Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Συστήματα Αναμονής - Ροή Δ

4η Ομάδα Ασκήσεων - Αναφορά

Ονοματεπώνυμο:	**Αριθμοί Μητρώου:**
Σταθά Ευσταθία	el16190

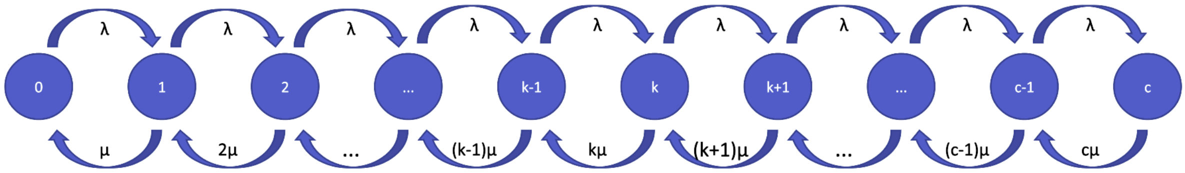
Πίνακας Περιεχομένων

Άσκηση 1 - Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου

Λύσεις - Σχόλια

Ερώτημα (1)

Το διάγραμμα ρυθμού μεταβάσεων της ουράς M/M/c/c, όπως είδαμε στο μάθημα, έχει την ακόλουθη μορφή:



Από το οποίο προκύπτουν οι ακόλουθες συναρτήσεις ισορροπίας:

$$\lambda P_0 = \mu P_1 \Rightarrow P_1 = \rho P_0$$
$$\lambda P_1 = 2\mu P_2 \Rightarrow P_2 = \frac{\rho^2}{2} P_0$$
$$\lambda P_{k-1} = k\mu P_k \Rightarrow P_k = \frac{\rho^k}{k!} P_0$$

επομένως, ισχύει,

$$P_k = \frac{\rho^k}{k!} P_0,\ k = 1, 2, .., c.$$

Όμως, το άθροισμα των πιθανοτήτων όλων των καταστάσεων πρέπει να είναι 1 κι, έτσι:

$$P_0 + P_1 + ... + P_{c-1} + P_c = 1 \Rightarrow$$
$$\Rightarrow P_0 = \frac{1}{\sum_{k=1}^{c} \frac{\rho^k}{k!}}.$$

Η πιθανότητα απόρριψης πελάτη ισούται με την πιθανότητα να βρίσκονται στο σύστημα c πελάτες, διότι σε αυτή την περίπτωση όλοι οι εξυπηρετητές είναι κατειλημμένοι. Επομένως,

$$\mathrm{P}_{blocking} = \frac{\rho^c/c!}{\sum_{k=0}^{c} \rho^k/k!},\ \rho = \frac{\lambda}{\mu}.$$

Όσων αφορά το μέσο ρυθμό απωλειών, γι’ αυτόν γνωρίζουμε πως ισούται με γ-λ, όπου γ η ρυθμαπόδοση του συστήματος και λ ο ρυθμός άφιξης πελατών. Επομένως:

$$\lambda - \gamma = \lambda - \lambda(1 - P_{blocking}) = \lambda P_{blocking} = \lambda \frac{\rho^c/c!}{\sum_{k=0}^{c} \rho^k/k!}.$$

Ο κώδικας της συνάρτησης erlangb_factorial παρατίθεται στη συνέχεια:

```
function result = erlangb_factorial(r,c)
  nom = r^c/factorial(c);
  den = 1;
  for k = 1:c
    den += (r^k/factorial(k));
  endfor
  result = nom/den;
endfunction
```

Ερώτημα (2)

Ο κώδικας της συνάρτησης erlangb_iterative παρατίθεται στη συνέχεια:

```
function result = erlangb_iterative(r,n)
 if (n > 0)
   prev = erlangb_iterative(r,n-1);
   result = (r*prev)/(r*prev+n);
 else
   result = 1;
 endif
endfunction
```

Ερώτημα (3)

Όπως φαίνεται και στο ακόλουθο στιγμιότυπο οθόνης (αφού πρώτα χρειάστηκε να αλλάξουμε το max_recursion_depth και max_stack_depth, προκειμένου να επιτραπεί ο υπολογισμός) το αποτέλεσμα της συνάρτησης erlangb_iterative είναι όμοιο με το αποτέλεσμα της συνάρτησης erlangb του πακέτου queueing του Octave. Ωστόσο, δεν παρατηρούμε το ίδιο και για την συνάρτηση erlangb_factorial, η οποία αδυνατεί να βγάλει αριθμητικό αποτέλεσμα και, ως εκ τούτου, επιστρέφει NaN. Η αδυναμία της συνάρτησης αυτής να δώσει αποτέλεσμα για τον συγκεκριμένο υπολογισμό, οφείλεται στο γεγονός πως χρειάζεται να υπολογιστούν πολύ μεγάλα νούμερα (όπως το 1024! και το 1024^{1024}), τα οποία το Octave δεν είναι σε θέση να διαχειριστεί.

```
>> erlangb(1024,1024)
ans =  0.024524
>> erlangb_iterative(1024,1024)
ans =  0.024524
>> erlangb_factorial(1024,1024)
ans =  NaN
```

Ερώτημα (4-α)

Γνωρίζουμε πως η εταιρεία απασχολεί 200 εργαζομένους, επομένως έχουμε πως:

$$\lambda = \frac{200clients}{60min}.$$

Ακόμη, γνωρίζουμε πως ο πιο απαιτητικός χρήστης χρησιμοποιεί το τηλέφωνο του για εξωτερικές κλήσεις κατά μέσο όρο 23 λεπτά σε μια ώρα και εμείς καλούμαστε να δουλέψουμε με αυτό τον χρήστη ως πρότυπο, επομένως:

$$E[S] = \frac{1}{\mu} = 23min.$$

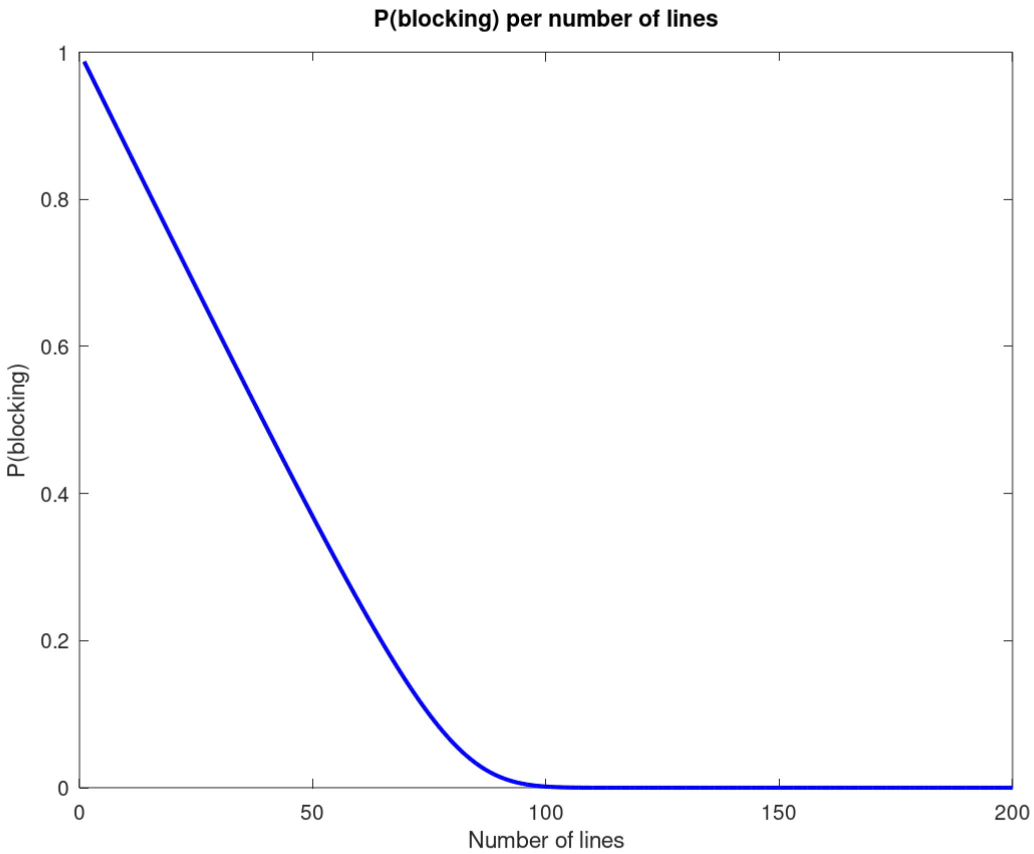
Επομένως, μπορούμε εύκολα να υπολογίσουμε τη συνολική ένταση του φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας:

$$\rho = \frac{\lambda}{\mu} = \frac{200}{60} \cdot 23 \Rightarrow$$

$$\Rightarrow \rho = 76.76\, Erlangs.$$

Ερώτημα (4-β)

Το ζητούμενο διάγραμμα είναι το ακόλουθο:



Ερώτημα (4-γ)

Όπως φαίνεται και από το διάγραμμα του προηγούμενου ερωτήματος, ο μικρότερος δυνατός αριθμός τηλεφωνικών γραμμών που επιτρέπει η πιθανότητα απόρριψης τηλεφωνικής κλήσης να είναι μικρότερη από 1% είναι σημαντικά μικρότερος από τις 200 γραμμές, τις οποίες η εταιρεία παρέχει ως τώρα στους εργαζομένους της. Ο αριθμός των γραμμών φαίνεται να είναι κοντά στις 100, ωστόσο για τον ακριβή υπολογισμό του χρησιμοποιήθηκε το Octave. Ο τελικός αριθμός, όπως φαίνεται και στο ακόλουθο στιγμιότυπο οθόνης, είναι 93 τηλεφωνικές γραμμές.

```
Number of lines needed so that P(blocking) is less than 1% is: 93.
```

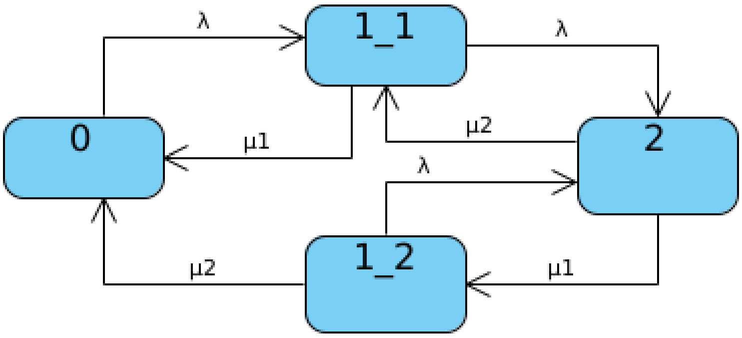
Κώδικας

```
p_blocking = zeros(1,200);
lambda = 200/60;
mu = 1/23;
r = lambda/mu;
found = false;

for i = 1:200
  p_blocking(i) = erlangb_iterative(r, i);
  if (p_blocking(i) < 0.01 && !found)
    answer = i;
    found = true;
  endif
endfor

plot(p_blocking, 'b', 'linewidth', 2);
title("P(blocking) per number of lines");
xlabel("Number of lines");
ylabel("P(blocking)");

display(["Number of lines needed so that P(blocking) is less than 1% is: ", num2str(answer), "."]);
```

Άσκηση 2 - Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές

Λύσεις - Σχόλια

Ερώτημα (1)



1_1: Ένας πελάτης βρίσκεται στο σύστημα και εξυπηρετείται από τον εξυπηρετητή 1
1_2: Ένας πελάτης βρίσκεται στο σύστημα και εξυπηρετείται από τον εξυπηρετητή 2

Για το σύστημα ισχύουν οι ακόλουθες εξισώσεις ισορροπίας:

$$\lambda P_0 = \mu_1 P_{1\_1} + \mu_2 P_{1\_2} \Rightarrow P_0 = 0.8P_{1\_1} + 0.4P_{1\_2}$$
$$(\mu_1 + \mu_2)P_2 = \lambda P_{1\_1} + \lambda P_{1\_2} \Rightarrow 1.2P_2 = P_{1\_1} + P_{1\_2}$$
$$(\lambda + \mu_1)P_{1\_1} = \lambda P_0 + \mu_2 P_2 \Rightarrow 1.8P_{1\_1} = P_0 + 0.4P_2$$
$$(\lambda + \mu_2)P_{1\_2} = \mu_1 P_2 \Rightarrow 1.4P_{1\_2} = 0.8P_2$$

Παράλληλα, ισχύει πως:

$$P_0 + P_{1\_1} + P_{1\_2} + P_2 = 1$$

Οπότε, οι εργοδικές πιθανότητες του συστήματος είναι οι ακόλουθες:

$$P_0 = 0.24951$$
$$P_{1\_1} = 0.21443$$
$$P_{1\_2} = 0.19493$$
$$P_2 = 0.34113$$

Και, όπως γνωρίζουμε, η πιθανότητα απόρριψης πελάτη ισούται με την πιθανότητα να υπάρχουν 2 πελάτες στο σύστημα. Επομένως,

$$P_{blocking} = 0.34113$$

Τέλος, ο μέσος αριθμός πελατών στο σύστημα υπολογίζεται όπως φαίνεται στη συνέχεια:

$$E[n] = \sum_{k=0} {}_{2} kP_k = 0 \cdot P_0 + 1 \cdot (P_{1\text{-}1} + P_{1\text{-}2}) + 2 \cdot P_2 \Rightarrow E[n] = 1.0916\, clients$$

Ερώτημα (2)

Το απόσπασμα του κώδικα όπου φαίνεται ο τρόπος με τον οποίο συμπληρώθηκαν τα κενά του προγράμματος είναι το ακόλουθο:

```
threshold_1a = lambda/(lambda+m1);
threshold_1b = lambda/(lambda+m2);
threshold_2_first = lambda/(lambda+m1+m2);
threshold_2_second = (lambda+m1)/(lambda+m1+m2);
```

Η προσομοίωση που δίνεται έχει ως κριτήριο σύγκλισης το απόλυτο της διαφοράς μεταξύ δύο διαδοχικών τιμών του μέσου αριθμού πελατών στο σύστημα. Συγκεκριμένα, η προσομοίωση τερματίζεται όταν η διαφορά αυτή γίνει μικρότερη από 0.001%.

Όσων αφορά τις πιθανότητες που υπολογίζονται από την προσομοίωση, αυτές φαίνονται στο ακόλουθο στιγμιότυπο οθόνης που λήφθηκε από το Octave. Οι ίδιες είναι αρκετά κοντά με αυτές που υπολογίστηκαν προηγουμένως. Παρατηρήθηκε πως σε διαφορετικές διεξαγωγές τις προσομοίωσης οι τιμές αυτές παρουσιάζουν μικρές αποκλίσεις. Το γεγονός οφείλεται πιθανότατα στο σχετικά ελαστικό κριτήριο τερματισμού της προσομοίωσης. Ένα αυστηρότερο κριτήριο θα μπορούσε να οδηγήσει σε περισσότερα βήματα και, συνεπώς, σε ακόμα καλύτερη σύγκλιση.

```
Probability of state 0 is: 0.24665.
Probability of state 1_1 is: 0.21244.
Probability of state 1_2 is: 0.19644.
Probability of state 2 is: 0.34447.
```

Κώδικας

```
clc;
clear all;
close all;

lambda = 1;
m1 = 0.8;
m2 = 0.4;

threshold_1a = lambda/(lambda+m1);
```

```
threshold_1b = lambda/(lambda+m2);
threshold_2_first = lambda/(lambda+m1+m2);
threshold_2_second = (lambda+m1)/(lambda+m1+m2);

current_state = 0;
arrivals = zeros(1,4);
total_arrivals = 0;
maximum_state_capacity = 2;
previous_mean_clients = 0;
delay_counter = 0;
time = 0;

while 1 > 0
  time = time + 1;

  if mod(time,1000) == 0
    for i=1:1:4
      P(i) = arrivals(i)/total_arrivals;
    endfor

    delay_counter = delay_counter + 1;

    mean_clients = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);

    delay_table(delay_counter) = mean_clients;

    if abs(mean_clients - previous_mean_clients) < 0.00001
       break;
    endif
    previous_mean_clients = mean_clients;
  endif

  random_number = rand(1);

  if current_state == 0
      current_state = 1;
      arrivals(1) = arrivals(1) + 1;
      total_arrivals = total_arrivals + 1;
  elseif current_state == 1
    if random_number < threshold_1a
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
```

```
  elseif current_state == 2
    if random_number < threshold_1b
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  else
      if random_number < threshold_2_first
        arrivals(4) = arrivals(4) + 1;
        total_arrivals = total_arrivals + 1;
      elseif random_number < threshold_2_second
        current_state = 2;
      else
        current_state = 1;
      endif
   endif

endwhile

display(["Probability of state 0 is: ", num2str(P(1)), "."]);
display(["Probability of state 1_1 is: ", num2str(P(2)), "."]);
display(["Probability of state 1_2 is: ", num2str(P(3)), "."]);
display(["Probability of state 2 is: ", num2str(P(4)), "."]);
```