Università degli Studi di Firenze
Facoltà di Ingegneria

Corso di Laurea in

Magistrale di Ingegneria Informatica

Fisica Statistica e Teoria dell'Informazione: Analisi del problema del check-in e minimizzazione costo con Simulated Annealing

Autori:

Giulia Fontanini
Pierpaolo Romani

Docente:

Prof. Franco Bagnoli

Anno Accademico 2013/2014

Indice

Introduzione

Nella presente relazione prenderemo in analisi la gestione di servizio di una coda, concentrandoci, nel particolare, nel caso realistico di una fila di attesa a un check-in aeroportuale in cui si vogliono minimizzare i costi a carico della compagnia aerea responsabile del servizio.

Per modellare questo scenario occorrerà fissare dei parametri e delle distribuzioni di probabilità di determinati eventi (saranno descritti nel capitolo 1), e utilizzeremo la strategia del *Simulated Annealing* per minimizzare i costi relativi alla gestione della coda (capitolo 2). Infine nei capitoli 3 e 4 saranno riportati e discussi alcuni risultati sperimentali del nostro lavoro.

Capitolo 1

Modellazione stocastica del problema della coda al check-in

Si consideri una distribuzione discreta di arrivi casuali al gate di un aeroporto mostrata in Figura 1.1.

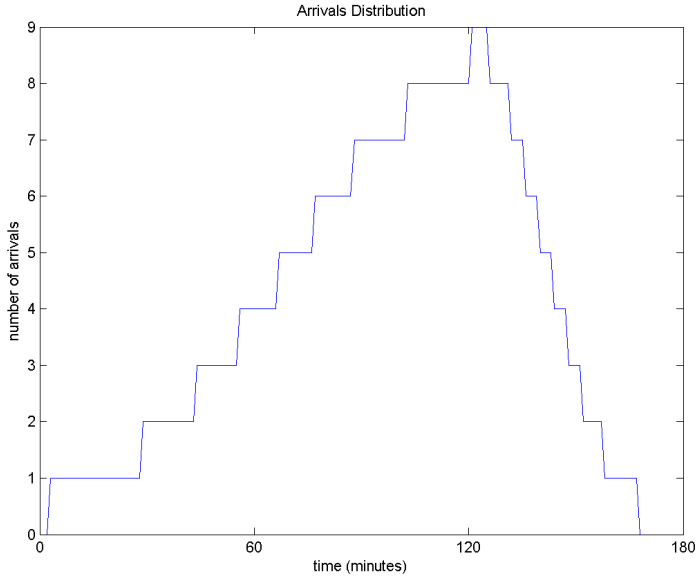

Figura 1.1: Distribuzione di arrivi casuali al gate

Tale distribuzione, che comprende un totale di 716 passeggeri, ha le seguenti caratteristiche:

- i passeggeri arrivano circa tre ore prima della partenza del volo
- vi è un picco di arrivi circa un'ora prima della partenza (minuto 120)
- gli arrivi terminano in tempo alla chiusura del gate, ovvero circa dieci minuti prima della partenza (minuto 170)

Si suppone poi che per smaltire la coda in tempo per la chiusura del gate siano disponibili sei banchi, ciascuno dei quali impiega un tempo medio $\hat{t} = 1$ minuto per servire un passeggero in coda.

In Tabella 1.1 vengono specificati i costi relativi al numero di banchi aperti, in euro all'ora.

Tabella 1.1: Costi per numero di banchi aperti

Numero di banchi aperti	1	2	3	4	5	6
Costo per banco	40	50	60	70	80	90
Costo totale	40	100	180	280	400	540

Si ipotizza che sia più conveniente, per abbassare il costo totale, non solo aprire i banchi quando necessario, ma anche tenere il numero di banchi al tempo t più basso possibile, in quanto l'apertura di un nuovo banco implica che il costo di ciascuno di essi aumenti, come mostrato in tabella.

Inoltre, abbiamo stabilito come ulteriore costo quello del rimborso biglietto a ciascun passeggero che non è stato servito entro la chiusura del gate, stimato come $penalty = 50$ euro.

Al variare delle soglie di apertura (che abbiamo fatto coincidere con le soglie di chiusura) di ciascun banco in funzione della lunghezza della coda, si possono ottenere le seguenti informazioni:

- lunghezza della coda al tempo t
- numero di banchi aperti al tempo t (e conseguente costo di apertura dei banchi)
- numero di passeggeri non serviti alla chiusura del gate (e conseguente costo di rimborso)
- costo complessivo

Ad esempio, possiamo supporre che un banco stia sempre aperto, impostando arbitrariamente le soglie di apertura degli altri banchi come mostrato in Tabella 1.2.

Tabella 1.2: Scelta arbitraria delle soglie per ciascun banco

Banco	1	2	3	4	5	6
Soglia di apertura	0	10	20	30	40	50

Con queste soglie, otteniamo i risultati ottenuti nelle Figure 1.2 e 1.3.

Inoltre, il numero di passeggeri che non sono stati serviti è pari a 24, con una conseguente penale di rimborso di 1200 euro, mentre il costo per i banchi aperti nel corso del tempo è di $939,66$ euro, per un costo complessivo di $2139,66$ euro.

L'obiettivo della parte successiva dell'elaborato è quello di minimizzare tale costo, scegliendo delle soglie di apertura ottime (e di conseguenza dei tempi di apertura ottimi) per ciascun banco, di modo che tutti i passeggeri vengano serviti prima della partenza del volo.

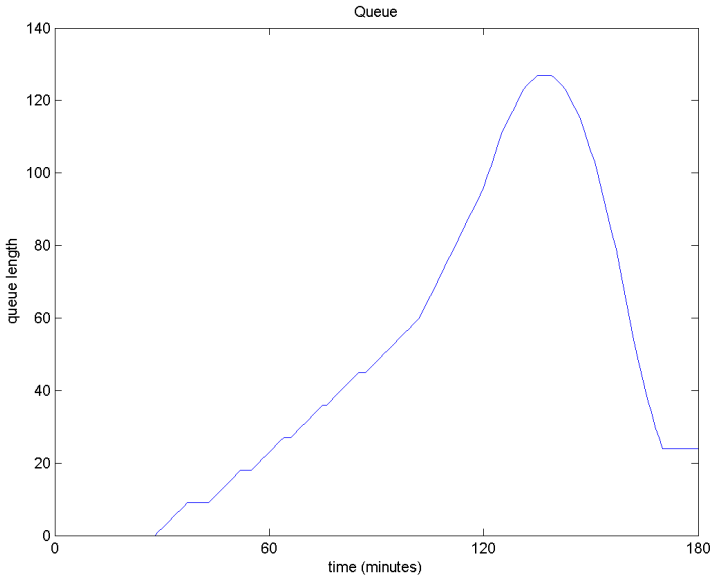

Figura 1.2: Lunghezza della coda in funzione del tempo

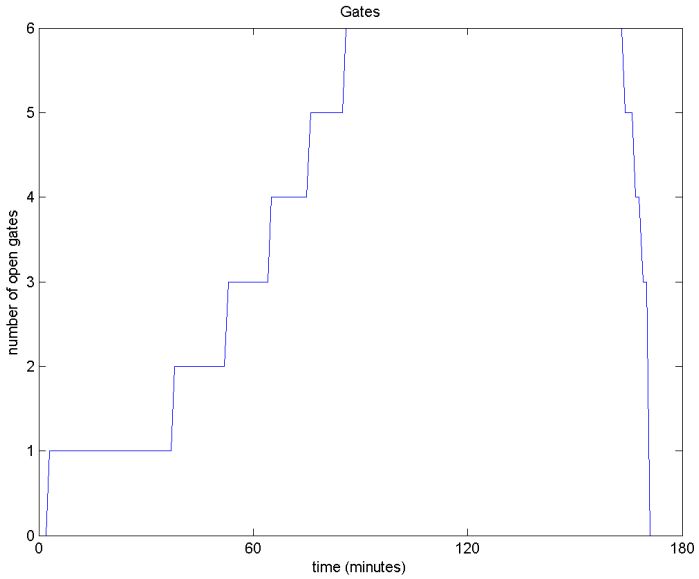

Figura 1.3: Numero di banchi aperti in funzione del tempo

Capitolo 2

Miminizzazione del costo con Simulated Annealing

Per affrontare il nostro problema di ottimizzazione abbiamo fatto ricorso alla strategia del *Simulated Annealing*, tradotto letteralmente in italiano come "ricottura simulata". L'annealing è una tecnica che trae origini dalla scienza dei metalli, dove la ricottura è il processo con il quale un solido, portato allo stato fluido mediante riscaldamento ad alte temperature, viene riportato di nuovo allo stato solido o cristallino a temperature basse controllando e riducendo gradualmente la temperatura. Ad alte temperature, gli atomi nel sistema si trovano in uno stato altamente disordinato e quindi l'energia del sistema è elevata. Per portare tali atomi in una configurazione cristallina altamente ordinata (*statisticamente*), deve essere abbassata la temperatura del sistema. Riduzioni veloci della temperatura possono causare difettosità nel reticolo cristallino con conseguente metastabilità, con fessurazioni e fratture del reticolo stesso (*stress termico*). L'annealing evita questo fenomeno procedendo ad un graduale raffreddamento del sistema, portandolo ad una struttura globalmente ottima e stabile. Il sistema si dice essere in equili-

brio termico alla temperatura T se la probabilità $P(E_i)$ di uno stato avente energia E_i è governata dalla *distribuzione di Boltzmann* [1]:

$$P(E_i) = \frac{e^{-\frac{E_i}{k_B T}}}{\sum_j e^{-\frac{E_j}{k_B T}}} \tag{2.1}$$

dove k_B è la costante di Boltzmann.

Si nota facilmente che a temperature alte i differenti stati di energia sono probabilmente possibili, mentre a temperature basse ci si trova sicuramente in stati con bassa energia.

Nel campo dell'ottimizzazione la temperatura T ha valore metaforico e non rappresenta altro che un parametro che regola l'incertezza con la quale si accetta o meno un certo valore come minimo. In altre parole, il *simulated annealing* è un metodo di ricerca del minimo che cerca di ovviare al problema di incappare in minimi locali (anziché globali) introducendo una certa probabilità (secondo la distribuzione di Boltzman) di accettare una data configurazione come minimo del problema. Andando nel dettaglio, l'algoritmo prevede

1. sia data una configurazione iniziale con energia E_0 e temperatura T_0;
2. per ogni stadio della temperatura (andrà a calare a ogni passo) effettua i seguenti comandi:
   (a) genera una configurazione ammissibile tramite una piccola perturbazione casuale della configurazione corrente. Valuta la differenza di energia ΔE fra le due configurazioni;
   (b) se $\Delta E \leq 0$:

- la configurazione trovata ha un valore della funzione obbiettivo inferiore rispetto a quello della configurazione corrente. Accetta la nuova soluzione e sostituiscila a quella corrente;

altrimenti:

- la configurazione candidato ha un valore della funzione obiettivo peggiore rispetto quello della configurazione corrente. Accetta tale soluzione con una probabilità

$$P(\Delta E) = e^{-\frac{\Delta E}{k_B T}} \tag{2.2}$$

e aggiorna la configurazione corrente se necessario;

(c) Se non è raggiunto l'equilibrio termico, torna allo step (a). Altrimenti vai a 3.

3. Se il processo annealing è incompleto, ridurre la temperatura e ritornare a 2.

Di seguito riportiamo lo pseudocodice relativo [2]:

$s \leftarrow s0;\ \ e \leftarrow E(s)$ // Stato iniziale e energia

$sbest \leftarrow s; ebest \leftarrow e$ // Soluzione migliore iniziale

$k \leftarrow 0$

$while\ \ k < kmax\ \ and\ \ e > emax$

$T \leftarrow temperature(k/kmax)$ // Calcola la temperatura

$snew \leftarrow neighbour(s)$ // Prendi alcuni stati vicini

$enew \leftarrow E(snew)$ // Computa la sua energia

$if P(e, enew, T) > random()$ // Ci muoveremo verso il nuovo stato?

$s \leftarrow snew; e \leftarrow enew$ // Sì, cambia stato

$if enew < ebest$ // È un nuovo ottimo?

$sbest \leftarrow snew; ebest \leftarrow enew$ // Salavalo nei best found

$k \leftarrow k + 1$

return sbest

Come si può notare, il ciclo *while* è governato da un numero massimo $kmax$ di iterazioni e da una soglia di energia massima $emax$ al di sotto della quale si esce dal loop. Nella prima riga sotto al *while* notiamo il calcolo della temperatura attraverso la funzione *temperature()* che prende in ingresso il valore della divisione del valore di k corrente per quello (invariante) di $kmax$.

Tale funzione si presta a diverse varianti di implementazioni. Tipicamente, in analogia con le tecniche di lavorazione dei metalli introdotte a inizio capitolo, si modella tale funzione di modo che più la temperatura si avvicina allo zero minore sarà il passo con cui questa diminuirà negli step successivi.

La chiamata di *neighbour(s)* restituisce un "vicino" dello stato s scelto in maniera casuale, la funzione $E(snew)$ calcola l'energia relativa allo stato $enew$, infine $P(e, enew, T)$ è un'implementazione dell'equazione 2.2 (dove e ed $enews$ sono usate per calcolare ΔE).

Capitolo 3

Risultati

L'utilizzo dell'algoritmo Simulated Annealing per la minimizzazione del costo, date le soglie iniziali in Tabella 1.2, produce i risultati mostrati nelle Figure 3.1 e 3.2.

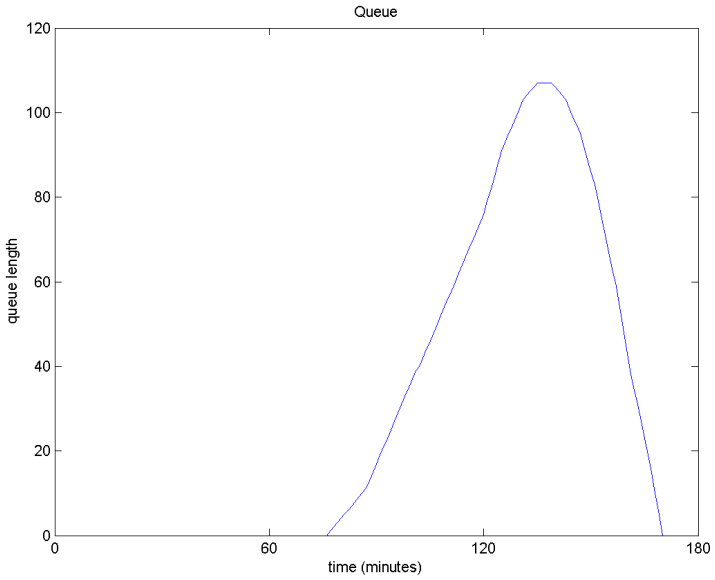

Figura 3.1: Lunghezza della coda in funzione del tempo. Tutti i passeggeri vengono serviti entro la chiusura del gate.

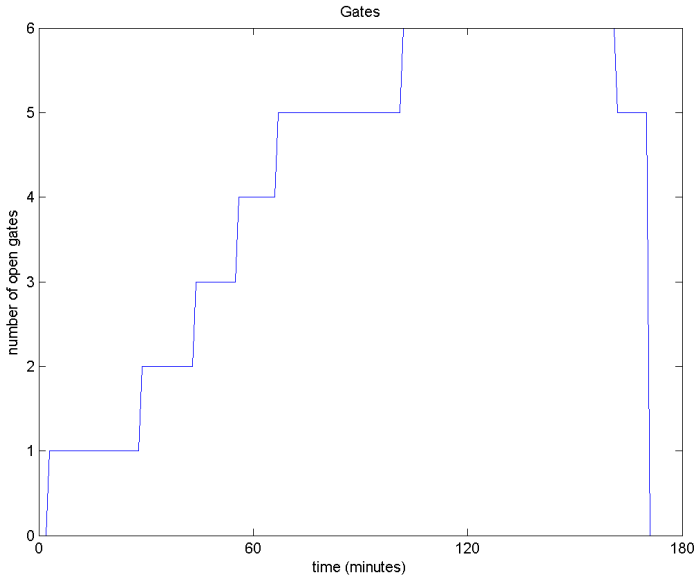

Figura 3.2: Soglie temporali ottime di apertura e chiusura di ciascun banco.

I valori delle soglie ottime di apertura di ciascun banco in funzione della lunghezza della coda sono mostrati in Tabella 3.1.

Tabella 3.1: Soglie ottime di lunghezza coda per cui aprire il corrispondente banco

Banco	1	2	3	4	5	6
Soglia di apertura	0	1	2	3	4	44

Analogamente, in Tabella 3.2, vengono mostrate le soglie ottime temporali di apertura e chiusura.

Poichè in questo caso tutti i passeggeri sono stati serviti, non c'è nessuna penale di rimborso da pagare e il costo totale, definito soltanto dal tempo di apertura di ciascun banco, è dato da 963 euro.

Per differenti valori di inizializzazione delle soglie, l'algoritmo di minimizzazione ricade sempre nello stesso costo, scostandosi di poco solo in alcuni casi: ciò è dovuto alla randomicità intrinseca del Simulated Annealing, che

Tabella 3.2: Soglie temporali ottime di apertura e chiusura di un banco

Banco	1	2	3	4	5	6
Soglia di apertura (minuto)	3	29	44	56	67	102
Soglia di chiusura (minuto)	170	170	170	170	170	161

può incappare in minimi locali (tutti molto simili tra loro) e all'arrotondamento a valori interi dei valori delle soglie (intese come lunghezza della coda, pertanto valori discreti).

Capitolo 4

Conclusioni

Nel nostro lavoro abbiamo preso in analisi la gestione di servizio di una coda, tramite l'esempio realistico di una fila di attesa a un check-in aeroportuale e la volontà di voler minimizzare i costi di gestione. Per fare ciò abbiamo implementato (in Matlab) un programma che, data una distribuzione di arrivo dei passeggeri, fissati i costi relativi all'apertura di ogni gate e alla (eventuale) mancata erogazione del servizio, cerca di ottimizzare l'utilizzo dei gates, cambiando le soglie della dimensione della coda alla quale un dato gate si attiva, in modo da trovare delle soglie che realizzino il costo di gestione minimo. Per cercare tale minimizzazione abbiamo fatto ricorso alla strategia del *Simulated Annealing* ottenendo dei risultati ottimi.

In maniera naturale, il nostro esperimento si presta a numerose varianti realizzabili, per esempio, cambiando la distribuzione di arrivo dei passeggeri, i costi associati ai vari gates e il loro numero (si pensi di passare da un aeroporto come l'Amerigo Vespucci di Firenze all'Hartsfield-Jackson di Atlanta).

Bibliografia

[1] V. Lacagnina, "Simulated annealing," *http://www.unipa.it/valerio.lacagnina/pub/ricercaOperativa/SimulatedAnnealing.pdf*.

[2] Simulated annealing, wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Simulated_annealing