Санкт-Петербургский Политехнический Университет Петра Великого

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №3

Курс: «Методы оптимизации и принятия решений»

Тема: «Оптимизация сетей систем массового обслуживания»

Выполнил студент:

Бояркин Никита Сергеевич
Группа: 13541/3

Проверил:

Сиднев Александр Георгиевич

Санкт-Петербург
2018 г.

Содержание

Лабораторная работа №3

1.1 Индивидуальное задание

Вариант 030

Задача №2.

Найти

$$min\ S = \sum_{i=1}^{M} c_i \mu_i^{a_i}$$

при ограничении

$$\lambda = e_1 G_M(N-1)/G_M(N) = \lambda^*.$$

Дано:

$$\left\{ \boldsymbol{\lambda}^*, \boldsymbol{M}, N, \quad \boldsymbol{\pi} = \{ p_{ij} \}_{\substack{i=\overline{1,M} \\ j=\overline{1,M}}}, \vec{\boldsymbol{c}} = (c_1, c_2, \ . \ . \ . \ , \ c_M), \vec{\boldsymbol{\alpha}} = (\alpha_1, \alpha_2, \ . \ . \ . \ , \ \alpha_M) \right\}$$

где

$\boldsymbol{M}$ — число узлов;

$\boldsymbol{N}$ — число заявок в сети;

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

$\vec{\boldsymbol{\alpha}} = (\alpha_1, \alpha_2, \ . \ . \ . \ , \ \alpha_M)$ — вектор, определяющий коэффициенты важности узлов. Коэффициенты важности α_i узлов ССМО входят в формулу расчета её стоимости.

$\boldsymbol{\lambda}^*$ — заданная интенсивность потока в 1-м узле: $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}^*$

$$\mathbf{1}_k = \left(\underset{1}{\mathbf{1}}, \underset{2}{\mathbf{1}}, \dots, \underset{k}{\mathbf{1}}\right)$$

Примечание. Используется следующее обозначение:

030		0	0,1	06	0,3		5	-	3	2_4	$\mathbf{1}_4$
		0,7	0	0,2	0,1						
		0,9	0	0	0,1						
		0,3	0,3	0,4	0						

1.2 Ход работы

1.2.1 Методика решения простыми итерациями

Задача оптимизации замкнутой однородной сети сводится к решению системы нелинейных уравнений:

$$\mu_1 = \lambda^*/U_1(N);$$
$$\mu_1^{a_i} = \frac{c_1 a_1}{c_i a_i}\mu_1^{a_1}\frac{L_i(N) - L_i(N-1)}{L_1(N) - L_1(N-1)}, \quad i = \overline{2, M}.$$

Для многоканальной сети МО используется итерационная процедура расчета среднего числа заявок в очереди. Процедура инициализируется следующими начальными условиями:

$$P_i(0,0) = 1, \quad i = \overline{1, M}\ ; \ r = 1$$

Первый шаг процедуры:

$$\overline{t_i}(r) = \sum_{n=1}^{r} \frac{n}{\mu_i(n)} P_i(n-1, r-1), \quad i = \overline{1, M}$$

$$\mu_i(n) = \begin{cases} n\mu_i, & n < m_i \\ m_i\mu_i, & n \ge m_i \end{cases}$$ где m_i — число каналов в i-м узле.

Второй шаг процедуры:

$$\lambda_1(r) = \frac{r}{\sum_{j=1}^{M} \frac{\omega_j}{\omega_1}\overline{t_j}(r)}$$

Третий шаг процедуры:

$$\begin{cases} P_i(n,r) = \dfrac{\omega_i \lambda_1(r)}{\omega_1 \mu_i(n)} P_i(n-1,\ r-1),\ n = \overline{1, r}, \\ P_i(0,r) = 1 - \sum_{n=1}^{r} P_i(n,r), \end{cases}$$

После этого значение r увеличивается на единицу до достижения N. В результате работы процедуры формируются значения следующих характеристик:

$$\overline{t_i}(N),\ i = \overline{1, M};\quad \{P_i(n,N)\},\ i = \overline{1, M};\ n = \overline{0, N}.$$

1.2.2 Решение задачи методом простых итераций

Разработаем скрипт для расчета результирующего вектора μ в среде MATLAB:

```
clear all;
close all;
clc;
format long g;

p =    [0 0.1 0.6 0.3;
        0.7 0 0.2 0.1;
```

```
        0.9 0 0 0.1;
        0.3 0.3 0.4 0];

M = 4;
N = 5;

lam_1_ = 3;

c = [2 2 2 2];
a = [1 1 1 1];

u = [1 1 1 1];

%% Initialize w

% A = p' - diag(ones(1, M));
% A(4, :) = [1; 1; 1; 1];
% b = [0; 0; 0; 1];
% w = inv(A) * b;
% w = w';

w = zeros(1,M);
w(1) = 1;
for i = 1 : M
     for j = 1 : M
        w(i) = w(i) + w(j) * p(j, i);
    end
end

w

%% Initialize u

for i = 1 : M
    u(i) = w(i) / (c(i) * w(1)) * lam_1_;
end

u

for iteration = 1 : 5
    %% Find L(N)
    [L_N, lam_1_N] = fl1(u, w, c, M, N);
    L_N_1 = fl1(u, w, c, M, N - 1);

    %% Find u1
    u(1) = lam_1_ / lam_1_N * u(1);

    %% Find u
    for i = 2 : M
       u(i) = u(1)  * (L_N(i) - L_N_1(i)) / (L_N(1) - L_N_1(1)) * (c(1) * a(1)) / (c(i) * a(i));
    end

    fprintf('Iteration #%d\nSumm u(i) = %f\nu(i) = [%f, %f, %f, %f]\n\n', iteration, sum(u), u(1), u(2), u(3), u(4));
end
```

Итеративная процедура:

```
function [L, lumbda1] = fl1(u, w, c, M, N)
    P = zeros(M, N + 1, N + 1);
    t = zeros(1, M);

    P(:, 1, 1) = 1;
    for r = 1 : N
        %% Step 1

```

```
        for i = 1 : M
            t(i) = 0;
            for n = 1 : r
                t(i) = t(i) + n / (min(n, c(i)) * u(i)) * P(i, n, r);
            end
        end

        %% Step 2

        temp = 0;
        for i = 1 : M
            temp = temp + w(i) * t(i) / w(1);
        end

        lumbda1 = r / temp;

        %% Step 3

        for i = 1 : M
            for n = 1 : r
                P(i, n + 1, r + 1) = w(i) / w(1) * lumbda1 / (min(n, c(i)) * u(i)) * P(i, n, r);
            end

            P(i, 1, r + 1) = 1;
            for n = 1 : r
                P(i, 1, r + 1) = P(i, 1, r + 1) - P(i, n + 1, r + 1);
            end
        end
    end

    L = t;
end
```

Результат расчета вектора μ:

```
Iteration #1
Summ u(i) = 26.030301
u(i) = [2.812500, 12.738971, 3.492944, 6.985887]

Iteration #2
Summ u(i) = 2.131577
u(i) = [1.651439, 0.000621, 0.466883, 0.012634]

Iteration #3
Summ u(i) = 4677058315956.762700
u(i) = [881.151124, 4676727098598.208000, 25284.797051, 331191192.606743]

Iteration #4
Summ u(i) = 1.500003
u(i) = [1.500001, 0.000000, 0.000002, 0.000000]

Iteration #5
Summ u(i) = NaN
u(i) = [NaN, NaN, NaN, NaN]
```

Алгоритм ожидаемо не сошелся, так как метод простых итераций не рекомендуется к использованию из-за ненадежности в плане сходимости.

1.2.3 Методика решения через через нормирующую константу методом Ньютона

Задача оптимизации замкнутой однородной сети сводится к решению системы нелинейных уравнений:

$$\mu_1 = \lambda^*/U_1(N);$$
$$\mu_1^{a_i} = \frac{c_1 a_1}{c_i a_i}\mu_1^{a_1}\frac{L_i(N) - L_i(N-1)}{L_1(N) - L_1(N-1)}, \quad i = \overline{2, M}.$$

Тогда интенсивность на выходе i -го узла однородной замкнутой сети СМО:

$$\lambda_i(N) = \omega_i G_M(N-1)/G_M(N)$$

Среднее число заявок в граничном узле однородной замкнутой сети СМО:

$$\overline{n_M}(N) = \frac{\sum_{n=1}^{N} n Z_M(n) G_{M-1}(N-n)}{G_M(N)}$$

Методика расчета нормирующей константы:

$$G_1(k) = Z_1(k) = \frac{\omega_i^k}{\prod_{j=1}^{k}\mu_1(j)}, \quad k = \overline{0, N};$$

$$G_r(0) = 1, \quad r = \overline{1, M};$$

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

Алгоритм Ньютона для решения системы нелинейных уравнений (W – матрица якоби для системы уравнений F):

$$x^{(k+1)} = x^{(k)} - W^{-1}(x^{(k)}) \cdot F(x^{(k)}), k = 0, 1, 2, \ldots$$

1.2.4 Решение задачи через нормирующую константу методом Ньютона

Решение задачи через нормирующую константу методом Ньютона с начальным приближением $u = [10, 10, 10, 10]$ и значением ошибки $e = 1e^{-6}$:

```
clear all;
close all;
clc;
format long g;

% delete(gcp);
% distcomp.feature('LocalUseMpiexec', false);
% parpool();

p =    [0 0.1 0.6 0.3;
        0.7 0 0.2 0.1;
```

```
        0.9 0 0 0.1;
        0.3 0.3 0.4 0];

M = 4;
N = 5;

lam_1_ = 3;

c = [2 2 2 2];
a = [1 1 1 1];

%% Initialize w

A = p' - diag(ones(1, M));
A(4, :) = [1; 1; 1; 1];
b = [0; 0; 0; 1];
w = inv(A) * b;
w = (1 / w(1)) * w';

%% Error

e = 1e-06;

%% First approximation

u0 = zeros(1, M);
for i = 1 : M
    u0(i) = w(i) / (c(i) * w(1)) * lam_1_;
end

fprintf('Start Conditions\ne = %f\nw = [%.6f %.6f %.6f %.6f]\nu0 = [%.6f %.6f %.6f %.6f]\n\n', e, w, u0);

%% Syms u

u = sym('u', [1 M]);

%% Find characteristics

G = gl3(u, w, c, M, N);
[L, Lam] = pl3(u, w, c, M, N, G);
[pL, pLam] = pl3(u, w, c, M, N - 1, G);

% [L, Lam] = fl3(u, w, c, M, N);
% [pL, pLam] = fl3(u, w, c, M, N - 1);

Function = sym(zeros(1, M));
Function(1) = simplify(lam_1_ / Lam(1) * u(1));

for i = 2 : M
    % Function(i) = simplify(Function(1) * (L(i) - pL(i)) / (L(1) - pL(1)) * (c(1) * a(1)) / (c(i) * a(i)));
    Function(i) = simplify(u(1) * (L(i) - pL(i)) / (L(1) - pL(1)) * (c(1) * a(1)) / (c(i) * a(i)));
end

%% Newton method with jacobian matrix

% uCurrent = u0;
uCurrent = [10 10 10 10];
uPrevious = zeros(1, M);

jaco = jacobian(Function - u);

index = 0;
condition = true;
```

```
while condition
    resf = zeros(M, 1);
    % parfor i = 1 : M
    for i = 1 : M
        resf(i) = subs(Function(i) - u(i), u, uCurrent);
    end

    rU = double(uCurrent);
    fprintf('%d | u = [%.6f %.6f %.6f %.6f] | F(u) = [%.6f %.6f %.6f %.6f] | S = %.6f\n', index, double(uCurrent), double(resf), c * rU');

    uPrevious = uCurrent;

    resjaco = zeros(M, M);
    % parfor i = 1 : M
    for i = 1 : M
        for j = 1 : M
            resjaco(i, j) = subs(jaco(i, j), u, uPrevious);
        end
    end

    resf = zeros(M, 1);
    % parfor i = 1 : M
    for i = 1 : M
        resf(i) = subs(Function(i) - u(i), u, uPrevious);
    end

    uCurrent = uPrevious - (resjaco \ resf)';

    condition = false;
    for i = 1 : M
        condition = condition || abs(uCurrent(i) - uPrevious(i)) > e;
    end

    index = index + 1;

    if (~condition)
        rU = double(uCurrent);
        fprintf('%d | u = [%.6f %.6f %.6f %.6f] | F(u) = [%.6f %.6f %.6f %.6f] | S = %.6f\n', index, double(uCurrent), double(resf), c * rU');
    end
end

resf = zeros(M, 1);
% parfor i = 1 : M
for i = 1 : M
    resf(i) = subs(Function(i) - u(i), u, uCurrent);
end

rU = double(uCurrent);
rL = double(subs(L, u, uCurrent));
rLam = double(subs(Lam, u, uCurrent));
rS = c * rU';
fprintf('\nResult\nu = [%.6f %.6f %.6f %.6f]\nF(u) = [%.6f %.6f %.6f %.6f]\nL(N) = [%.6f %.6f %.6f %.6f]\nlam(N) = [%.6f %.6f %.6f %.6f]\nS = %.6f\n', rU, double(resf), rL, rLam, rS);
```

Расчет нормирующей константы:

```
function [G] = gl3(u, w, c, M, N)
    G = sym(zeros(M, N + 1));
    G(:, 1) = 1;

    for k = 0 : N
        %% Find G(1, k)

        tempMul = 1;
```

```
        for j = 1 : k
            tempMul = tempMul * min(j, c(1)) * u(1);
        end

        G(1, k + 1) = (w(1) ^ k) / tempMul;
    end

    for r = 2 : M
        for k = 1 : N
            %% Find G(r, k)

            tempSum = 0;
            for h = 0 : k
                Z = 0;
                if (h == 0)
                    Z = 1;
                else
                    %% Find Z(r, h)
                    tempMul = 1;
                    for j = 1 : h
                        tempMul = tempMul * min(j, c(r)) * u(r);
                    end

                    Z = (w(r) ^ h) / tempMul;
                end

                tempSum = tempSum + Z * G(r - 1, k - h + 1);
            end

            G(r, k + 1) = tempSum;
        end
    end

    G = simplify(G);
end
```

Расчет основных характеристик СМО:

```
function [L, Lam] = pl3(u, w, c, M, N, G)
    L = sym(zeros(1, M));
    Lam = sym(zeros(1, M));

    for i = 1 : M
        tempSum = 0;
        for n = 1 : N
            Z = 0;
            if (n == 0)
                Z = 1;
            else
                %% Find Z(i, n)
                tempMul = 1;
                for j = 1 : n
                    tempMul = tempMul * min(j, c(i)) * u(i);
                end

                Z = (w(i) ^ n) / tempMul;
            end

            tempSum = tempSum + n * Z * G(M - 1, N - n + 1);
        end

        L(i) = tempSum / G(M, N + 1);
        Lam(i) = w(i) * G(M, N - 1 + 1) / G(M, N + 1);
    end

    L = simplify(L);
    Lam = simplify(Lam);
```

```
end
```

Результат решения задачи методом Ньютона с начальным приближением $u = [10, 10, 10, 10]$ и значением ошибки $e = 1e^{-6}$:

```
Start Conditions
e = 0.000001
w = [1.000000 0.220779 0.805195 0.402597]
u0 = [1.500000 0.331169 1.207792 0.603896]

0 | u = [10.000000 10.000000 10.000000 10.000000] | F(u) = [-8.115572 -9.883869 -5.310920 -9.452506] | S = 80.000000
1 | u = [1.950351 0.066599 1.553885 0.345572] | F(u) = [8.060789 3195.369625 0.505314 52.660296] | S = 7.832812
2 | u = [6.139607 0.235956 5.078620 1.243394] | F(u) = [2.780702 4560.949367 0.429830 75.412690] | S = 25.395155
3 | u = [7.236115 0.323986 6.115423 1.745173] | F(u) = [0.461988 2370.124917 0.023511 39.297704] | S = 30.841395
4 | u = [6.386103 0.343664 5.434348 1.897411] | F(u) = [0.083022 901.425404 -0.046716 14.756548] | S = 28.123053
5 | u = [5.297395 0.352083 4.512774 1.967574] | F(u) = [0.039631 316.907201 -0.023056 4.768651] | S = 24.259653
6 | u = [4.327837 0.363774 3.684367 1.950805] | F(u) = [0.048224 108.424858 -0.004624 1.171350] | S = 20.653566
7 | u = [3.567383 0.386452 3.032279 1.777383] | F(u) = [0.071425 37.202756 0.002715 0.136602] | S = 17.526996
8 | u = [3.045719 0.428871 2.583950 1.532460] | F(u) = [0.090938 13.122322 0.003303 0.007226] | S = 15.182001
9 | u = [2.735042 0.497981 2.316126 1.364156] | F(u) = [0.082449 4.723376 0.003070 0.008377] | S = 13.826611
10 | u = [2.569108 0.591590 2.172423 1.272798] | F(u) = [0.055265 1.634580 0.002543 0.006852] | S = 13.211837
11 | u = [2.493841 0.688836 2.106835 1.230292] | F(u) = [0.025161 0.460804 0.001435 0.003851] | S = 13.039607
12 | u = [2.470983 0.748757 2.086766 1.216992] | F(u) = [0.005178 0.069517 0.000342 0.000914] | S = 13.046996
13 | u = [2.468085 0.761909 2.084199 1.215249] | F(u) = [0.000187 0.002179 0.000013 0.000035] | S = 13.058885
14 | u = [2.468013 0.762357 2.084136 1.215205] | F(u) = [0.000000 0.000002 0.000000 0.000000] | S = 13.059421
15 | u = [2.468013 0.762357 2.084136 1.215205] | F(u) = [0.000000 0.000002 0.000000 0.000000] | S = 13.059421

Result
u = [2.468013 0.762357 2.084136 1.215205]
F(u) = [0.000000 0.000000 0.000000 0.000000]
L(N) = [1.794934 0.865945 1.606536 1.142912]
lam(N) = [3.000000 0.662338 2.415584 1.207792]
S = 13.059421
```

Алгоритм успешно сходится за 16 итераций. С каждой итерацией алгоритма значения целевых функций стремятся к нулю, результирующая цена также постепенно уменьшается.

1.3 Вывод

Как и ожидалось, алгоритм простых итераций не сходится, поэтому пришлось использовать более надежный алгоритм Ньютона. Модифицированный алгоритм Ньютона чрезвычайно чувствителен к выбору начального приближения, поэтому был использован вариант алгоритма с расчетом матрицы Якоби. Для увеличения производительности данного алгоритма можно подключить пакет MATLAB Parallel Computing Toolbox, который использует MPI для эффективного распараллеливания.