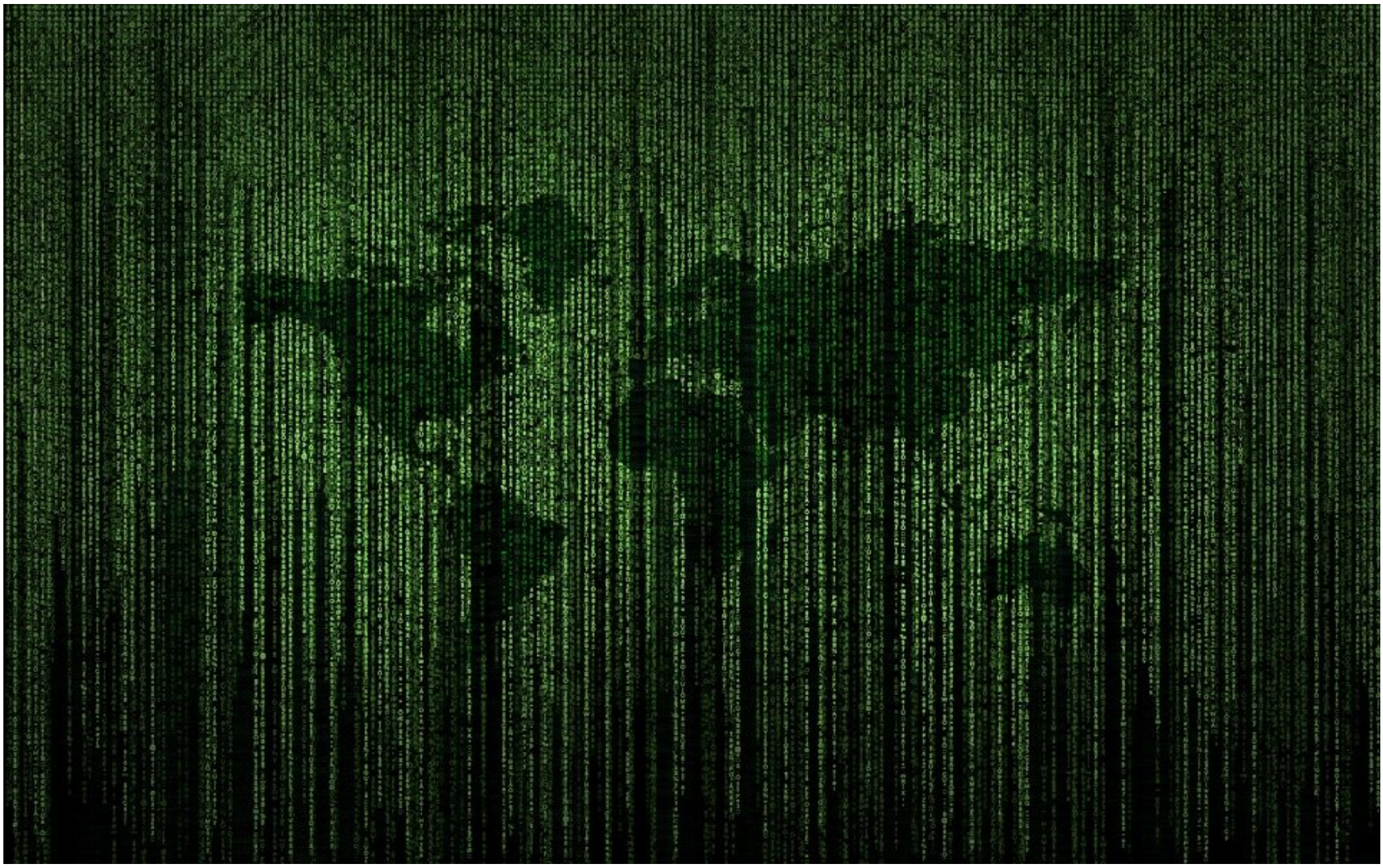
Matlab gruppeaflevering

Gruppe 8

Christopher Houstrup Heltborg – 202105942
Mathias Bruun Houmøller – 202006837
Christoffer Regnar Mølck – 202009347
Martin Teit Bruun Andersen – 202109592

24. november 2022

Denne aflevering har til formål at udvikle MATLAB-kode til at løse forskellige kvadratiske tildelingsproblemer.
Dette omhandler optimal placering af forskellige enheder. Eller et forsøg på den bedst mulige med givne forudsætninger.
Der benyttes forskellige metoder gennem afleveringen til netop dette formål.



"It's not a bug; it's an undocumented feature" — Anonymous

Delopgave 1

Denne opgave omhandler hvorledes man kan udregne en målfunktionsværdi MV for en given placering af enheder.

Her testes funktionen med følgende værdier givet fra opgaven

Test af funktion

```
% Definere de to test matricer
Dst = [0 8 17; 8 0 11; 17 11 0];
Tr = [0 132 0; 132 0 103; 0 103 0];

% Tester funktionen med 2 forskellige rækkefølger
MV1 = MaalFkt([1 2 3],Tr,Dst)
MV2 = MaalFkt([3 1 2],Tr,Dst)

Dst = [0 8 17 22;
    8 0 11 6;
    17 11 0 4;
    22 6 4 0];
Tr = [0 47 0 85;
    47 0 103 77;
    0 103 0 0
    85 77 0 0];

MV3 = MaalFkt([2 3 4 1], Tr, Dst)
```

Følgende output gives hvilket matcher resultaterne fra opgaven.

Output

```
MV1 = 2189
MV2 = 3203
MV3 = 3507
```

Nedenfor ses MATLAB-koden til delopgave 1.

Matlab function

```
function MV = MaalFkt(Pl,Tr,Dst)
% MaalFkt - Bestemmer den totale score af funktionen ved at
% gange de to matricer sammen en indgang af gangen
% for alle rækker i og kolonner j
% Kald: MV = MaalFkt(Pl,Tr,Dst)
% Input:
% Pl = Placeringen af lokationer
% Tr = Numersik værdi af tilskrevede placering
% fx Tr(2,1) kan være mængden der transporteres fra lokation 1 til 2
% Dst = Afstande mellem placeringer
% fx Dst(2,1) kan være værdien fra lokation 1 til 2
% Output:
% MV = den totale score

    sum = 0; % bruges i forloop til at sumere continuert

    %næstede forloop starter
    for i = 1:length(Pl)-1 %forloop der køre fra 1 til længden af Pl

        for j = i+1:length(Tr) %forloop der køre fra i til længden af Tr
            % Første tal i j fjernes for hver gang dette forloop er kørt
            % Hver indgang ganges nu sammen i de to matricer Dst og Tr
            sum = Dst(i,j) .* Tr(Pl(i), Pl(j)) + sum;
            % Det ses at der bruges indgangen for placeringen(Pl) for Tr
            % da placeringenerne kan ændres men afstanden imellem pladser
            % ikke ændres
        end
    end
    % Ved slut er alle tal i den øvre trekant af matrixen gennemgået
    MV = sum; % summen af værdierne udskrives
end
```

I delopgave 1 og 2 bruges funktionen til at udregne Ton Km ved at have en Tr matrix med Ton værdier og en Dst matrix med kilometer afstandsværdier.
I delopgave 3 og 4 bruges funktionen til at udregne den optimale bordplan. Her er Tr et tal for personrelationerne og Dst er tal for kommunikations potentialet.

Delopgave 2

Denne opgave omhandler hvordan den optimale placering af enheder i en kvadratisk tildelingsproblem. Der skal opstilles kode som er så general som mulig, således den ville fungere uanset problemstørrelsen.

Her testes funktionen med følgende værdier givet fra opgaven.

Test af funktion

```
%test af 3 enheder til 3 lokaliteter
%test data ens med delopgave 1
Dst = [0 8 17; 8 0 11; 17 11 0];
Tr = [0 132 0; 132 0 103; 0 103 0];

%kalder funktionen
disp('Opt1:')
Opt1 = OptPlaceMinMV(Tr,Dst);

%test af 4 enheder til 4 lokaliteter
%test data
Dst = [0 8 17 22; 8 0 11 6; 17 11 0 4; 22 6 4 0];
Tr = [0 47 0 85; 47 0 103 77; 0 103 0 0; 85 77 0 0];

%kalder funktionen
disp('Opt2:')
Opt2 = OptPlaceMinMV(Tr,Dst);
```

Følgende outputs fåes ved testen. Disse matcher resultater givet i opgaven.

Output

```
Opt1:
OptMV = 2189
OptPl = 1x3
     1     2     3

Opt2:
OptMV = 2143
OptPl = 1x4
     3     2     1     4
```

Nedenfor ses MATLAB-koden til delopgave 2.

Matlab function

```
function [OptPl,OptMV] = OptPlacMinMV(Tr,Dst)
% OptPlaceMinMV - Returnerer den optimale rute samt tilsvarende numerisk
% værdi anskrevet denne rute på baggrund af input dataene.
% Kald: Opt = OptPlaceMinMV(Tr,Dst);
% Input:
% Tr = Numeriske værdier tilskrevet placering
% Dst = Afstanden mellem placeringer
% Output:
% Opt = Ruten der returnerer den mindste numeriske værdi.

    % Mut returnerer en ændring af routen baseret på størrelsen af matrice
    % inputtet fra Tr.
    mut = perms(1:size(Tr, 1));

    % Et tomt array oprettes som kan lagere værdier fra for-loop.
    arr = [];

    % Der tilføjes en tom plads, hvori dataen danner row & colum fra den
    % tidligere oprettede Maalfunktion. end+1 gives som indeks til arrayet
    % da det ikke ønskes muligt at overskrive tidligere indeks.
    for i = 1:size(mut, 1)
        arr(end+1) = MaalFkt(mut(i,:),Tr,Dst);
    end

    % OptMV returnerer den mindste numeriske værdi fundet i arr.
    OptMV = min(arr);

    % k & OptPl finder den tilhørende rute til den største numeriske værdi
       .
    k = find(arr==min(arr));
    OptPl = mut(k,:);
end
```

Det ses at de optimale placeringer for lagre og fabrikker og den samlede Ton Km afstand findes af matricer af alle størrelser så længe de to er lige store.
Bemærk dog at perms funktionen er meget beregningstung og dermed sættes hukomelsesbegræsning der gør at matricer maximalt kan være 11×11 og stadig udregnes inden for rimlig tid.

Delopgave 3

Følgende opgave omhandler at finde den optimale placering af personer i det givne eksempel, hvor målværdien skal maksimeres.

Her testes funktionen med følgende værdier givet fra opgaven.

Test af funktion

```
%Test data improteres fra excel fil
KommPot = xlsread('Data, pladser og personer.xlsx', ...
'11 pladser-personer','B14:L24');
PersRel = xlsread('Data, pladser og personer.xlsx', ...
'11 pladser-personer','B29:L39');

%kalder funktionen
Opt = OptPlaceMaxMV(PersRel,KommPot);
```

Følgende outputs fåes ved testen. Disse matcher resultater givet i opgaven.

Output

```
OptMV = 18
OptPl = 3x11
    11     10      9      7      3      4      6      5      1      2      8
     1     10      9      7      3      4      6      5     11      8      2
     1     10      9      7      3      4      6      5     11      2      8
```

Nedenfor ses MATLAB-koden til delopgave 3.

Matlab function

```
function [OptPl,OptMV] = OptPlacMaxMV(Tr,Dst)
% OptPlaceMaxMV - Returnerer den optimale rute samt tilsvarende numerisk
% værdi anskrevet denne rute på baggrund af input dataene.
% Kald: Opt = OptPlaceMinMV(Tr,Dst);
% Input:
% Tr = Numeriske værdier tilskrevet placering
% Dst = Afstanden mellem placeringer
% Output:
% Opt = Ruten der returnerer den største numeriske værdi.

    % Mut returnerer en ændring af routen baseret på størrelsen af matrice
    % inputtet fra Tr.
    mut = perms(1:size(Tr, 1));

    % Et tomt array oprettes som kan lagere værdier fra for-loop.
    arr = [];

    % Der tilføjes en tom plads, hvori dataen danner row & colum fra den
    % tidligere oprettede Maalfunktion. end+1 gives som indeks til arrayet
    % da det ikke ønskes muligt at overskrive tidligere indeks.
    for i = 1:size(mut, 1)
        arr(end+1) = MaalFkt(mut(i,:),Tr,Dst);
    end

    % OptMV returnerer den største numeriske værdi fundet i arr.
    OptMV = max(arr);

    % k & OptPl finder den tilhørende rute til den største numeriske værdi
        .
    k = find(arr==max(arr));
    OptPl = mut(k,:);
end
```

Bemærk at funktionen tager lang tid at køre da der bruges to 11×11 matricer som input, dette skyldes perms funktionen og en 12×12 ville nok ikke kunne køre.
Det gode ved funktionen er at den finder den suverænt bedste bordplan eller som i dette tilfælde hvor der er 3 som er lige gode.

Delopgave 4

Denne opgave omhandler at finde den mest optimale løsning ved at udregne den suboptimale løsning hvor målværdien maksimeres

Her testes funktionen med følgende værdier givet fra opgaven

Test af funktion

```
%Test data improteres fra excel fil
KommPot = xlsread('Data, pladser og personer.xlsx', ...
                  '11 pladser-personer','B14:L24');
PersRel = xlsread('Data, pladser og personer.xlsx', ...
                  '11 pladser-personer','B29:L39');

%Start bordplan
Bordplan0 = [6 3 11 7 8 5 1 2 4 9 10];

%kalder funktionen
SubOptPlacMaxMV(PersRel,KommPot,Bordplan0);
```

Følgende outputs fåes ved testen. Disse matcher resultater givet i opgaven

Output

```
SubOptMV = 14
SubOptPl = 1x11
          7     5     4     6     8    11     2     1     3     9    10
```

Til højre ses MATLAB-koden til delopgave 4.
Det bemærkes at koden er meget hurtigere at køre end den fra delopgave 3 men giver en mindre godt resultat. Ved store data sæt kan den metode dog blive nødvendig.

Matlab function

```
function [SubOptPl,SubOptMV] = SubOptPlacMaxMV(Tr,Dst,Pl0)
% SubOptPlacMaxMV - Returnerer den rute der afgiver den højeste numeriske
% værdi baseret på Maalfunktionen (MaalFkt) ved suboptimal placering.
% Kald: SubOptPlacMaxMV(Tr,Dst,Pl0)
% Input:
% Pl0 = Start placeringen af lokationer eller udgangspunktet.
% Tr = Numeriske værdier tilskrevet placering.
% Dst = Afstanden mellem placeringer eller kommunikationspotentialet.
% Output:
% SubOptPlacMaxMV(Tr,Dst,Pl0) - Numerisk værdi & vektor med "rute".
% SubOptPl = Numerisk værdi der afgøre hvor god placeringen er,
% højst er bedst.
% SubOptMV = vektor med placeringerne der har en god placering.

% Stedholdere for den tidligere bedste numeriske værdi og den mulige
% opkommende bedste numeriske værdi.
max = MaalFkt(Pl0, Tr, Dst);
oldMax = MaalFkt(Pl0, Tr, Dst) - 1;

% Variable & Stedholder for bedste rute.
bedsteMatrix = Pl0;
while max > oldMax %Der laves et while loop som sørger for at kontroller
    % om den nye plan kan føre til en bedre placering.
    % Hvis dette ikke er muligt stoppes loopet.
    oldMax = MaalFkt(bedsteMatrix, Tr, Dst);
    % Nestede forloop starter
    for j = 1:length(Tr)
        for i = 1:length(Tr)
            mutMatrix = bedsteMatrix;
            % Et eksempel på en meget simpelt mutMatrix [1, 2] i først
            % for loop sker der ikke noget da i og j = 1. I næste for
            % loop så vil position i og j blive flippede. muMatrix
            % vil således blive: [2, 1]
            pos1 = bedsteMatrix(j);
            pos2 = bedsteMatrix(i);
            mutMatrix(i) = pos1;
            mutMatrix(j) = pos2;

            %Hvis MaalFkt giver en større max værdi end den tidligere
            %satte max så vil denne nye MaalFkt blive sat til max og
            %den nye bedste matrix sat.
            if MaalFkt(mutMatrix, Tr, Dst) > max
                max = MaalFkt(mutMatrix, Tr, Dst);
                bedsteMatrix = mutMatrix;
            end
        end
    end
end
%Hvis alle mulige græne som ikke er afskået er gennemgået og der
%ikke fandtes en max værdi som er støre en den tidligere fundet max
%værdi så må vi have fundet den sub optimale løsning hvilket udskrives.
SubOptMV = max;
SubOptPl = bedsteMatrix;
end
```