Санкт-Петербургский Политехнический Университет Петра Великого

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №4

Курс: «Методы оптимизации и принятия решений»

Тема: «Оптимизация сетей систем массового обслуживания»

Выполнил студент:

Волкова Мария Дмитриевна
Группа: 13541/2

Проверил:

Сиднев Александр Георгиевич

Санкт-Петербург
2019 г.

Содержание

Лабораторная работа №4

1.1 Индивидуальное задание

Найти

$$max\lambda = e_1 G_M(N-1)/G_M(N)$$

при ограниении

$$S = \sum_{i=1}^{M} c_i \mu_i^{a_i} = S^*, \quad \mu > 0$$

Дано

$$\{S^*, M, N, \{\mathbf{p}_{ij}\}_{i=\overrightarrow{0,M}, j=\overrightarrow{0,M}}, \vec{c} = (c_1, c_2, ..., c_M), \vec{a} = (a_1, a_2, ..., a_M)\}$$

где M - число узлов;

N - число заявок в сети;

$\vec{c} = (c_1, c_2, ..., c_M)$ - вектор, определяющи число каналов в узле;

$\vec{a} = (a_1, a_2, ..., a_M)$ - вектор, определяющий коэффициенты важности узлов. Коэффициенты важности узлов ССМО входят в формулу расчета ее стоимости.

S^* - стоимость сети.

Параметры

$\pi = \{\mathbf{p}_{ij}\}_{i=\overrightarrow{0,M}, j=\overrightarrow{0,M}}$				N	S	$\vec{c}$	$\vec{a}$
0	0.3	0.3	0.4	5	5	1 1 1 1	1 1 1 1
0.4	0	0.2	0.4				
0.1	0.2	0	0.7				
0.2	0.7	0.1	0				

1.2 Ход работы

Вычисление вероятностей w_i

Для этого решаем систему уравнений:

$$w_i = \sum_{i=1}^{M} w_i * p_{ij}, \qquad j = i..M$$

$$\sum_{i=1}^{M} w_i = 1$$

которая в данном случае принимает вид

$$\begin{cases} w_1 = 0w_1 + 0.3w_2 + 0.3w_3 + 0.4w_4 \\ w_2 = 0.4w_1 + 0w_2 + 0.2w_3 + 0.4w_4 \\ w_3 = 0.1w_1 + 0.2w_2 + 0w_3 + 0.7w_4 \\ w_4 = 0.2w_1 + 0.7w_2 + 0.1w_3 + 0w_4 \end{cases}$$

Листинг 1.1: Скрипт для нахождения вероятностей

```
w = fsolve(@wfun,[1;0;0;0]);
    function F = wfun(w)
        for j = 1:M
            sum_t = 0;
            for i = 1:M
                sum_t = sum_t + w(i)*p(i,j);
            end
            F(j) = sum_t - w(j);
        end
        F = [F(1); F(2);F(3);F(4); sum(w) - 1];
    end
options = optimset('Display','iter');
[w,fval] = fsolve(@wfun,[1;0;0;0],options)
```

После 25 обращений к функциям нули найдены:

Iteration	Func-count	Residual	First-Order optimality	Lambda	Norm of step
0	5	1.34	1.34	0.01	-
1	10	5.49043e-05	0.00789	0.001	0.919789
2	15	2.36409e-11	5e-06	0.0001	0.00599099
3	20	1.04197e-19	3.27e-10	1e-05	3.95646e-06
4	25	4.75859e-30	2.18e-15	1e-06	2.64023e-10

Получаем:

$$\begin{cases} w_1 = 0.2064 \\ w_2 = 0.3170 \\ w_3 = 0.1572 \\ w_4 = 0.3194 \end{cases}$$

Значение целевой функции:

```
fval =

   1.0e-14 *

   -0.1804
    0.0111
    0.0999
    0.0666
   -0.0222
```

Расчет норминирующей константы

$$G_r(k) = \sum_{l=0}^{k} Z_r(l) G_{r-1}(k-l)$$

$$Z_i(n_i) = \frac{w_i^{n_i}}{\prod\limits_{j=1}^{n_i} u_i(j)}, \qquad u_i(j) = \begin{cases} ju_i & j < m_i \\ m_i u_i & j >= m_i \end{cases}$$

Листинг 1.2: Скрипт для расчета норминирующей константы

```
for r = 2:1:M
    for k = 1:1:N
        sum = 0;
        for l = 0:1:k
            sum = sum + z(r, l + 1)*G(r - 1, k - l + 1);
        end
        G(r, k + 1) = sum;
    end
```

```
end
```

Нахождение интенсивности на выходе 1 узла

Находим интенсивность по следующей формуле:

$$\lambda_i(N)\frac{w_i G_M(N-1)}{G_M(N)}$$

Листинг 1.3: Скрипт для нахождения интенсивности

```
lambda = G(M, N - 1)/G(M,N);
```

Функция оптимизации

Будем оптимизировать нашу функцию с помощью функции fmincon. fmincon находит минимум для скалярной функции нескольких переменных с ограничениями начиная с начального приближения.

Листинг 1.4: fmincon

```
fun = @(x)(-findlambda(w,x));
[my_u,fval] = fmincon(fun,w,[],[],[],[],[0;0;0;0],[],@limitation,optimset('
    Display','iter'))
```

со следующим нелинейным ограничением:

Листинг 1.5: Ограничения

```
function [ctmp , ceqtmp] = limitation(x)
ctmp = 0;
    for i = 1:M
        ctmp = ctmp + c(i)*x(i)^a(i);
    end
    ctmp = ctmp  - S;
    ceqtmp = [];
end
```

Результаты

Посмотрим на работу отпимизации:

Iter	Func-count	f(x)	Feasibility	First-order optimality	Norm of step
0	5	-5.333333e-01	0.000e+00	9.110e-01	-
1	10	-1.234964e+00	0.000e+00	5.957e-01	6.427e-01
2	15	-2.702824e+00	0.000e+00	5.358e-01	1.391e+00
3	21	-2.507462e+00	0.000e+00	2.767e-01	1.882e-01
4	27	-2.604093e+00	0.000e+00	1.556e-01	8.964e-02
5	32	-2.629467e+00	0.000e+00	1.003e-01	3.378e-02
6	37	-2.685129e+00	0.000e+00	2.694e-02	5.611e-02
7	42	-2.710478e+00	0.000e+00	9.348e-04	2.523e-02
8	47	-2.711170e+00	0.000e+00	2.129e-04	8.948e-04
9	52	-2.711170e+00	0.000e+00	2.000e-04	1.798e-04
10	57	-2.711330e+00	0.000e+00	4.004e-05	1.572e-04
11	62	-2.711370e+00	0.000e+00	4.023e-07	4.019e-05

После 11 итераций (62 обращений к функции) оптимизация завершается успешно.

Листинг 1.6: Результаты

```
my_u =
    1.0682
    1.4485
    0.9602
    1.5231
```

```
6
7 fval = -2.7114
```

1.3 Вывод

В данной работе была роведена оптимизация СМО. В результате проведенной работы был написан скрипт по оптимизации одноканальной, однородной (все заявки однотипные), замкнутой (с фиксированым числом заявок) сети СМО, который также подходит для многоканальной замкнутой сети систем массового обслуживания. Для увеличения производительности данного алгоритма можно подключить пакет MATLAB Parallel Computing Toolbox, который использует MPI для эффективного распараллеливания.

Листинг 1.7: Полный текст скрипта

```
function test
clc;
% nachal'nie znacheniya
S = 5;
N = 5;    % число заявок в сети
M = 4;    % число узлов

p = [0   0.3  0.3  0.4;...
    0.4  0    0.2  0.4;...
    0.1  0.2  0    0.7;...
    0.2  0.7  0.1  0  ];

m = [1;1;1;1];       % число каналов в iом- узле
c = [1 1 1 1];       % стоимостные коэффициенты
a = [1;1;1;1];       % коэффициенты нелинейности

% nakhodeniye veroyatnostey w
w = fsolve(@wfun,[1;0;0;0]);
    function F = wfun(w)
        for j = 1:M
            sum_t = 0;
            for i = 1:M
                sum_t = sum_t + w(i)*p(i,j);
            end
            F(j) = sum_t - w(j);
        end
        F = [F(1); F(2);F(3);F(4); sum(w) - 1];
    end

options = optimset('Display','iter');
[w,fval] = fsolve(@wfun,[1;0;0;0],options)

% find lambda
function [ lambda ] = findlambda(w,u)
for i = 1:1:M
   for n = 0:1:N
       my = 1;
       for j = 1:1:n
           if (j >= m(i))
               my = my*m(i)*u(i);
           else my = my*j*u(i);
           end
       end
       z(i,n+1) = (w(i)^n)/my;
   end
end
G(1, :) = z(1, :);

for r = 2:1:M
    for k = 1:1:N
        sum = 0;
        for l = 0:1:k
            sum = sum + z(r, l + 1)*G(r - 1, k - l + 1);
        end
        G(r, k + 1) = sum;
    end
end

lambda = 0;
```

```
lambda = G(M, N - 1)/G(M,N);
end

% nelineynoye ogranicheniye
function [ctmp , ceqtmp] = limitation(x)
ctmp = 0;
    for i = 1:M
        ctmp = ctmp + c(i)*x(i)^a(i);
    end
    ctmp = ctmp  - S;
    ceqtmp = [];
end

% optimization
fun = @(x)(-findlambda(w,x));
[my_u,fval] = fmincon(fun,w,[],[],[],[],[0;0;0;0],[],@limitation,optimset('
   Display','iter'))
end
```