Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Συστήματα Αναμονής

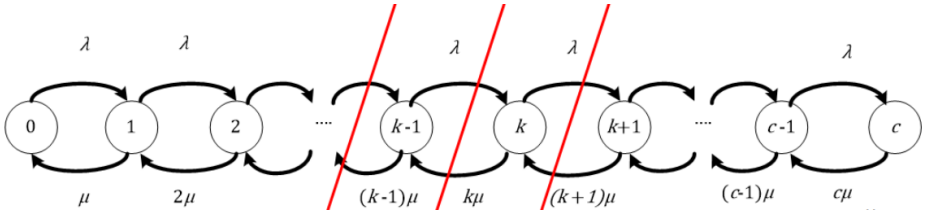
4η ομάδα ασκήσεων

Ελευθερία Αρκαδοπούλου

el19442

Ανάλυση και σχεδιασμός τηλεφωνικού κέντρου

1) Το διάγραμμα ρυθμού μεταβάσεων του συστήματος M/M/c/c:



Για να υπολογίσουμε την πιθανότητα Pblocking, παίρνουμε τις εξισώσεις ισορροπίας:

$$P_k = \left[\frac{\lambda}{\mu * k}\right] * P_{k-1} = \left(\frac{\rho^k}{k!}\right) * P_0$$

$$\sum P_k = 1$$

Προκύπτει ότι

$$P_0 = \frac{1}{\sum \frac{\rho^k}{k!}}$$

$$P_{blocking} = B(\rho, c) = \frac{\frac{\rho^c}{c!}}{\sum \frac{\rho^k}{k!}}$$

Ο μέσος ρυθμός απωλειών πελατών από την ουρά είναι ίσος με λ*Pblocking. Υλοποιούμε τη συνάρτηση erlangb_factorial που δέχεται ως ορίσματα την ένταση του φορτίου ρ και τον αριθμό εξυπηρετητών c και υπολογίζει με βάση τον πάνω τύπο την πιθανότητα απόρριψης του πελάτη από το σύστημα B(ρ,c).

```
function Pblocking = erlangb_factorial(r,c)
  sum=0;
  for k=0:c
    sum=sum+(r^k/factorial(k));
  endfor
  Pblocking = (r^c/factorial(c))/sum;
  display(Pblocking);
endfunction
```

Ενδεικτικά για παραμέτρους 3,5:

```
>> erlangb_factorial(3,5)
ans =  0.11005
```

2) Υλοποιούμε με επαναληπτικό τρόπο τη συνάρτηση erlangb_iterative, που υπολογίζει την πιθανότητα απόρριψης πελάτη από το σύστημα με βάση τον τύπο:

$$B(\rho,0) = 1$$

$$B(\rho,n) = \frac{\rho B(\rho,n-1)}{\rho B(\rho,n-1) + n}, n = 1,2,\ldots,c$$

```
function Pblocking = erlangb_iterative(r,c)
  Pblocking = 1;
  for n=1:c
    Pblocking = (r*Pblocking)/(r*Pblocking+n);
  endfor
endfunction
```

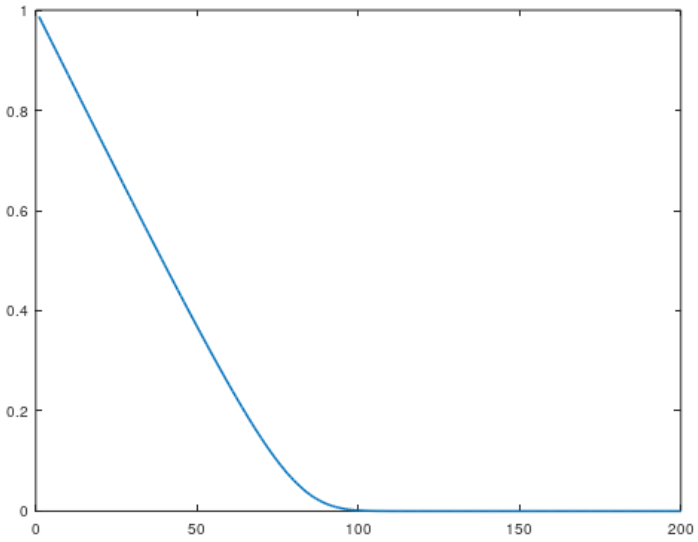
```
>> erlangb_iterative(3,5)
Pblocking =  0.11005
ans =  0.11005
```

3) Συγκρίνουμε τις εξόδους των erlangb_factorial(1024,1024) και erlangb_iterative(1024,1024). Παρατηρούμε ότι από τις δύο συναρτήσεις, η erlangb_factorial δεν επιστρέφει απάντηση (NaN). Αυτό δικαιολογείται από το γεγονός ότι η συνάρτηση δεν έχει την υπολογιστική ισχύ για να υπολογίσει πολύ μεγάλους αριθμούς όπως το 1024!.

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

4) Για τον πιο απαιτητικό πελάτη, ρ=200*23/60=76.67 Erlangs.

Το διάγραμμα πιθανότητας απόρριψης πελάτη από το σύστημα, με χρήση της erlangb_iterative:



Για να προσδιορίσουμε τον κατάλληλο αριθμό τηλεφωνικών γραμμών ώστε η πιθανότητα απόρριψης της τηλεφωνικής κλήσης να είναι μικρότερη του 1%, χρησιμοποιούμε το flag minimum. Προκύπτει ότι ο κατάλληλος αριθμός είναι ίσος με 93.

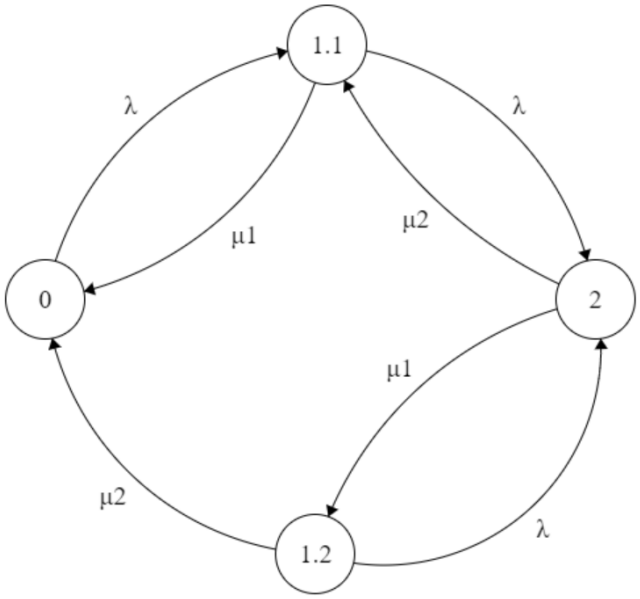
```
Pblocking < 1% for
 93
```

Ο κώδικας:

```
clear all;
close all;
clc;

function Pblocking = erlangb_iterative(r,c)
  Pblocking = 1;
  for n=1:c
    Pblocking = (r*Pblocking)/(r*Pblocking+n);
  endfor
endfunction

r=200*23/60;
c=200;

minimum = true;
for i=1:c
  Pblocking(i)=erlangb_iterative(r,i);
  if Pblocking(i) < 0.01 && minimum
    disp("Pblocking < 1% for ")
    disp(i)
    minimum = false;
  endif
endfor

plot(1:200,Pblocking,"linewidth",1.5);
```

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές

1) Το διάγραμμα ρυθμών μεταβάσεων του συστήματος από δύο εξυπηρετητές χωρίς δυνατότητα αναμονής, με τον πελάτη να δρομολογείται πάντα στον εξυπηρετητή 1 όταν είναι και οι δύο διαθέσιμοι:



Για να υπολογίσουμε εργοδικές πιθανότητες και πιθανότητα απόρριψης:

$$\lambda * P0 = \mu1 * P11 + \mu2 * P12 => \ P0 = 0.8 * P11 + 0.4 * P12$$

$$\mu1 * P12 + \lambda * P11 = \mu2 * P2 + \lambda * P0$$

$$\mu2 * P2 + \mu1 * P2 = \lambda * P11 + \lambda * P12$$

$$\lambda * P12 + \mu2 * P12 = \ \mu1 * P2$$

Προκύπτει ότι:

$$P2 = \frac{5}{6} * (P11 + P12), \quad P11 = \frac{5}{9} * P0 + \frac{2}{9} * P2.$$

$$P12 = \frac{4}{7} * P2, \qquad P0 + P11 + P12 + P2 = 1$$

Όπου $\mathrm{P11} = 0.8594 * \mathrm{P0}, \qquad \mathrm{P12} = 0.7813 * \mathrm{P0}, \qquad \mathrm{P2} = 1.367 * \mathrm{P0}$

Άρα, επιλύοντας τις εξισώσεις:

$$P0 = 0.243, P11 = 0.214, P12 = 0.195, P2 = 0.341$$

$$Pblocking = P2 = 0.341$$

Εξίσωση για τον μέσο αριθμό πελατών:

$$E[n(t)] = \sum k * Pk = 1.09 \text{ πελάτες}$$

2) Τροποποιούμε το αρχείο demo4.m, προσθέτοντας τα παρακάτω:

```
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);
```

Κριτήριο σύγκλισης της προσομοίωσης αποτελεί ο μέσος αριθμός πελατών να διαφέρει από τον προηγούμενο λιγότερο από 0.00001.

```
if abs(mean_clients - previous_mean_clients) < 0.00001
```

Οι πιθανότητες που υπολογίζονται:

```
0.25305
0.21329
0.19146
0.34220
```

Ο ολοκληρωμένος 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(P(1));
display(P(2));
display(P(3));
display(P(4));
```