Lorenzo Grassi 904123

Report progetto nr.2

Caratteristiche del simulatore

Per lo sviluppo di questo simulatore sono stati identificati sei possibili eventi:

- "FAILED": è il primo evento in ordine cronologico che può essere associato ad un customer quando esce dalla Delay Station (nel codice i customer molto spesso sono chiamati con la variabile "item"). All'inizio della simulazione vengono generati N (il numero dei customer nel sistema) eventi FAILED.

- "FAULTY": è l'evento che viene assegnato quando un customer passa dalla coda Test alla coda delle riparazioni veloci della stazione Repair. Questo evento capita solo dopo l'evento FAILED.

- "DAMAGED_REPAR": è l'evento che viene assegnato quando un customer passa dalla coda Test alla coda delle riparazioni lunghe della stazione Repair (coda damaged). Questo evento può capitare dopo gli eventi FAILED, DAMAGED_SPARE o SERVICED.

- "DAMAGED_SPARE": è l'evento che viene assegnato quando un customer passa dalla coda damaged alla coda dei pezzi di ricambio (spare-parts). Questo evento capita solo dopo l'evento DAMAGED_REPAIR

- "SERVICED": è l'evento che viene assegnato quando un customer passa dalla coda damaged alla coda Fix della stazione Test/Fix .Questo evento capita solo dopo l'evento DAMAGED_REPAIR.

- "REPAIRED": è l'ultimo evento in ordine cronologico che può essere assegnato ad un customer prima di ritornare nella Delay station. Può capitare dopo gli eventi FAILED, FAULTY o SERVICED.

Il metodo "engine" processa la Future Event List e legge il tipo di evento corrispondente all'item. In generale, una volta letto il tipo di evento, viene chiamato un altro metodo il cui nome corrisponde alla stazione del modello; nello specifico se viene processato l'evento FAILED o SERVICED viene chiamato il metodo "test_fix", per gli eventi FAULTY e DAMAGED_RAPAIR viene chiamato il metodo "repair", per l'evento "DAMAGED_SPARE" viene chiamato il metodo "spare_parts" e per l'evento REPAIRED viene chiamato il metodo "repaired".

Il funzionamento dei metodi che gestiscono gli specifici eventi si basa innanzitutto su liberare la coda da dove proviene il customer e quindi (se nella coda c'è qualcuno) programmare il prossimo evento generato da quella coda, successivamente, se la stazione (nel caso degli eventi FAILED, FAULTY, DAMAGED_REPAIR o SERVICED) o la singola coda (il caso DMAGED_SPARE) è libera, servire il customer e programmare il prossimo evento, altrimenti inserire il customer nella coda specifica.
Per esempio, prendiamo il caso in cui il metodo "engine" processi un evento DAMAGED_REPAIR che abbia come evento precedente FAILED. In questo caso viene chiamato il metodo "repair", il quale controlla qual è l'evento precedente dell'item,

aggiorna le variabili che servono a contare quanti customer sono presenti nelle code coinvolte (in questo caso la coda failed e la stazione test/fix) e programma il prossimo evento dalla coda/stazione precedente tramite un altro metodo. Nell'esempio il metodo controlla la stazione Test/fix e nel caso in cui la priorità sia sugli eventi test, si valuta prima se nella coda test c'è qualcuno, in caso positivo si procede a programmare il prossimo evento che da, sempre nell'esempio, FAILED potrà diventare FAULTY, REPAIRED o DAMAGED_REPAIR secondo le routing probability (questo procedimento viene svolto nel prima nel metodo nextTestFix() che libera la coda precedente e poi nel metodo exitFailed() che sceglie il prossimo evento, per le altre code il procedimento è il medesimo). A questo punto si passa a controllare se la stazione repair è vuota e in caso positivo si programma il prossimo evento che potrà essere DAMAGED_SPARE o SERVICED; questa operazione viene fatta dal metodo "exit_damaged" che tenendo conto delle routing probability sceglie il tipo del prossimo evento e lo temporalizza secondo la distribuzione iperesponenziale inserendolo nella future event list.

Per trovare i **regeneration point**, al momento della creazione degli item FAILED, si sceglie un item e lo si marca come TAGGED = true; quando questo item tagged arriva ad essere un evento REPAIRED, quindi può tornare nella delay station, si controlla che non ci sia nessun customer che abbia un service time generato da una distribuzione non esponenziale, quindi o uniforme o iperesponenziale; in sostanza si effettua un controllo per determinare se la coda test e la coda damaged siano entrambe vuote, in caso positivo si chiama il metodo "regenerate" che calcola le statistiche per trovare il tempo il tempo medio dei customer da quando passano allo stato damaged a quando ritornano repaired. Nel caso in cui invece ci siano customer attivi con service time non esponenziale, si riprogramma un altro evento tagged e si continuano a raccogliere le statistiche.

Per quanto riguarda le **statistiche** e nello specifico come sono stati trovati i valori del tempo medio in cui un customer è nello stato damaged, innanzitutto, quando l'item entra nel metodo "repair" (che sarebbe la stazione repair del modello) e dopo aver controllato che il tipo dell'item sia DAMAGED_REPAIR viene salvato il valore del clock in quel momento dentro una variabile; nel momento in cui l'item arriva al metodo "repaired", viene calcolata la differenza tra il valore del clock attuale e quello salvato in precedenza per trovare il tempo in cui l'item è stato nello stato damaged (compreso eventualmente il tempo che ha passato nelle stato fix e/o spare_parts), il valore viene aggiunto ad una variabile globale che contiene la somma di tutti questi tempi. A questo punto viene anche incrementata una variabile globale che segna quanti item hanno raggiunto lo stato repaired dopo essere stati damaged all'interno del ciclo. Se il regeneration point viene trovato (tramite il meccanismo spiegato nel paragrafo precedente) viene calcolata la media del tempo damaged del ciclo, aggiunta questa media ad un'altra variabile che contiene la somma di tutte le medie precedenti, viene calcolata la media su quest'ultima variabile su il numero di cicli finora compiuti. Se, dopo aver calcolato gli estremi dell'intervallo di confidenza la precisione è ancora più grande di quella richiesta, allora viene eseguito un altro ciclo altrimenti la simulazione si ferma salvando gli ultimi valori degli intervalli calcolati.

L'**output** del simulatore è riportato a pagina 7 del report.

Primo step di validazione

Per realizzare il primo step di validazione, è stato modificato il simulatore in modo che tutte le istanze di numeri casuali generati siano di distribuzioni esponenziale e le stazioni Test/Fix e Repair sono state modificate in modo che ci siano quattro code indipendenti senza il meccanismo della priorità. Nello specifico quando un customer, per esempio, entra nella coda Faulty, non controlla più che la stazione che comprende anche la coda damaged repair sia libera ma tenta direttamente di entrare nella coda faulty.

Bottleneck Analysis

Visit Count

V1 = 1, V2 = 1, V3 = 0.28, V4 = 0.5, V5 = 0.4, V6 = 0.12

Dove V1 è la infinite service station, V2 e V3 sono nella Test/Fix, V4 e V5 sono nella Repair e V6 corrisponde alla Spare-parts.

Service time medi di un customer nelle varie code
D2 = S1 * V2 = 5
D3 = S'1 * V3 = 1.12
D4 = S2 * V4 = 25
D5 = S'2 * V5 = 80
D6 = S'3 * V6 = 60

Da questi calcoli la **bottleneck station**, senza contare la delay station, è la 5.

Il **massimo throughput** del sistema è 1/D5 = 0.125

Il **cycle time** del sistema con un customer è:
R(1) = D2 + D3 + D4 + D5 + D6 = 171.12

Il **punto di saturazione** derivato dall'equazione R(1) = N * D5 - Z è:
N* = 33.389

Il numero di Customer minimo per far si che si formi coda all'interno del sistema è:
N*s = R(1)/D5 = 2.129

Il numero medio di Customer nel sistema in condizioni di saturazione è:
N*t = Z/D5 = 31.25

Validazione con simulatore e algoritmo MVA

La classe “ComputationalAlgorithm.java” implementa l’algoritmo MVA sul modello semplificato del sistema. Vengono riportati di seguito i risultati con N = 20 (lo stesso numero usato dal simulatore), l’output completo è nel file MVA.txt.

```
System with 20 customers inside **************************


Station number: 1 ++++++++++++
Average number in queue:        17.845871465676716
Average waiting time:           2500.0
Average throughput:             0.007138348586270687
Average utilization:            0.8922935732838357

Station number: 2 ++++++++++++
Average number in queue:        0.03694749734779357
Average waiting time:           5.175916656529685
Average throughput:             0.007138348586270687
Average utilization:            0.03569174293135343

Station number: 3 ++++++++++++
Average number in queue:        0.008056367875682065
Average waiting time:           4.030728125058135
Average throughput:             0.0019987376041557922
Average utilization:            0.007994950416623169

Station number: 4 ++++++++++++
Average number in queue:        0.2146370178072176
Average waiting time:           60.136322908083486
Average throughput:             0.0035691742931353433
Average utilization:            0.17845871465676716

Station number: 5 ++++++++++++
Average number in queue:        1.1854157785606547
Average waiting time:           415.15756909118517
Average throughput:             0.002855339434508275
Average utilization:            0.571067886901655

Station number: 6 ++++++++++++
Average number in queue:        0.7090718727319316
Average waiting time:           827.7730067891127
Average throughput:             8.566018303524823E-4
Average utilization:            0.42830091517624114
```

Il valore teorico derivato dall’algoritmo MVA per il tempo medio di un customer damaged sarà sicuramente più alto dell’average waiting time della stazione 5 dato che il 30% dei customer finisce nella stazione 6 che ha un average waiting time più alto.

Il valori degli intervalli prodotti dal simulatore del modello semplificato sono:

```
-------------------------------------------------------------
Interval of average total repair time of damaged machines:

left limit: 427.7280706065865, right limit: 472.65877409213664
reached precision: 4.9901554815124225%, after: 359 cycles
Percentage of times the theoretical value is within the calculated
interval of the cycle 90.80779944289694%
-------------------------------------------------------------
```

In effetti i risultati del simulatore confermano le ipotesi precedenti. Impostando all'interno del simulatore un valore teorico di 423, la percentuale di volte in cui questo valore è all'interno dell'intervallo calcolato in un ciclo è circa il 90%.

Il valore teorico derivato dall'algoritmo MVA per il tempo medio di un customer faulty è circa 60 (= average waiting time della stazione 4).

Il valori degli intervalli prodotti dal simulatore del modello semplificato sono:

```
------------------------------------------------------------
Interval of average total repair time of faulty machines:

left limit: 55.2329593324079, right limit: 60.88797114901979
reached precision: 4.869933261098304%, after: 28 cycles
Percentage of times the theoretical value is within the calculated
interval of the cycle 96.42857142857143%
------------------------------------------------------------
```

Anche in questo caso il valore teorico è coperto dagli estremi dell'intervallo e più 90% delle volte ricade all'interno degl'intervalli calcolati nel ciclo.

Secondo step di validazione

Catene di Markov

Per questo step è stato prima di tutto definito lo state space, il cui schema si trova nel file "State space.pdf" in cui sono stati presi in considerazione gli aspetti della priorità sulla stazione repair (la coda damaged ha la priorità) e la distribuzione iperesponenziale per la coda damaged; successivamente è stato trasformato in matrice nel file "transition rates matrix.xlsx". La matrice è stata manipolata nello script "PiVector.m" in modo da ottenere la trasposta, sottratte a tutti gli elementi della diagonale la somma degli elementi sulle colonne e aggiungere la condizione di normalizzazione. È stato quindi risolto il sistema di equazioni utilizzando come matrice dei coefficienti la matrice appena creata e vettore dei termini noti un vettore colonna con tutti 0 e ultimo elemento uguale a 1. Il vettore delle soluzione corrisponde al vettore delle steady-state probability distrubution (è riportato sotto la matrice nel foglio Excel). Di seguito si riportano i risultati ottenuti tramite le catene di Markov.

Coda x0000	
P(0)	0,041952
P(1)	0,047277
P(2)	0,060108
P(3)	0,807859
P(x0000)	**2,59107**
ro	0,958048
lambda	0,000383
waiting t	**6761,328**

Coda 0x000	
P(0)	0,99464
P(1)	0,00534
P(2)	2E-05
P(3)	3,9E-08
P(0x000)	**0,00538**
ro	0,00536
lambda	0,00107
waiting tim	**5,01895**

Coda 00x00	
P(0)	0,9981
P(1)	0,00189
P(2)	4,5E-06
P(3)	4,3E-09
P(00x00)	**0,0019**
ro	0,0019
lambda	0,00047
waiting time	**4,00943**

Coda 000x0	
P(0)	0,90627
P(1)	0,0711
P(2)	0,02263
P(3)	7,3E-06
P(000x0)	**0,11638**
ro	0,09373
lambda	0,00187
waiting ti	**62,0773**

Coda 0000x	
P(0)	0,84038
P(1)	0,12111
P(2)	0,03715
P(3)	0,00136
P(0000x)	**0,1995**
ro	0,15962
lambda	0,0008
waiting ti	**249,96**

La coda x0000 sarebbe la delay station, la coda 0x000 è la coda test, la coda 00x00 è la coda fix, la coda 000x0 è la coda faulty mentre la coda 0000x è la coda delle riparazioni lunghe.
Di seguito sono invece riportati i risultati dell'algoritmo MVA con 3 customer nel sistema (lo stesso numero usato per costruire la catena di Markov).

```
System with 3 customers inside ***************************


Station number: 1 ++++++++++++
Average number in queue:        2.8662383706441634
Average waiting time:           2500.0
Average throughput:             0.0011464953482576653
Average utilization:            0.9554127902147211

Station number: 2 ++++++++++++
Average number in queue:        0.0057544501470761025
Average waiting time:           5.019165717349983
Average throughput:             0.0011464953482576653
Average utilization:            0.005732476741288326

Station number: 3 ++++++++++++
Average number in queue:        0.0012851756924701985
Average waiting time:           4.003429402804713
Average throughput:             3.210186975121463E-4
Average utilization:            0.0012840747900485852

Station number: 4 ++++++++++++
Average number in queue:        0.029215918468467772
Average waiting time:           50.965611875996444
Average throughput:             5.732476741288327E-4
Average utilization:            0.028662383706441634

Station number: 5 ++++++++++++
Average number in queue:        0.0975060850478224
Average waiting time:           212.6177075118597
Average throughput:             4.5859813930306616E-4
Average utilization:            0.09171962786061323

Station number: 6 ++++++++++++
Average number in queue:        0.0
Average waiting time:           0.0
Average throughput:             1.3757944179091982E-4
Average utilization:            0.0
```

In questa versione l'algoritmo MVA è stato modificato in modo che il service time della stazione 6 sia uguale a 0; l'output completo di questa versione è nel file MVA2.txt.
Anche in questo caso i valori del numero medio di persone in coda e il waiting time medio combaciano abbastanza con i risultati dei calcoli derivati dalla catena di Markov.
Di seguito si riportano i risultati del simulatore in cui non ci sono distribuzioni uniformi ma iperesponenziali (coda damaged) e esponenziali, c'è il meccanismo della stazione con priorità che è stato lasciato solo nella stazione repair con priorità sulla coda damaged.

```
------------------------------------------------------------
Interval of average total repair time of damaged machines:

left limit: 205.78556657797037, right limit: 227.4410672485268
reached precision: 4.998654048409508%, after: 13198 cycles
------------------------------------------------------------
```

Anche con questo step di validazione i risultati del simulatore corrispondono abbastanza sia ai risultati dell'algoritmo MVA che a quelli ottenuti con le catene di Markov.

Performance del sistema

1.Priorità alle macchine failed e faulty e tempi di riparazione con distribuzione iperesponenziale

```
----------------------------------------
Interval of average total repair time of damaged machines:

left limit: 664.9402892189768, right limit: 734.9231031724117
reached precision: 4.99926023737811%, after: 2730 cycles
----------------------------------------
```

2.Priorità alle macchine failed e faulty e tempi di riparazione con distribuzione esponenziale

```
----------------------------------------
Interval of average total repair time of damaged machines:

left limit: 583.1835547387142, right limit: 644.5478348687365
reached precision: 4.998184509206257%, after: 333 cycles
----------------------------------------
```

3.Priorità alle macchine serviced e damaged e tempi di riparazione con distribuzione iperesponenziale

```
----------------------------------------
Interval of average total repair time of damaged machines:

left limit: 627.7338107878791, right limit: 693.8040559193616
reached precision: 4.999496934287926%, after: 3884 cycles
----------------------------------------
```

4.Priorità alle macchine serviced e damaged e tempi di riparazione con distribuzione esponenziale

```
----------------------------------------
Interval of average total repair time of damaged machines:

left limit: 375.3790721004119, right limit: 414.86411972702217
reached precision: 4.996569161867912%, after: 343 cycles
----------------------------------------
```