lo devo prendere dalle seconda e terza componente, quindi devo partire da stati che hanno seconda e terza componente incrementata di 1 (un incremento alla volta, non incremento entrambe!), perchè poi verrà decrementata in favore della prima. Definiamo $p_{i,j}$ come la probabilità di transitare dallo stato i allo stato j.

- $(0,2,0)$, con un job del centro 2 che va nel centro 1, ovvero con tasso $\mu_2 * p_{2,1}$
- $(1,1,0)$ perchè rientra in sè stesso, con tasso $\mu_1 * p_{1,1}$.
- $(0,1,1)$ se il job nel centro 3 va nel job di centro 1, ovvero con tasso $\mu_3 * p_{3,1}$

Adesso ci concentriamo sul secondo '1' nello stato. Come ci arrivo? Per avere '1' sulla seconda componente, questo uno deve essere fornito da prima o terza componente, quindi dagli stati con tali componenti incrementati di uno.

- $(2,0,0)$ con tasso $\mu_1 * p_{1,2}$.
- $(0,1,1)$ già avuto prima, non lo conto due volte!

Questo ragionamento lo applico solo ai centri *non vuoti*, quindi non serve ragionare sulla terza componente. In figura osserviamo anche le *uscite*.



A tutto ciò possiamo applicare il **Bilanciamento del flusso**, ovvero per il singolo stato possiamo scrivere, tramite equazione, che il flusso in entrata equivale al flusso in uscita. Per il singolo stato, stiamo applicando il bilanciamento globale dello stato. Se esiste una soluzione stazionaria, allora queste equazioni possono essere scritte come:

$\pi(\bar{s})$(flusso *OUT* dallo stato) = flusso *IN* dello stato = $\sum_{\forall \bar{s} \in E} \pi(\bar{s}) *$(flusso *IN* $\bar{s}$)

Dove $\pi(\bar{s})$ è la probabilità stazionaria di un certo stato $\bar{s}$.

Applichiamola all'esempio: $P_{stato(1,1,0)}(\mu_1 p_{1,1} + \mu_2 p_{2,1} + \mu_1 p_{1,2} + \mu_1 p_{1,2}) = P_{1,1,0}\mu_1 + P_{2,0,0}\mu_2 + ...$

Riprendiamo la formula di prima: $\pi(\bar{s})$(flusso *OUT* dallo stato) = flusso *IN* dello stato = $\sum_{\forall \bar{s} \in E} \pi(\bar{s}) *$(flusso *IN* $\bar{s}$)

se portiamo la componente di sinistra a destra, e sommiamo $\pi(\bar{s})$ a destra e sinistra, abbiamo: $\pi(\bar{s}) = \pi(\bar{s})(1 - Flusso_{out}) + \sum \pi(\bar{s}) Flusso_{in}$

che mi permette di scrivere il tutto in forma matriciale:

$\bar{\pi} = \bar{\pi} Q$

dove abbiamo $\bar{\pi}$ = tutti gli stati e Q = matrice diagonale sparsa, di dimensione NxM(in questo caso 6, ma in generale molto più grande). Sulla diagonale principale di Q abbiamo componente "$1 - \sum q_{i,j}$" per ogni riga, ovvero: $Q = \begin{pmatrix} ... & ... & q_{1,j} & ... & ... \\ q_{i,1} & q_{i,2} & 1 - \sum q_{i,j} & ... & q_{i,|E|} \\ ... & ... & q_{|E|,j} & ... & ... \end{pmatrix}$

Per ogni riga, la somma deve essere 1. $q_{i,j}$ è la frequenza di transazione da s_i a s_j, ovvero tra gli **stati**, non i centri!

Procediamo con $\bar{\pi} Q - \bar{\pi} = 0$, passo a $\bar{\pi}(Q - I) = 0$ $\bar{\pi} S = 0$, ovvero S è detto *generatore del processo*, in cui abbiamo sottratto 1 agli elementi sulla diagonale principale. (nb: 0 è matrice nulla, non un semplice numero.)

Manca la condizione di *normalizzazione*, perchè stiamo trattando probabilità, inoltre senza questa avrei infinite soluzioni.

$\sum_{\forall s_j \in E} \pi(s_j) = 1$

Ciò che si fa è prendere una qualsiasi colonna di S, metterci tutti '1', ovvero $[1, 1, ..., 1]^T$, generando così S'.

Allora, se ad esempio faccio tale sostituzione alla prima colonna, *impongo* $\bar{\pi} S' = [1, 0, ..., 0]^T$ ovvero "*probabilità di tutti gli stati*" x " *prima colonna di tutti '1'* " = *normalizzazione*

Concludo trovando la soluzione $\bar{\pi} = [1, 0, ..., 0]S^{-1}$, poichè in questo primo caso la soluzione sta nella prima riga, ovvero mi interesserà solo la prima riga della matrice inversa S^{-1} ,ma ovviamente possiamo generalizzare tutto per una riga $i - esima$.

Riprendendo la figura di sopra, se abbiamo $t \to \infty$, raggiungiamo la *stazionarietà*. Ciò che accade è che la probabilità di stato assume una certa forma $\pi(staz)$. Essa esiste sotto le condizioni:

- *irriducibilità*: ogni stato è raggiungibile a partire da tutti gli altri in un numero finito di passi.
- *aperiodicità*: posso tornare ad uno stato s_i in ogni istante t_i, non c'è periodicità del ritorno.
- *ricorrenza positiva*: il tempo di ritorno in uno stato è finito.

I primi due aspetti implicano l'esistenza di una soluzione stazionaria, ovvero $\pi(s_i) = \lim_{t\to\infty} \pi(s_i, t)$. Il terzo aspetto implica la sua unicità e non banalità (ovvero diversa da 0).

Le equazioni di bilanciamento valgono se il sistema è stazionario.

Processo Ergodico

Una catena di Markov è *ergodica* se tutti i suoi stati sono ergodici (passano per tutti gli stati possibili). Inoltre i suoi stati sono aperiodici e ricorrenti non nulli. Una catena di Markov finita (se e solo se è "a stati finiti"), irriducibile (se e solo se lo è la rete sottostante) e con stati aperiodici comporta una *soluzione stazionaria*.

Indici di Prestazione Locali

Supponiamo che il sistema sia stabile e pronto per essere valutato. Disponiamo delle probabilità $\pi(n_1, ..., n_M) = \pi(\bar{s})$

Definiamo:

- **Probabilità marginale della lunghezza di coda** $p_i(n) = \sum_{\bar{s}:n_i=n} \pi(\bar{s})$ Ad esempio, fissati i=2 e n=3, stiamo sommando le probabilità degli stati aventi n=3 job nel centro i=2.
- **Popolazione media nel centro i**: $E(n_i) = \sum_{n_i=0}^{n} n_i p_i(n_i)$ dove moltiplichiamo *valore* e *peso*.
- **Occupazione media del centro i:** $E(C_i) = \sum_{j=1}^{m_i-1} j p_i(j) + \sum_{j=m_i}^{N} m_i p_i(j)$ definita come nel caso ERLANG-C
- **Utilizzazione in funzione dei serventi:**

- Se il centro è a **servente singolo**, $U_i = 1 - p_i(0)$
- Se il centro è **multi-server** (ovvero come se avesse m_i serventi paralleli), $U_i = \frac{E(c_i)}{m_i} = \sum_{j=1}^{m_i-1} \frac{jp_i(j)}{m_i} + \sum_{j=m_i}^{N} \frac{m_i p_i(j)}{m_i}$ Dove abbiamo chi viene subito servito e chi aspetta in coda. In realtà è più limitato rispetto a Erlang-C, perchè la rete è chiusa. E' come se fosse la probabilità che, guardando il sistema per un certo intervallo di tempo, trovassi m_i server occupati nel centro i.
- Se il centro è **infinite server**, è come se non avessi coda, perchè accolgo tutti, non attendo mai, trovo sempre un server libero. Quindi $m_i \to N$ e $U_i = \frac{E(c_i)}{N} = \sum_{j=1}^{N} \frac{jp_i(j)}{N}$

Throughput Locali

Nel sistema aperto si trattava di un parametro di input, ovvero il traffico entrante (se $\lambda < \mu$) Qui abbiamo una popolazione N che circola indefinitivamente nella rete.

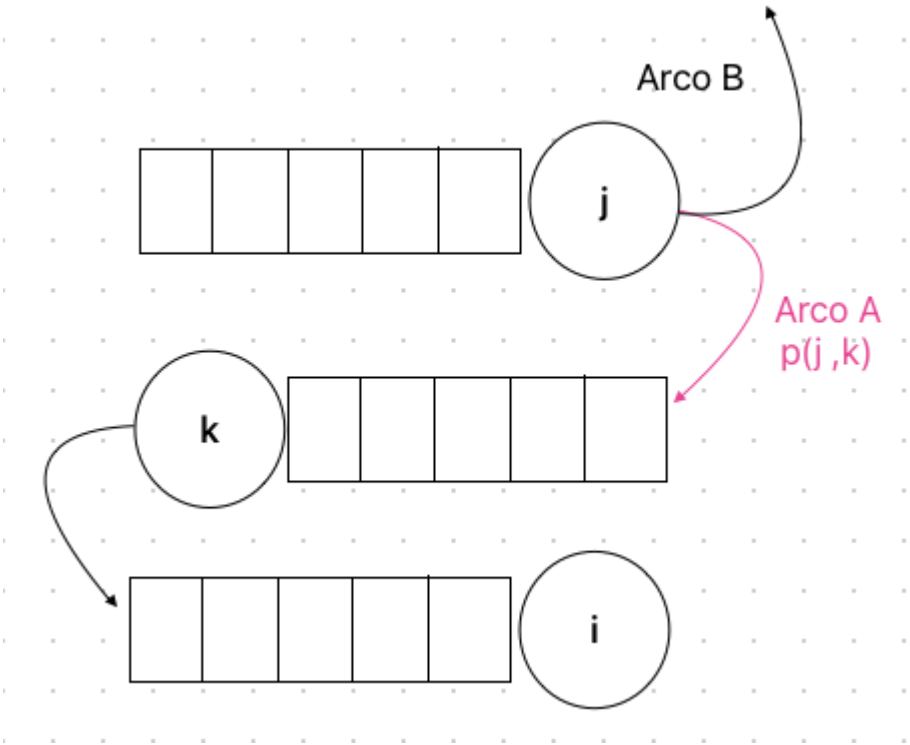
- **Servente singolo**: $X_i = \mu_i U_i$, ovvero tasso del centro i-esimo per la probabilità di non essere vuoto (leggasi anche: percentuale in cui è pieno). Deriva da Little.
- **Multi-server:** $X_i = m_i \mu_i U_i = m_i \mu_i \frac{E(c_i)}{m_i} = E(c_i)\mu_i$ ovvero occupazione media per tasso servente singolo. Andando a sostituire l'occupazione media del centro i, ciò che otteniamo è una somma pesata.
- **Infinite Server:** Similmente al caso degli indici, poniamo N al posto di m_i: $X_i = N\mu_i U_i = \sum_{j=1}^{N} j\mu_i p_i(j)$ ovvero numero dei job per tasso del singolo job, ovvero il *livello del throughput*, moltiplicato per la probabilità che vi siano j job, cioè un'altra somma pesata.

Tempi con Little

L'ultimo indice *locale* è, calcolato con Little, ed è il **tempo di residenza** $E(t_i) = \frac{E(n_i)}{X_i}$, dipendente quindi dal centro. Sarebbe il tempo di risposta del centro i. Non parliamo di *tempo di risposta generale* perchè vorrebbe dire che ad un certo punto si esca dalla rete, cosa impossibile in una rete chiusa.

Indici globali

Ha senso parlare di **throughput globale**. Devo però scegliere un riferimento, ad esempio j



Sia $p_{(j,k)}$ la probabilità di andare dal centro j al centro k. Se fossi interessato al throughput rispetto i, ovvero quanti job escono da i, cio X_i Se fossi interessato al throughput rispetto *l'arco A*, avrei $X_A = X_j p_{(j,k)}$

Se volessi il **tempo di risposta globale**, dovrei mettere un timestamp su ogni job, per segnare quando parte e quando torna. Faccio la media per un numero di job molto grandi ed ottengo il tempo di risposta medio rispetto ad i. E' *il tempo che ci mette il resto del centro a rispondere ad i*. Se sommo il tempo di risposta del singolo centro $E(Ts_i)$ ottengo il **tempo di ciclo**. La legge del **tempo di risposta** è:

$E(t_{r/i}) = \sum_{j=1, j \neq i} E(t_j) v_{j/i}$ La prima componente mi dice: "sto nel centro i, esce un certo job, dopo quanto tempo questo job rientra nella coda del centro i?" Faccio la media per più job. La seconda componente mi dice il numero di volte che il job visita i dopo aver visitato j.

Dobbiamo risolvere le **equazioni di traffico**, che presenta equazioni del tipo $\bar{y} = \bar{y}P$ con P matrice di routing che ci dice come il carico si ripartisca tra i centri. E' sempre un'uguaglianza tra ciò che entra e ciò che esce. Nell'esempio iniziale con 3 centri, avremmo:

- centro1: $y_1 = y_1 p_{1,1} + y_2 + y_3$
- centro2: $y_2 = y_1 p_{1,2}$
- centro3: $y_3 = y_1 p_{1,3}$

Sono *linearmente indipendenti* e vengono chiamati *Throughput relativi*, perchè sono rappresentazioni astratte di quale % passerà dal centro i, non dipende dal carico. Le **Visite**

sono definite come: $V_{j/i} = \frac{y_j}{y_i}$

Se fisso $y_1 \doteq 1$ allora $y_2 = p_{1,2}$ e $y_3 = p_{1,3}$, cioè sto calcolando le *visite rispetto al centro 1*. Allora ottengo $v_{1,2} = \frac{y_1}{y_2} = \frac{1}{p_{1,2}}$,

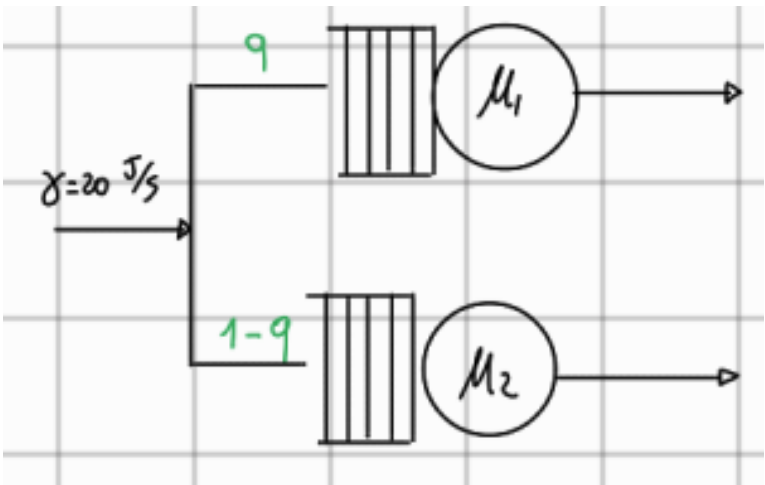
Considerazioni sul caso aperto

Non si parla più di y_i, le equazioni di traffico diventano:

- *Probabilità di uscire a partire dal centro i*: $p_{i,\emptyset} = 1 - \sum_{j=1}^{M} p_{i,j}$
- *Flusso entrante in i*: $\lambda_i = \gamma + \sum_{j=1}^{M} \lambda_j p_{j,i} \doteq X_i$ Ovvero il throughput vero stazionario, somma di ciò che entra dall'esterno e di ciò che entra dagli altri centri.
- *Visite in i*: $V_i = \frac{\lambda_i}{\gamma} = \frac{X_i}{\gamma}$, ovvero rapporto tra throughput i vero, e throughput della rete vero. E' il numero medio di volte che passo per i prima di uscire.

Esercizio

In un contesto di allocazione file, ho due dischi diversi. Il tempo di acceso al primo disco è di 35ms, per il secondo 46,5 ms. Il flusso è 20j/s. Vorrei trovare q ottimale per migliorare il tempo di allocamento e minimizzare $E(T_s)$



Il sistema è un *modello aperto*, abbiamo due risorse. Dobbiamo scrivere le equazioni di traffico, calcolare le visite, e cercare di avere **stessa utilizzazione** ρ **per entrambi**. Sembra un'idea controintuitiva. Non dovrei *massimizzare il disco più veloce*? Se facessi così, avrei un collo di bottiglia. (Il secondo disco, seppur più lento, da un contributo migliore rispetto al non usarlo proprio). L'idea è che avere utilizzazioni non uguali, comporta al creare un collo di bottiglia sul meno usato.

Ricaviamo $\mu_1 = 1/0.035 = 33.33333$ j/s e $\mu_2 = 1/0.465 = 21.74$ j/s

Trattandosi di un sistema *aperto*, parliamo di λ e non di y. Pongo $\rho_1 = \rho_2$ da cui:

Con tali dati si ottiene
rho_1 = rho_2 = 0.36317, cioè la condizione imposta.

- $\lambda_1 = q\gamma$
- $\lambda_2 = (1-q)\gamma$

I dischi non comunicano, quindi matrice di routing è nulla, ogni $p_{i,j} = 0$

Sappiamo che $\rho_i = \frac{\lambda_i}{\mu_i}$, tuttavia non possiamo nè λ_1 nè λ_2

Vediamo se le *visite* ci sono di aiuto: $v_1 = \frac{\lambda_1}{\gamma} = q$; $v_2 = \frac{\lambda_2}{\gamma} = 1 - q$

Allora posso dire che $\rho_1 = \frac{q\gamma}{\mu_1} = \rho_2 = \frac{(1-q)\gamma}{\mu_2}$

Tale equazione è vera se solo se $q = 0.6052$, ovvero il disco1 riceve il 60% del traffico.

Quale è il tempo necessario per attraversare l'intera rete?

$E(t_r) = q\frac{1}{\mu_1 - \lambda_1} + (1-q)\frac{1}{\mu_2 - \lambda_2} = 0.570286$ vale perchè non ho feedback, con feedback non lo uso.

_____ corrisponde ad R1

Questa formula viene dalla KP, essendo i tempi esponenziali, infatti: $\frac{q}{\mu_1 - \lambda_1} = \frac{q}{\mu_1 - \gamma q} = \frac{q}{\mu_1 - \rho_1\mu_1} = \frac{q}{\mu_1[1-\rho_1]} = \frac{qE[s_i]}{1-\rho_1}$

dalla prima formula alla seconda, ho usato v_1 e ricavato γq.

Successivamente $\rho_1 = \frac{\gamma q}{\mu_1}$ La formula finale è proprio $E[Ts]$ nella *M/M/1*

Sia R il tempo che va dall'entrata del job nel centro all'attimo prima di essere serviti, allora
R = v1R1 + v2R2 = 0.0570286 = tempo di risposta

v(i)= lambda(i)/gamma