

DAFTAR PUSTAKA

- Rahman, A., Yuniarno, E. M., & Purnama, K. E. (2014). Optimasi Penjadwalan Perkuliahan Menggunakan Metode Harmony Search. *al-Khawarizmi, II*, 47-58.
- Widiasih, B. (2007). *Program Linear Bilangan Bulat Dual*. Yogyakarta: Universitas Sanata Dharma.
- Kamila, N. S., Yorizon, & Dewi, M. P. (2018). Optimasi Penyusunan Jadwal Menggunakan Pendekatan Pembangkit Kolom (Column Generation). 57-64.
- Rafflesia, U., & Widodo, F. H. (2014). *Pemrograman Linier*. Bengkulu: Badan Penerbitan Fakultas Pertanian UNIB.
- Nufus, H., & Nurdin, E. (2016). *Program Linear*. Pekanbaru: Cahaya Firdaus.
- Meiliana. (2012). *Penentuan Batas Bawah pada Metode Branch and Price*. Medan: Universitas Sumatra Utara.
- Puspasari, A., Novianingsih, K., & Agustina, F. (2019). Penyelesaian Masalah Penjadwalan Perkuliahan Menggunakan Algoritma Genetika. *Jurnal EurekaMatika*, 7, 80-92.
- Bard, J. F., & Purnomo, H. W. (2005). Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 510-534.
- Mulyono, S. (2017). *Riset Operasi*. Klaten: Mitra Wacana Media.
- Wu, N., & Coppins, R. (1981). *Linear Programming and Extenstions*. McGraw-Hills Serie in Industrial Engineering and Management Science.
- Lesmana, N. I. (2016). Penjadwalan Produksi untuk Meminimalkan Waktu Produksi dengan Menggunakan Metode Branch and Bound. *Jurnal Teknik Industri*, 17, 42-50.
- Suhartono, E. (2015). Optimasi Penjadwalan Mata Kuliah dengan Algoritma Genetika. *INFOKAM*, 11, 132-146.
- Aplikasi *OR Simplex*
<https://play.google.com/store/apps/details?id=testbachirhabib.highdimensionalorsimplex>
- Aplikasi *Linier Optimization LITE*
<https://play.google.com/store/apps/details?id=com.vishalaksh.optimization>

LAMPIRAN

Lampiran 1 Matriks hubungan kelas dengan mata kuliah program studi Matematika

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Lampiran 2 Matriks hubungan mata kuliah yang tidak boleh bentrok program studi
Matematika

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Lampiran 3 Matriks nilai pemilihan waktu mata kuliah program studi Matematika

0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1
1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Lampiran 4 Matriks jumlah slot waktu tiap mata kuliah program studi Matematika

2
2
2
2
1
1
1
2
2
2
2
2
1
1
1
1
1
1
1
2
2
2

Lampiran 5 Program MATLAB Penjadwalan Mata Kuliah program studi Matematika

```

function Matematika
clear all; clc;

options = optimset('LargeScale', 'off', 'Simplex', 'on',
'Display', 'off');

% INPUT DATA
H=14; MK=23; jg=5;
d=xlsread('MTK.xlsx',4);
C1=xlsread('MTK.xlsx',1);
for i = 1 : jg
    C(i,:) = C1(i,:);
end
jk=2;
nk=[4; 1];
K1=xlsread('MTK.xlsx',2);
for i = 1 : jk
    K(i,:) = K1(i,:);
end
S1 = xlsread('MTK.xlsx',3);
for i = 1 : H
    S(i,:) = S1(i,:);
end

comb = combnk(1:H,2);
Pola2 = zeros(H,size(comb,1));
i = size(comb,1);
j = 1;
while i >= 1
    Pola2(comb(i,1),j) = 1;
    Pola2(comb(i,2),j) = 1;
    i = i - 1;
    j = j + 1;
end
comb = combnk(1:H,1);
Pola1 = zeros(H,size(comb,1));
for i = 1 : size(comb,1)
    Pola1(comb(i,1),i) = 1;
end

% PEMBENTUKAN MASTER PROBLEM (MP)

% Fungsi Objektif MP
jvmp = 0;
for i = 1:MK
    if d(i) == 1
        temp2 = size(Pola1,2);
    else
        temp2 = size(Pola2,2);
    end
    jvmp = temp2 + jvmp;
end
f = zeros(1,jvmp);
count = 1;

```

```

for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2); pola = Pola1;
    else
        PMK = size(Pola2,2); pola = Pola2;
    end
    for j = 1 : PMK
        temp = find(pola(:,j) == 1);
        if size(temp,1) == 1
            f(count) = S(temp,i);
        else
            f(count) = S(temp(1),i) + S(temp(2),i);
        end
        count = count + 1;
    end
end

% Pembentukan Kendala MP
A = zeros(((jk*H)+(jg*H)),size(f,2));
b = zeros(size(A,1),1);
count = 1;
for i = 1 : jk
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2); pola = Pola2;
            end
            if K(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = nk(i);
        count = count + 1;
    end
end
for i = 1 : jg
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2);
                pola = Pola2;
            end
            if C(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = 1; count = count + 1;
    end
end

```

```

    end
end
Aeq = zeros(MK,size(f,2));
beq = zeros(size(Aeq,1),1);
start = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    Aeq(i,(start+1):(PMK+start)) = ones(1,PMK);
    beq(i) = 1;
    start = PMK + start;
end

lb = zeros(1,size(f,2));
ub = ones(1,size(f,2));

% PEMBENTUKAN RESTRICTED MASTER PROBLEM (RMP)
pol = 6;
fRMP = zeros(size(f));
temp3 = fRMP;
start = 0;
count = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    [temp pos] = sort(f((start+1):(PMK+start)), 'descend');
    for j = 1 : pol
        fRMP(start+pos(j)) = f(start+pos(j));
    end
    temp2 = find(temp(1:pol) == 0);
    if ~isempty(temp2)
        for j = 1 : length(temp2)
            count = count + 1;
            temp3(count) = start + pos(temp2(j));
        end
    end
    start = PMK + start;
end
addv = temp3(1:count);

ARMP = zeros(size(A));
bRMP = b;
AeqRMP = zeros(size(Aeq));
beqRMP = beq;

% ITERASI UTAMA COLUMN GENERATION

basis = zeros(MK,3);
nul = find(fRMP == 0);
basis(1,:) = [-1 nul(1) 0];
counter = 0;

```

```

[p q] = sort(basis(:,1));
t = 1;
while p(t) < 0
    while p(t) < 0 && fRMP(basis(q(t),2)) ~= 0 && t ~= size(basis,1)
        t = t + 1;
    end
    if (t == size(basis,1) && fRMP(basis(q(t),2)) ~= 0) || p(t) > 0
        disp(' ');
        disp('Program selesai: semua basis yang negatif sudah terdapat semua di fungsi objektif RMP');
        error('Program terminated');
    end

% A. Pembentukan RMP setelah penambahan variabel baru

fRMP(basis(q(t),2)) = basis(q(t),3);
temp = find(fRMP ~= 0);
for j = 1 : size(A,1)
    for i = 1 : length(temp)
        if A(j,temp(i)) == 1
            ARMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if A(j,addv(z)) == 1
            ARMP(j,addv(z)) = 1;
        end
    end
end
for j = 1 : size(Aeq,1)
    for i = 1 : length(temp)
        if Aeq(j,temp(i)) == 1
            AeqRMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if Aeq(j,addv(z)) == 1
            AeqRMP(j,addv(z)) = 1;
        end
    end
end
for j = 1 : size(basis,1)
    for i = 1 : length(temp)
        if basis(j,temp(i)) == 1
            basis(j,temp(i)) = 0;
        end
    end
end

% B. Solve RMP
[x, fval] = linprog(
    fRMP, ARMP, bRMP, AeqRMP, beqRMP, lb, ub, [], options);

% C. Pembentukan dual RMP
fd = [bRMP;beqRMP]';
Ad = [ARMP;AeqRMP]';
bd = fRMP';
[xd, fdval] = dsimplex('min',fd,Ad,bd);

% Subproblem masing-masing MK
basis = zeros(MK,3);
for i = 1 : MK

```

```

% E. Pembentukan subproblem
pos = find(K(:,i) ~= 1);
w = xd((pos-1)*H+1):(pos*H));
pos = find(C(:,i) == 1); strt = jk*H;
v = xd((strt+((pos-1)*H)+1)):(strt+(H*pos)));
w_ = w - S(:,i);
v_ = v;
t = w_ + v_;

% F. Solve subproblem
curr_patt = zeros(H,1);
[temp pos] = sort(t);
M = 0;
for j = 1 : d(i)
    M = temp(j) + M;
    curr_patt(pos(j)) = 1;
end
% Update basis
selector = M + xd(jk*H+jg*H+i);
basis(i,1) = selector;
if selector < 0
    if d(i) == 1
        for j = 1 : size(Pola1,2)
            if Pola1(:,j) == curr_patt
                break
            end
        end
    else
        for j = 1 : size(Pola2,2)
            if Pola2(:,j) == curr_patt
                break
            end
        end
    end
    temp = 0;
    for r = 1 : (i-1)
        if d(r) == 1
            temp2 = size(Pola1,2);
        else
            temp2 = size(Pola2,2);
        end
        temp = temp2 + temp;
    end
    temp = temp + j;
    basis(i,2) = temp;
    basis(i,3) = f(temp);
end
[p, q] = sort(basis(:,1));
t = 1;

% Penghitung iterasi
counter = counter + 1;
disp('ITERASI KE:');
disp(counter);
disp('Nilai Fungsi Objektif RMP:');

```

```

        disp(-fval);
%     disp('Variabel Dual RMP:');
%     disp(xd');
disp('Nilai Fungsi Objektif Dual RMP:');
disp(fdval);
disp('Basis:');
disp(basis);
disp('=====');
disp(' ');
end

pos = find(x == 1); count = 1;
counter = 1;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
        pola = Pola1;
    else
        PMK = size(Pola2,2);
        pola = Pola2;
    end
    for j = 1 : PMK
        if count <= size(pos,1)
            if counter == pos(count)
                fprintf('x%d%d : ',j,i);
                disp(pola(:,j)');
                count = count + 1;
            end
        end
        counter = counter + 1;
    end
end
function [x,z] = dsimplex(type, c, A, b)
% The Dual Simplex Algorithm for solving the LP problem xz
% min (max) z = c*x
% Subject to Ax >= b
% x >= 0

if (type == 'min')
    mm = 0;
else
    mm = 1;
    c = -c;
end
A = -A;
b = -b(:);
c = c(:)';
[m, n] = size(A);
A = [A eye(m) b];
A = [A; [c zeros(1,m+1)]];
subs = n+1:n+m;
[bmin, row] = Br(b);
tbn = 0;
while ~isempty(bmin) && bmin < 0 && abs(bmin) > eps
    if A(row,1:m+n) >= 0
        varargout(1)={subs(:)};

```

```

varargout(2)={A};
varargout(3) = {zeros(n,1)};
varargout(4) = {0};
return
end
col = MRTD(A(m+1,1:m+n),A(row,1:m+n));
subs(row) = col;
A(row,:)= A(row,:)/A(row,col);
for i = 1:m+1
    if i ~= row
        A(i,:)= A(i,:)-A(i,col)*A(row,:);
    end
end
tbn = tbn + 1;
[bmin, row] = Br(A(1:m,m+n+1));
end
x = zeros(m+n,1);
x(subs) = A(1:m,m+n+1);
x = x(1:n);
if mm == 0
    z = -A(m+1,m+n+1);
else
    z = A(m+1,m+n+1);
end
end
function [m, j] = Br(d)
% Implementation of the Bland's rule applied to the array d.
% This function is called from within the following functions:
% simplex2p, dsimplex, addconstr, simplex and cpa.
% Output parameters:
% m - first negative number in the array d
% j - index of the entry m.
ind = find(d < 0);
if ~isempty(ind)
    j = ind(1);
    m = d(j);
else
    m = [];
    j = [];
end
end
function col = MRTD(a, b)

% The Maximum Ratio Test performed on vectors a and b.
% This function is called from within the function dsimplex.
% Output parameter:
% col - index of the pivot column.

m = length(a);
c = 1:m;
a = a(:);
b = b(:);
l = c(b < 0);
[mi, col] = max(a(l)./b(l));
col = l(col);
end

```

Lampiran 6 Matriks hubungan kelas dengan mata kuliah program studi Ilmu Komputer

Lampiran 7 Matriks hubungan mata kuliah yang tidak boleh bentrok program studi Ilmu Komputer

Lampiran 8 Matriks nilai pemilihan waktu mata kuliah program studi Ilmu Komputer

0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Lampiran 9 Matriks jumlah slot waktu tiap mata kuliah program studi Ilmu Komputer

2
2
2
2
2
1
1
1
1
1
2
2
2
2
2
1
1
1
1
1
1
1
1
1
1
1
1
1
2
1

Lampiran 10 Program MATLAB Penjadwalan Mata Kuliah program studi Ilmu Komputer

```

function Ilmu_Komputer
clear all; clc;

options = optimset('LargeScale', 'off', 'Simplex', 'on',
'Display', 'off');

% INPUT DATA
H=14; MK=32; jg=6;
d=xlsread(' ILKOM.xlsx',4);
C1=xlsread(' ILKOM.xlsx',1);
for i = 1 : jg
    C(i,:) = C1(i,:);
end
jk=2;
nk=[4; 1];
K1=xlsread(' ILKOM.xlsx',2);
for i = 1 : jk
    K(i,:) = K1(i,:);
end
S1 = xlsread(' ILKOM.xlsx',3);
for i = 1 : H
    S(i,:) = S1(i,:);
end

comb = combnk(1:H,2);
Pola2 = zeros(H,size(comb,1));
i = size(comb,1);
j = 1;
while i >= 1
    Pola2(comb(i,1),j) = 1;
    Pola2(comb(i,2),j) = 1;
    i = i - 1;
    j = j + 1;
end
comb = combnk(1:H,1);
Pola1 = zeros(H,size(comb,1));
for i = 1 : size(comb,1)
    Pola1(comb(i,1),i) = 1;
end

% PEMBENTUKAN MASTER PROBLEM (MP)

% Fungsi Objektif MP
jvmp = 0;
for i = 1:MK
    if d(i) == 1
        temp2 = size(Pola1,2);
    else
        temp2 = size(Pola2,2);
    end
    jvmp = temp2 + jvmp;
end
f = zeros(1,jvmp);
count = 1;

```

```

for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2); pola = Pola1;
    else
        PMK = size(Pola2,2); pola = Pola2;
    end
    for j = 1 : PMK
        temp = find(pola(:,j) == 1);
        if size(temp,1) == 1
            f(count) = S(temp,i);
        else
            f(count) = S(temp(1),i) + S(temp(2),i);
        end
        count = count + 1;
    end
end

% Pembentukan Kendala MP
A = zeros(((jk*H)+(jg*H)),size(f,2));
b = zeros(size(A,1),1);
count = 1;
for i = 1 : jk
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2); pola = Pola2;
            end
            if K(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = nk(i);
        count = count + 1;
    end
end
for i = 1 : jg
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2);
                pola = Pola2;
            end
            if C(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = 1; count = count + 1;
    end
end

```

```

    end
end
Aeq = zeros(MK,size(f,2));
beq = zeros(size(Aeq,1),1);
start = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    Aeq(i,(start+1):(PMK+start)) = ones(1,PMK);
    beq(i) = 1;
    start = PMK + start;
end

lb = zeros(1,size(f,2));
ub = ones(1,size(f,2));

% PEMBENTUKAN RESTRICTED MASTER PROBLEM (RMP)
pol = 6;
fRMP = zeros(size(f));
temp3 = fRMP;
start = 0;
count = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    [temp pos] = sort(f((start+1):(PMK+start)), 'descend');
    for j = 1 : pol
        fRMP(start+pos(j)) = f(start+pos(j));
    end
    temp2 = find(temp(1:pol) == 0);
    if ~isempty(temp2)
        for j = 1 : length(temp2)
            count = count + 1;
            temp3(count) = start + pos(temp2(j));
        end
    end
    start = PMK + start;
end
addv = temp3(1:count);

ARMP = zeros(size(A));
bRMP = b;
AeqRMP = zeros(size(Aeq));
beqRMP = beq;

% ITERASI UTAMA COLUMN GENERATION

basis = zeros(MK,3);
nul = find(fRMP == 0);
basis(1,:) = [-1 nul(1) 0];
counter = 0;

```

```

[p q] = sort(basis(:,1));
t = 1;
while p(t) < 0
    while p(t) < 0 && fRMP(basis(q(t),2)) ~= 0 && t ~= size(basis,1)
        t = t + 1;
    end
    if (t == size(basis,1) && fRMP(basis(q(t),2)) ~= 0) || p(t) > 0
        disp(' ');
        disp('Program selesai: semua basis yang negatif sudah terdapat semua di fungsi objektif RMP');
        error('Program terminated');
    end

% A. Pembentukan RMP setelah penambahan variabel baru

fRMP(basis(q(t),2)) = basis(q(t),3);
temp = find(fRMP ~= 0);
for j = 1 : size(A,1)
    for i = 1 : length(temp)
        if A(j,temp(i)) == 1
            ARMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if A(j,addv(z)) == 1
            ARMP(j,addv(z)) = 1;
        end
    end
end
for j = 1 : size(Aeq,1)
    for i = 1 : length(temp)
        if Aeq(j,temp(i)) == 1
            AeqRMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if Aeq(j,addv(z)) == 1
            AeqRMP(j,addv(z)) = 1;
        end
    end
end
for j = 1 : size(basis,1)
    for i = 1 : length(temp)
        if basis(j,temp(i)) == 1
            basis(j,temp(i)) = 0;
        end
    end
end

% B. Solve RMP
[x, fval] = linprog(
    fRMP, ARMP, bRMP, AeqRMP, beqRMP, lb, ub, [], options);

% C. Pembentukan dual RMP
fd = [bRMP;beqRMP]';
Ad = [ARMP;AeqRMP]';
bd = fRMP';
[xd, fdval] = dsimplex('min',fd,Ad,bd);

% Subproblem masing-masing MK
basis = zeros(MK,3);
for i = 1 : MK

```

```

% E. Pembentukan subproblem
pos = find(K(:,i) ~= 1);
w = xd((pos-1)*H+1):(pos*H));
pos = find(C(:,i) == 1); strt = jk*H;
v = xd((strt+((pos-1)*H)+1)):(strt+(H*pos)));
w_ = w - S(:,i);
v_ = v;
t = w_ + v_;

% F. Solve subproblem
curr_patt = zeros(H,1);
[temp pos] = sort(t);
M = 0;
for j = 1 : d(i)
    M = temp(j) + M;
    curr_patt(pos(j)) = 1;
end
% Update basis
selector = M + xd(jk*H+jg*H+i);
basis(i,1) = selector;
if selector < 0
    if d(i) == 1
        for j = 1 : size(Pola1,2)
            if Pola1(:,j) == curr_patt
                break
            end
        end
    else
        for j = 1 : size(Pola2,2)
            if Pola2(:,j) == curr_patt
                break
            end
        end
    end
    temp = 0;
    for r = 1 : (i-1)
        if d(r) == 1
            temp2 = size(Pola1,2);
        else
            temp2 = size(Pola2,2);
        end
        temp = temp2 + temp;
    end
    temp = temp + j;
    basis(i,2) = temp;
    basis(i,3) = f(temp);
end
[p, q] = sort(basis(:,1));
t = 1;

% Penghitung iterasi
counter = counter + 1;
disp('ITERASI KE:');
disp(counter);
disp('Nilai Fungsi Objektif RMP:');

```

```

        disp(-fval);
%     disp('Variabel Dual RMP:');
%     disp(xd');
disp('Nilai Fungsi Objektif Dual RMP:');
disp(fdval);
disp('Basis:');
disp(basis);
disp('=====');
disp(' ');
end

pos = find(x == 1); count = 1;
counter = 1;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
        pola = Pola1;
    else
        PMK = size(Pola2,2);
        pola = Pola2;
    end
    for j = 1 : PMK
        if count <= size(pos,1)
            if counter == pos(count)
                fprintf('x%d%d : ',j,i);
                disp(pola(:,j)');
                count = count + 1;
            end
        end
        counter = counter + 1;
    end
end
function [x,z] = dsimplex(type, c, A, b)
% The Dual Simplex Algorithm for solving the LP problem xz
% min (max) z = c*x
% Subject to Ax >= b
% x >= 0

if (type == 'min')
    mm = 0;
else
    mm = 1;
    c = -c;
end
A = -A;
b = -b(:);
c = c(:)';
[m, n] = size(A);
A = [A eye(m) b];
A = [A; [c zeros(1,m+1)]];
subs = n+1:n+m;
[bmin, row] = Br(b);
tbn = 0;
while ~isempty(bmin) && bmin < 0 && abs(bmin) > eps
    if A(row,1:m+n) >= 0
        varargout{1}={subs(:)};

```

```

varargout(2)={A};
varargout(3) = {zeros(n,1)};
varargout(4) = {0};
return
end
col = MRTD(A(m+1,1:m+n),A(row,1:m+n));
subs(row) = col;
A(row,:)= A(row,:)/A(row,col);
for i = 1:m+1
    if i ~= row
        A(i,:)= A(i,:)-A(i,col)*A(row,:);
    end
end
tbn = tbn + 1;
[bmin, row] = Br(A(1:m,m+n+1));
end
x = zeros(m+n,1);
x(subs) = A(1:m,m+n+1);
x = x(1:n);
if mm == 0
    z = -A(m+1,m+n+1);
else
    z = A(m+1,m+n+1);
end
end
function [m, j] = Br(d)
% Implementation of the Bland's rule applied to the array d.
% This function is called from within the following functions:
% simplex2p, dsimplex, addconstr, simplex and cpa.
% Output parameters:
% m - first negative number in the array d
% j - index of the entry m.
ind = find(d < 0);
if ~isempty(ind)
    j = ind(1);
    m = d(j);
else
    m = [];
    j = [];
end
end
function col = MRTD(a, b)

% The Maximum Ratio Test performed on vectors a and b.
% This function is called from within the function dsimplex.
% Output parameter:
% col - index of the pivot column.

m = length(a);
c = 1:m;
a = a(:);
b = b(:);
l = c(b < 0);
[mi, col] = max(a(l)./b(l));
col = l(col);
end

```

Lampiran 11 Matriks hubungan kelas dengan mata kuliah program studi Aktuaria

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Lampiran 12 Matriks hubungan mata kuliah yang tidak boleh bentrok program studi Aktuaria

Lampiran 13 Matriks nilai pemilihan waktu mata kuliah program studi Aktuaria

0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Lampiran 14 Matriks jumlah slot waktu tiap mata kuliah program studi Aktuaria

2
2
2
2
1
1
1
2
2
2
2
2
1
1
1
1
1
1
1
2
2
2
2

Lampiran 15 Program MATLAB Penjadwalan Mata Kuliah program studi Aktuaria

```

function Aktuaria
clear all; clc;

options = optimset('LargeScale', 'off', 'Simplex', 'on',
'Display', 'off');

% INPUT DATA
H=14; MK=24; jg=5;
d=xlsread('AKTUARIA.xlsx',4);
C1=xlsread('AKTUARIA.xlsx',1);
for i = 1 : jg
    C(i,:) = C1(i,:);
end
jk=2;
nk=[4; 1];
K1= xlsread('AKTUARIA.xlsx',2);
for i = 1 : jk
    K(i,:) = K1(i,:);
end
S1 = xlsread('AKTUARIA.xlsx',3);
for i = 1 : H
    S(i,:) = S1(i,:);
end

comb = combnk(1:H,2);
Pola2 = zeros(H,size(comb,1));
i = size(comb,1);
j = 1;
while i >= 1
    Pola2(comb(i,1),j) = 1;
    Pola2(comb(i,2),j) = 1;
    i = i - 1;
    j = j + 1;
end
comb = combnk(1:H,1);
Polal = zeros(H,size(comb,1));
for i = 1 : size(comb,1)
    Polal(comb(i,1),i) = 1;
end

% PEMBENTUKAN MASTER PROBLEM (MP)

% Fungsi Objektif MP
jvmp = 0;
for i = 1:MK
    if d(i) == 1
        temp2 = size(Polal,2);
    else
        temp2 = size(Pola2,2);
    end
    jvmp = temp2 + jvmp;
end
f = zeros(1,jvmp);
count = 1;
for i = 1 : MK

```

```

if d(i) == 1
    PMK = size(Pola1,2); pola = Pola1;
else
    PMK = size(Pola2,2); pola = Pola2;
end
for j = 1 : PMK
    temp = find(pola(:,j) == 1);
    if size(temp,1) == 1
        f(count) = S(temp,i);
    else
        f(count) = S(temp(1),i) + S(temp(2),i);
    end
    count = count + 1;
end
end

% Pembentukan Kendala MP
A = zeros(((jk*H)+(jg*H)),size(f,2));
b = zeros(size(A,1),1);
count = 1;
for i = 1 : jk
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2); pola = Pola2;
            end
            if K(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = nk(i);
        count = count + 1;
    end
end
for i = 1 : jg
    for j = 1 : H
        start = 0;
        for k = 1 : MK
            if d(k) == 1
                PMK = size(Pola1,2);
                pola = Pola1;
            else
                PMK = size(Pola2,2);
                pola = Pola2;
            end
            if C(i,k) == 1
                A(count,(start+1):(PMK+start)) = pola(j,:);
            end
            start = PMK + start;
        end
        b(count) = 1; count = count + 1;
    end
end

```

```

end
Aeq = zeros(MK,size(f,2));
beq = zeros(size(Aeq,1),1);
start = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    Aeq(i,(start+1):(PMK+start)) = ones(1,PMK);
    beq(i) = 1;
    start = PMK + start;
end

lb = zeros(1,size(f,2));
ub = ones(1,size(f,2));

% PEMBENTUKAN RESTRICTED MASTER PROBLEM (RMP)
pol = 6;
fRMP = zeros(size(f));
temp3 = fRMP;
start = 0;
count = 0;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
    else
        PMK = size(Pola2,2);
    end
    [temp pos] = sort(f((start+1):(PMK+start)), 'descend');
    for j = 1 : pol
        fRMP(start+pos(j)) = f(start+pos(j));
    end
    temp2 = find(temp(1:pol) == 0);
    if ~isempty(temp2)
        for j = 1 : length(temp2)
            count = count + 1;
            temp3(count) = start + pos(temp2(j));
        end
    end
    start = PMK + start;
end
addv = temp3(1:count);

ARMP = zeros(size(A));
bRMP = b;
AeqRMP = zeros(size(Aeq));
beqRMP = beq;

% ITERASI UTAMA COLUMN GENERATION

basis = zeros(MK,3);
nul = find(fRMP == 0);
basis(1,:) = [-1 nul(1) 0];
counter = 0;
[p q] = sort(basis(:,1));

```

```

t = 1;
while p(t) < 0
    while p(t) < 0 && fRMP(basis(q(t),2)) ~= 0 && t ~= size(basis,1)
        t = t + 1;
    end
    if (t == size(basis,1) && fRMP(basis(q(t),2)) ~= 0) || p(t) >0
        disp(' ');
        disp('Program selesai: semua basis yang negatif sudah terdapat semua di fungsi objektif RMP');
        error('Program terminated');
    end

% A. Pembentukan RMP setelah penambahan variabel baru

fRMP(basis(q(t),2)) = basis(q(t),3);
temp = find(fRMP ~= 0);
for j = 1 : size(A,1)
    for i = 1 : length(temp)
        if A(j,temp(i)) == 1
            ARMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if A(j,addv(z)) == 1
            ARMP(j,addv(z)) = 1;
        end
    end
end
for j = 1 : size(Aeq,1)
    for i = 1 : length(temp)
        if Aeq(j,temp(i)) == 1
            AeqRMP(j,temp(i)) = 1;
        end
    end
    for z = 1 : length(addv)
        if A(j,addv(z)) == 1
            AeqRMP(j,addv(z)) = 1;
        end
    end
end
end

% B. Solve RMP
[x, fval] = linprog(
fRMP,ARMP,bRMP,AeqRMP,beqRMP,lb,ub,[],options);

% C. Pembentukan dual RMP
fd = [bRMP;beqRMP]';
Ad = [ARMP;AeqRMP]';
bd = fRMP';
[xd, fdval] = dsimplex('min',fd,Ad,bd);

% Subproblem masing-masing MK
basis = zeros(MK,3);
for i = 1 : MK

```

```

% E. Pembentukan subproblem
pos = find(K(:,i) ~= 1);
w = xd((pos-1)*H+1):(pos*H));
pos = find(C(:,i) == 1); strt = jk*H;
v = xd((strt+((pos-1)*H)+1)):(strt+(H*pos)));
w_ = w - S(:,i);
v_ = v;
t = w_ + v_;

% F. Solve subproblem
curr_patt = zeros(H,1);
[temp pos] = sort(t);
M = 0;
for j = 1 : d(i)
    M = temp(j) + M;
    curr_patt(pos(j)) = 1;
end
% Update basis
selector = M + xd(jk*H+jg*H+i);
basis(i,1) = selector;
if selector < 0
    if d(i) == 1
        for j = 1 : size(Pola1,2)
            if Pola1(:,j) == curr_patt
                break
            end
        end
    else
        for j = 1 : size(Pola2,2)
            if Pola2(:,j) == curr_patt
                break
            end
        end
    end
    temp = 0;
    for r = 1 : (i-1)
        if d(r) == 1
            temp2 = size(Pola1,2);
        else
            temp2 = size(Pola2,2);
        end
        temp = temp2 + temp;
    end
    temp = temp + j;
    basis(i,2) = temp;
    basis(i,3) = f(temp);
end
[p, q] = sort(basis(:,1));
t = 1;

% Penghitung iterasi
counter = counter + 1;
disp('ITERASI KE:');
disp(counter);
disp('Nilai Fungsi Objektif RMP:');
disp(-fval);

```

```

%      disp('Variabel Dual RMP:');
%      disp(xd');
disp('Nilai Fungsi Objektif Dual RMP:');
disp(fdval);
disp('Basis:');
disp(basis);
disp('=====');
disp(' ');
end

pos = find(x == 1); count = 1;
counter = 1;
for i = 1 : MK
    if d(i) == 1
        PMK = size(Pola1,2);
        pola = Pola1;
    else
        PMK = size(Pola2,2);
        pola = Pola2;
    end
    for j = 1 : PMK
        if count <= size(pos,1)
            if counter == pos(count)
                fprintf('x%d%d : ',j,i);
                disp(pola(:,j)');
                count = count + 1;
            end
        end
        counter = counter + 1;
    end
end
end
function [x,z] = dsimplex(type, c, A, b)
% The Dual Simplex Algorithm for solving the LP problem xz
% min (max) z = c*x
% Subject to Ax >= b
% x >= 0

if (type == 'min')
    mm = 0;
else
    mm = 1;
    c = -c;
end
A = -A;
b = -b(:);
c = c(:)';
[m, n] = size(A);
A = [A eye(m) b];
A = [A; [c zeros(1,m+1)]];
subs = n+1:n+m;
[bmin, row] = Br(b);
tbn = 0;
while ~isempty(bmin) && bmin < 0 && abs(bmin) > eps
    if A(row,1:m+n) >= 0
        varargout(1)={subs(:)};
        varargout(2)={A};
    end
end

```

```

varargout(3) = {zeros(n,1)};
varargout(4) = {0};
return
end
col = MRTD(A(m+1,1:m+n),A(row,1:m+n));
subs(row) = col;
A(row,:)= A(row,:)/A(row,col);
for i = 1:m+1
    if i ~= row
        A(i,:)= A(i,:)-A(i,col)*A(row,:);
    end
end
tbn = tbn + 1;
[bmin, row] = Br(A(1:m,m+n+1));
end
x = zeros(m+n,1);
x(subs) = A(1:m,m+n+1);
x = x(1:n);
if mm == 0
    z = -A(m+1,m+n+1);
else
    z = A(m+1,m+n+1);
end
end
function [m, j] = Br(d)
% Implementation of the Bland's rule applied to the array d.
% This function is called from within the following functions:
% simplex2p, dsimplex, addconstr, simplex and cpa.
% Output parameters:
% m - first negative number in the array d
% j - index of the entry m.
ind = find(d < 0);
if ~isempty(ind)
    j = ind(1);
    m = d(j);
else
    m = [];
    j = [];
end
end
function col = MRTD(a, b)

% The Maximum Ratio Test performed on vectors a and b.
% This function is called from within the function dsimplex.
% Output parameter:
% col - index of the pivot column.

m = length(a);
c = 1:m;
a = a(:);
b = b(:);
l = c(b < 0);
[mi, col] = max(a(l)./b(l));
col = l(col);
end

```

Lampiran 16 Program MATLAB Penjadwalan Dosen

```
clear all; clc;
f = xlsread('Data Dosen.xlsx',3);
A = xlsread('Data Dosen.xlsx',1);
b = xlsread('Data Dosen.xlsx',2);
[x,Z] = bintprog(-f,A,b);
disp(x);
```