ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

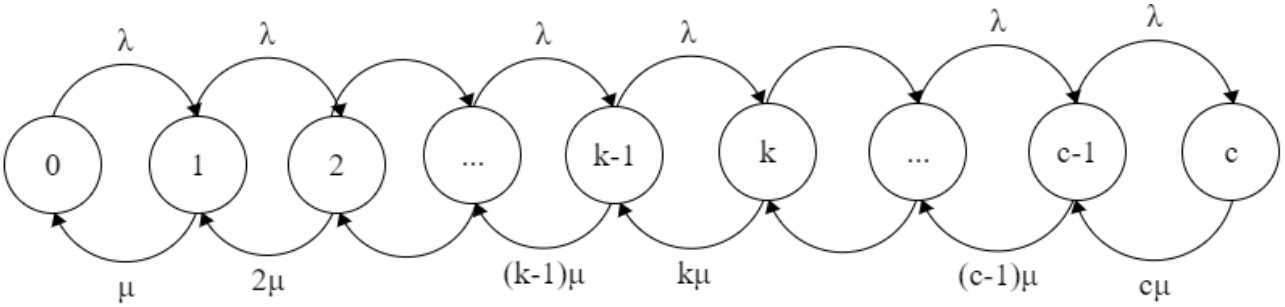
Ονοματεπώνυμο: Γκούμε Λαουρεντιάν
Α.Μ.: 031 18 014
Έτος/Εξάμηνο: 3ο/6ο

ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ (Queuing Systems)

4η Σειρά Ασκήσεων

Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου:

(1) Το ζητούμενο διάγραμμα ρυθμού μεταβάσεων για την ουρά M/M/c/c:



Από τις εξισώσεις ισορροπίας:

- $\lambda P_0 = \mu P_1$, $\lambda P_1 = (2\mu)P_2$, ..., $\lambda P_{k-1} = k\mu P_k$ => $P_k = \left(\frac{\lambda}{k\mu}\right) P_{k-1}$ => $\mathbf{P_k = \left(\frac{\rho^k}{k!}\right) P_0}$, για $k = 1, 2, \ldots, c$ και όπου $\rho = (\lambda/\mu)$

- Συνθήκη κανονικοποίησης: $P_0 + P_1 + P_2 + \ldots + P_{c-1} + P_c = 1 \Rightarrow P_0 \left(1 + \frac{\rho^1}{1!} + \frac{\rho^2}{2!} + \ldots + \frac{\rho^{(c-1)}}{(c-1)!} + \frac{\rho^c}{c!}\right) = 1 \Rightarrow P_0 = \frac{1}{\sum_{k=0}^{c} \left(\frac{\rho^k}{k!}\right)}$

- Εύκολα προκύπτει τώρα ότι $\mathbf{P_{blocking} = P_c} = \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} \left(\frac{\rho^k}{k!}\right)}$

Ο **μέσος ρυθμός απωλειών πελατών από την ουρά** είναι $\lambda P_{blocking} = \lambda P_c = \frac{\frac{\lambda \rho^c}{c!}}{\sum_{k=0}^{c} (\frac{\rho^k}{k!})}$

Υλοποιούμε την συνάρτηση **erlangb_factorial**:

```
function P_blocking = erlangb_factorial(r, c)
  numerator = power(r, c) / factorial(c);
  denominator = 0;
  for k = 0 : c
    denominator = denominator + (power(r, k) / factorial(k));
  endfor
  P_blocking = 100 * (numerator / denominator);
  printf("[Factorial] Probability of rejecting a client (ρ = %d, c = %d) is: %f%%\n", r, c, P_blocking)
endfunction
```

(2) Υλοποιούμε τη συνάρτηση **erlangb_iterative**:

```
function P_blocking = erlangb_iterative(r, c)
  B = ones(1, c + 1);                 #array containing (c+1) elements
  for i = 2 : (c + 1)
    numerator = r * B(i - 1);
    denominator = numerator + (i - 1);
    B(i) = numerator / denominator;
  endfor
  P_blocking = 100 * B(c + 1);
  printf("[Iterative] Probability of rejecting a client (ρ = %d, c = %d) is: %f%%\n", r, c, P_blocking)
endfunction
```

(3) Εκτελώντας τις 2 παραπάνω συναρτήσεις, έχουμε τα εξής αποτελέσματα:

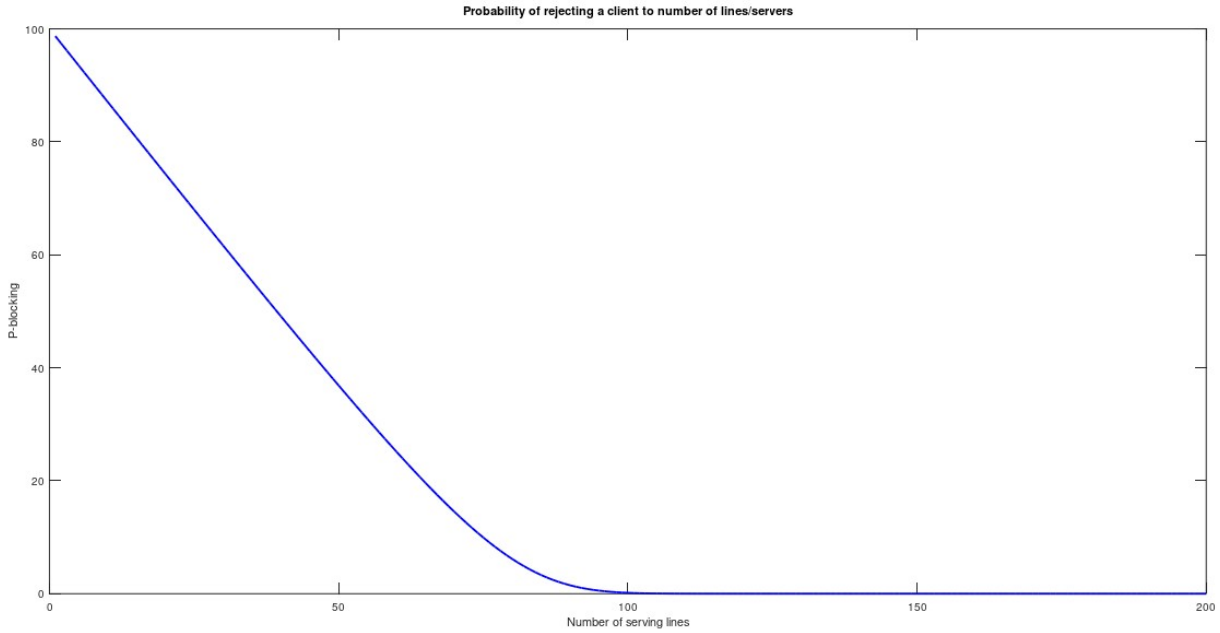
```
>> x = erlangb_factorial(1024, 1024);
[Factorial] Probability of rejecting a client (ρ = 1024, c = 1024) is: NaN%
>> x = erlangb_iterative(1024, 1024);
[Iterative] Probability of rejecting a client (ρ = 1024, c = 1024) is: 2.452426%
```

Παρατηρούμε πως η συνάρτηση erlangb_factorial έχει ως αποτέλεσμα Not_a_Number (NaN), ενώ η erlangb_iterative μας δίνει το αναμενόμενο αποτέλεσμα. Αυτό συμβαίνει, καθώς η υλοποίηση της πρώτης διαχειρίζεται αριθμούς που προκαλούν **υπερχείλιση** (μέχρι και 1024!).

(4)

(α) Για την ευκολότερη μοντελοποίηση του προβλήματος, κάνουμε την αναλογία εργαζόμενος/client, γραμμή/server. Θεωρούμε ως πρότυπο τον απαιτητικότερο χρήστη, επομένως, ισοδύναμα ο κάθε χρήστης χρησιμοποιεί τη γραμμή του 23 λεπτά ανά ώρα. Άρα η συνολική ένταση του φορτίου που καλείται να εξυπηρετηθεί από το δίκτυο της εταιρείας είναι: $\rho = 200 \frac{23}{60}$ => $\rho = \frac{230}{3}$ => **ρ = 76.67 Erlangs**.

(β) Το ζητούμενο διάγραμμα, έχει ως εξής:



Ο κώδικας που παράγει την ανωτέρω γραφική παράσταση (απαραίτητη προϋπόθεση είναι το αρχείο του παρακάτω κώδικα να βρίσκεται στον ίδιο φάκελο με το αρχείο του κώδικα erlangb_iterative.m) :

```
clc;
clear all;
close all;
#Task4_1_4_b
r = 200 * (23/60);
c = 1 : 200;
for i = 1 : 200
  P_blocking(i) = erlangb_iterative(r, i);
endfor

plot(c, P_blocking, "b", "linewidth", 1.3)
xlabel("Number of serving lines")
ylabel("P-blocking")
title("Probability of rejecting a client to number of lines/servers")
```

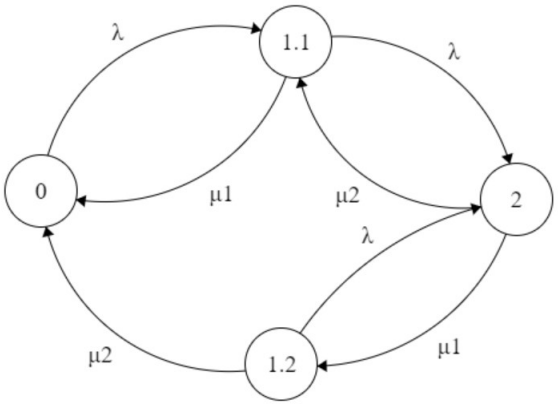
(γ) Το πλήθος των τηλεφωνικών γραμμών, που απαιτείται, έτσι ώστε η πιθανότητα απόρριψης τηλεφωνικής κλήσης να είναι μικρότερη του 1% είναι **92 γραμμές**.

```
#Task4_1_4_c
i = 1;
while(P_blocking(i) >= 1)
  i = i + 1;
endwhile
i = i - 1;    #since indexing begins from 1
printf("P-blocking is less than 1%% for %d serving lines\n", i)
```

```
P-blocking is less than 1% for 92 serving lines
```

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές:

(1) Το **διάγραμμα των ρυθμών μεταβάσεων** του συστήματος παρουσιάζεται παρακάτω:



Για τις **εργοδικές πιθανότητες** έχουμε:

- $\lambda P_0 = \mu_1 P_{1.1} + \mu_2 P_{1.2}$
- $(\lambda + \mu_1) P_{1.1} = \lambda P_0 + \mu_2 P_2$
- $(\lambda + \mu_2) P_{1.2} = \mu_1 P_2$
- $\lambda(P_{1.1} + P_{1.2}) = (\mu_1 + \mu_2) P_2$

Χρησιμοποιούμε τα δεδομένα: $\lambda = 1$ πελάτης/sec, $\mu_1 = 0.8$ πελάτες/sec και $\mu_2 = 0.4$ πελάτες/sec, οπότε και προκύπτει:

- $P_0 = 0.8(P_{1.1}) + 0.4(P_{1.2})$
- $1.8(P_{1.1}) = P_0 + 0.4(P_2)$
- $1.4(P_{1.2}) = 0.8(P_2)$
- $(P_{1.1} + P_{1.2}) = 1.2(P_2)$

Κάνοντας τις κατάλληλες πράξεις στο παραπάνω σύστημα και λαμβάνοντας υπόψιν την συνθήκη κανονικοποίησης: $P_0 + P_{1.1} + P_{1.2} + P_2 = 1$, προκύπτουν: $\mathbf{P_0 = 24.06\%}$, $\mathbf{P_{1.1} = 21.70\%}$, $\mathbf{P_{1.2} = 19.72\%}$, $\mathbf{P_2 = 34.51\%}$. Η πιθανότητα απόρριψης ενός πελάτη από το σύστημα είναι: $\mathbf{P_{blocking} = P_2 = 34.51\%}$. Ο **μέσος αριθμός πελατών** στο σύστημα είναι: $0 * P_0 + 1 * P_{1.1} + 1 * P_{1.2} + 2 * P_2 =$ **1.1044.**

(2) Ορίζουμε τα παρακάτω **όρια**:

```
10 threshold_1a = lambda/(lambda + m1);
11 threshold_1b = lambda/(lambda + m2);
12 threshold_2_first = lambda/(lambda + m1 + m2);
13 threshold_2_second = (m1 + lambda)/(lambda + m1 + m2);
```

Οπότε και παίρνουμε τις εξής εξόδους, οι οποίες και συνάδουν με τα θεωρητικά υπολογιζόμενα αποτελέσματα (στην πραγματικότητα παρατηρούνται πολύ μικρές αποκλίσεις, οι οποίες οφείλονται στην ακρίβεια του κριτηρίου σύγκλισης που επιλέχθηκε):

```
Mean number of clients is: 1.090949
Ergodic Probability P(0) = 25.138137%
Ergodic Probability P(1) = 21.369765%
Ergodic Probability P(2) = 19.259034%
Ergodic Probability P(3)/Rejection = 34.233064%
```

Αναφορικά με τα κριτήρια σύγκλισης, υπάρχει 1 και αφορά τον μέσο αριθμό πελατών στο σύστημα. Συγκεκριμένα, θεωρούμε πως η προσομοίωση **συγκλίνει όταν ο πρώην με τον τωρινό μέσο αριθμό πελατών στο σύστημα διαφέρουν λιγότερο από 0.00001**. Η σύγκριση αυτή γίνεται ανά 1000 επαναλήψεις, όπως και φαίνεται παρακάτω:

```
26  if mod(time,1000) == 0
27    for i=1:1:4
28      P(i) = arrivals(i)/total_arrivals;
29    endfor
30
31    delay_counter = delay_counter + 1;
32
33    mean_clients = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);
34
35    delay_table(delay_counter) = mean_clients;
36
37    if abs(mean_clients - previous_mean_clients) < 0.00001
38       break;
39    endif
40    previous_mean_clients = mean_clients;
41  endif
```