קובץ הכנה – DATA

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 20.1.0.0 Data

\* Author: almog

\* Creation Date: Aug 26, 2021 at 11:44:27 AM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SheetConnection sheet("Small.xlsx");

S from SheetRead (sheet, "S");

/\* Part A - Export 'B\_round' to XL \*/

/\*

B\_round to SheetWrite (sheet,"B\_round");

\*/

/\* Part B - Once we have the 'B\_round' matrix, calculate two more matrices and export them to XL \*/

/\* Before running this part - delete '/\*' from part B , and add for Part A \*/

t from SheetRead (sheet, "t");

B\_round from SheetRead (sheet, "B\_round");

Num\_Stations\_in\_Between to SheetWrite (sheet,"NSiB");

Time\_Between\_Stations to SheetWrite (sheet,"TBS");

קובץ הכנה – MOOD

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 20.1.0.0 Model

\* Author: Almog, Daniel, Nir

\* Creation Date: Aug 26, 2021 at 11:44:07 AM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int S=...;

range stations\_index = 0..(S-1);

/\* Part A - Calculation of B\_round - a boolean matrix which sets '1' if we pass station '0'

on the way between station i to station j, and sets '0' otherwise. \*/

/\*

int M=100\*S;

dvar boolean B\_round[stations\_index ,stations\_index];

minimize 0;

subject to{

forall (o in stations\_index, d in stations\_index) B\_round[o][d] \* (d-o) \* M <= 0 ;

forall (o in stations\_index, d in stations\_index) d-(o\*(1-B\_round[o][d]))>= 0 ;

}

\*/

/\* Part B - Once we have the 'B\_round' matrix, calculate the matrix:

1 - 'Num\_Stations\_in\_Between' - Calculate the number of stations between station i station j

2 - 'Time\_Between\_Stations' - Calculate the ride-time between station i station j \*/

/\* Before running this part - delete '/\*' from part B , and add for Part A \*/

int B\_round[stations\_index][stations\_index] = ...;

float t[stations\_index] = ...;

dvar int Num\_Stations\_in\_Between[stations\_index ,stations\_index];

dvar float Time\_Between\_Stations[stations\_index ,stations\_index];

minimize 0;

subject to{

forall (o in stations\_index, d in stations\_index)

Num\_Stations\_in\_Between[o,d]==((1-B\_round[o,d])\*(d-o)+(B\_round[o,d])\*(S-o+d));

forall (o in stations\_index, d in stations\_index)

Time\_Between\_Stations[o,d]==

(1-B\_round[o,d])\*(sum (i in (o..d-1)) t[i]) +

(B\_round[o,d])\*(sum (i in (o..S-1)) t[i] + sum (i in (0..d-1)) t[i]);

}

קובץ אלגוריתם – DATA

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 20.1.0.0 Data

\* Author: almog

\* Creation Date: Aug 25, 2021 at 8:33:02 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SheetConnection sheet("Small.xlsx");

alpha1 from SheetRead (sheet, "alpha1");

alpha2 from SheetRead (sheet, "alpha2");

Capacity from SheetRead (sheet, "Capacity");

L from SheetRead (sheet, "L");

n from SheetRead (sheet, "n");

S from SheetRead (sheet, "S");

t from SheetRead (sheet, "t");

V from SheetRead (sheet, "V");

requests from SheetRead (sheet, "requests");

sigma from SheetRead (sheet, "sigma");

B\_round from SheetRead (sheet, "B\_round");

Num\_Stations\_in\_Between from SheetRead (sheet, "NSiB");

Time\_Between\_Stations from SheetRead (sheet, "TBS");

קובץ אלגוריתם – MOOD

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 20.1.0.0 Model

\* Author: almog

\* Creation Date: Aug 25, 2021 at 8:18:20 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int S=...;

range stations\_index = 0..(S-1);

int n=...;

range requests\_index = 1..n;

int V=...;

range vehicles\_index = 1..V;

int L=...;

range rounds\_index = 1..L;

int alpha1=...;

int alpha2=...;

int Capacity=...;

int sigma=...;

float t[stations\_index] = ...;

int requests[requests\_index][1..5] = ...;

int B\_round[stations\_index][stations\_index] = ...;

int Num\_Stations\_in\_Between[stations\_index ,stations\_index] = ...;

float Time\_Between\_Stations[stations\_index ,stations\_index] = ...;

dvar int y[vehicles\_index];

dvar float p[requests\_index];

dvar float dr[requests\_index];

dvar int num[vehicles\_index][stations\_index][rounds\_index];

dvar boolean treatment[requests\_index][vehicles\_index][rounds\_index];

dvar boolean B\_Stop[requests\_index][vehicles\_index][stations\_index][rounds\_index];

minimize alpha1\*(sum (k in vehicles\_index)y[k]) + alpha2\*(sum(i in requests\_index) requests[i][1]\*dr[i]) ;

subject to {

sum (i in requests\_index, k in vehicles\_index,r in rounds\_index) treatment[i][k][r]== n ; //Every request is served

forall (i in requests\_index) (sum (k in vehicles\_index ,r in rounds\_index) treatment[i][k][r]) == 1 ; // //Every request is served exactly once

forall (i in requests\_index, k in vehicles\_index,r in rounds\_index)

treatment[i][k][r] <= B\_Stop[i][k][requests[i][2]][r] ;

forall (i in requests\_index) p[i] >= requests[i][3] ; //Make sure request pick-up time is relevant

forall (i in requests\_index) dr[i] >= requests[i][5] ; //Make sure request drop-up time is relevant

}