# Application of Linear Programming in the Orbital Tracking and Surveillance field

Paolo Tognola - Davide Zamblera

POLITECNICO
MILANO 1863

**Course of Operations Research**
**Academic Year 2021-2022**

# Introduction

When observing space objects there is the necessity of observing multiple objects in a certain limited amount of time. Multiple sessions are not always available because of either lack of time or excessive cost.

In these cases we might need to only look at a certain subset of all available objects while only choosing the ones that respect our objectives.

These objectives can range from observing certain objects for the longest possible sum of arc of orbits to minimizing the variance between the measured arc lengths, in order to avoid wasting resources on very small arcs that wouldn't be able to give us accurate results.

The model should also account for the fact that objects have a rise time and a set time, which depends on the particular object.

*This report mostly refers to the following paper:* ***The application of linear programming on the space surveillance of high-altitude objects - Xin Wang, Jianning Xiong, Hongbo Wang, Wei Zhang - Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008, China***

# Lambert's theorem

In celestial mechanics, Lambert's problem is concerned with the determination of an orbit from two position vectors and the time of flight, posed in the 18th century by Johann Heinrich Lambert and formally solved with mathematical proof by Joseph-Louis Lagrange.

Direct consequence of this problem and its theorem is that an object orbit can be characterized by only observing it two times and by measuring the time that passes between the two observations.

But as the reported observed positions may not be perfect and may have some errors, it's in our best interest to have the initial and final positions as far away from each other (by maximizing the arc length) to be able to improve the accuracy of the resulting orbit.

# Mathematical programming model

Both models have the same objective: to maximize the total arc length observed during one session given a scarce resource, the visibility of the objects, as they rise and set in different time periods. The constraints are mostly the same:

- It's possible to only observe one object at a time (limitation of the observing station).
- If an object is observed, it must be observed 2 times: one time to characterize the initial position and one time to characterize the final position. Other observation would only waste useful time slots, as only two positions are needed (consequence of the Lambert's theorem).

All the other additional constraints are trivial and related to the coherency of the problem (e.g. an object's initial observation must happen before the final one). Those will be better specified in each model.

It is assumed that for every observation a certain amount of time is needed (e.g. 3 minutes) to switch from an object to another and to gather data. Since the total observation window is limited (e.g. one night) then the whole observation window can be divided into N periods, during which only one object can be observed.

The problem ultimately becomes which object should be selected for every available time window to maximize our objective, the total arc length.

The data to us available are the Visibility matrix (formulated in a different way for each model) and the following sets:

- OBJ := [1..TS]: all of the possible objects that could be observed, where TS is the last visible object in our dataset.
- TIME := [1..TP]: all of the possible short time periods during which an object could be observed, where TP is the last available time period.

# Model 1

The chosen variables $Y_{i,j}$ are defined as BINARY. They are divided into the ones relative to the first and the last observation as they are the most important time frames.

$$YS_{i,j} = \begin{cases} 1, & \text{if object (i) is observed for the } \textbf{first time} \text{ during time period (j)} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$YE_{i,j} = \begin{cases} 1, & \text{if object (i) is observed for the } \textbf{last time} \text{ during time period (j)} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

There is only one parameter of interest in this problem, the so called Visibility matrix $V_{i,j}$, an intrinsic property of each object.

$$V_{i,j} = \begin{cases} 1, & \text{if object (i) is } \textbf{visible} \text{ during time period (j)} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

**Notice that an object could be visible but not be observed during a certain time frame.** For each observed object it is trivial then that the observed arc length, if present, is defined as the difference between the final and initial time slots. If we sum each individual observed arc for every object we find the total arc length to be maximized:

$$maximize \quad \sum_{i=1}^{TS}\sum_{j=1}^{TP}(j \times YE_{i,j} - (j-1) \times YS_{i,j}) \tag{4}$$

The function is subject to the following restrictions. First off, there can be at most only one starting observation window (and only one final one) for every object.

$$s.t. \quad \sum_{j=1}^{TP} YS_{i,j} \leq 1 \quad \forall\, i \in [1..TS] \tag{5}$$

$$s.t. \quad \sum_{j=1}^{TP} YE_{i,j} \leq 1 \quad \forall\, i \in [1..TS] \tag{6}$$

Obviously the final observation period must be temporally placed after the initial one. This can be formulated like this:

$$s.t. \quad \sum_{j=1}^{TP} j \times (YE_{i,j} - YS_{i,j}) \geq 0 \quad \forall\, i \in [1..TS] \tag{7}$$

For consistency's sake, if an object has a first observation period it also must have a last one (and if one is lacking it must have no one).

$$s.t. \quad \sum_{j=1}^{TP} (YE_{i,j} - YS_{i,j}) = 0 \quad \forall\, i \in [1..TS] \tag{8}$$

In every time period only one object can be observed.

$$s.t. \quad \sum_{i=1}^{TS} (YE_{i,j} + YS_{i,j}) \leq 1 \quad \forall\, j \in [1..TP] \tag{9}$$

Lastly, we can formulate the visibility constraint, the one that let's us adapt the model to a certain visibility matrix.

$$s.t. \quad YS_{i,j} + YE_{i,j} \leq V_{i,j} \quad \forall\, i \in [1..TS] \quad \forall\, j \in [1..TP] \tag{10}$$

We have placed all the necessary constraints to properly formulate the model.

# Model 2

*We changed the indexing suggested in the paper as it was easier to analyze the results in MATLAB.*

In the second model only one variable is chosen to define the initial and final observations of the object, $Y_{j,k,i}$.

$$Y_{j,k,i} = \begin{cases} 1, & \text{if object (i) is observed } \textbf{first} \text{ during time period (j) and } \textbf{last} \text{ during period (k)} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The visibility matrix is defined as:

$$V_{j,k,i} = \begin{cases} 1, & \text{if object (i) is visible during time period (j) and period (k)} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Obviously, we can only select one element per "plane", as we observe an object's arc only one time.

$$s.t. \quad \sum_{j=1}^{TP}\sum_{k=1}^{TP} Y_{j,k,i} = 1 \quad \forall\ i \in [1..TS] \tag{13}$$

For temporal continuity reasons, as t.e final observation must take place after the initial one.
*With respect to the paper the sign in equation 14 was changed from "less" to "less or equal" as one cannot start and finish the observation in the same time period. Obviously the indexing in equation 15 was also changed.*

$$Y_{j,k,i} = 0 \quad if\ k \leq j, \quad \forall\ i \in [1..TS]. \tag{14}$$

This can be translated into mathematical programming language as:

$$s.t. \quad \sum_{i=1}^{TS}\sum_{j=2}^{TP}\sum_{k=1}^{j} Y_{j,k,i} = 0 \tag{15}$$

Only one object can be observed at the same time.

$$s.t. \quad \sum_{i=1}^{TS} \times \left( \sum_{k=1}^{j} Y_{k,j,i} + \sum_{k=j+1}^{TP} Y_{j,k,i} \right) \leq 1 \quad \forall\ j \in [1..TP] \tag{16}$$

The observations need to comply with the visibility matrix.

$$s.t. \quad Y_{j,k,i} - V_{j,k,i} \leq 0 \quad \forall\ i \in [1..TS] \quad \forall\ j \in [1..TP] \quad \forall\ k \in [1..TP] \tag{17}$$

The total arc length to be optimized would be:

$$maximize \quad \sum_{i=1}^{TS}\sum_{j=1}^{TP}\sum_{k=1}^{TP}(k-j+1) \times Y_{j,k,i} \tag{18}$$

But built like this the model would serve no other purpose than to be a copy of the first one. We can insert a new parameter that is equal to the mean value of the single arc lengths selected by the first model and we can modify the second model to output the same objects but with an improved arc length, with less variance between each one.

$$minimize \quad \sum_{i=1}^{TS}\sum_{j=1}^{TP}\sum_{k=1}^{TP}(k-j+1-MEAN)^2 \times Y_{j,k,i} \tag{19}$$

With this modification the second model acts as a refinement tool for the first ones result. Using both models in this order would be very beneficial for big sets of data, as the first one is computationally light $O(n^2)$, and can be applied easily and the second one, more computationally heavy $O(n^3)$, can be used to refine the results.

*In equations 18 and 19 the indexes were changed as the paper made a typo error (the index relative to the object was subtracted from the one relative to the time).*

## Implementation and validation

The models were implemented in **AMPL** and **MATLAB**.

The generation of a random visibility matrix was needed because of the scarcity of data about this particular problem. For this reason and to aid post-processing of the results the MATLAB language and the associated **AMPL-MATLAB API** was used.

The 2 main files are *FIRST_ROUND.m* and *SECOND_ROUND.m*.

In the first file the the visibility matrix is generated as a TS x TP matrix with the possibility of controlling the maximum visibility range of any objects and the maximum initial time an object can appear.

Then the **AMPL** model *model1.mod* with the objective of solving for maximum total arc length is loaded and solved with the *cplex MIP solver*. The mean of the arc lengths is also computed.

The results are then saved and the data is reshaped to fit the format of the second model, and a second optimization round is then carried on by the *model2.mod* file. The observed objects initial time and final time of observation are represented by the three dimensional Y variable is optimized with the objective of minimizing the variance of the arc lengths between the objects.

Both the **AMPL** models were tried by running both of them on a simple test case with the same objective function.

***No data (.dat) files were created as the data was created directly with MATLAB and run in AMPL directly from the MATLAB environment. The results can be reproduced by installing the MATLAB-AMPL API and running the code. A fixed RNG seed was used for reproducibility reasons.***

Sources:

`https://ampl.com/resources/api/#Windows`

`https://ampl.com/api/latest/matlab/index.html#`

## Optimization results

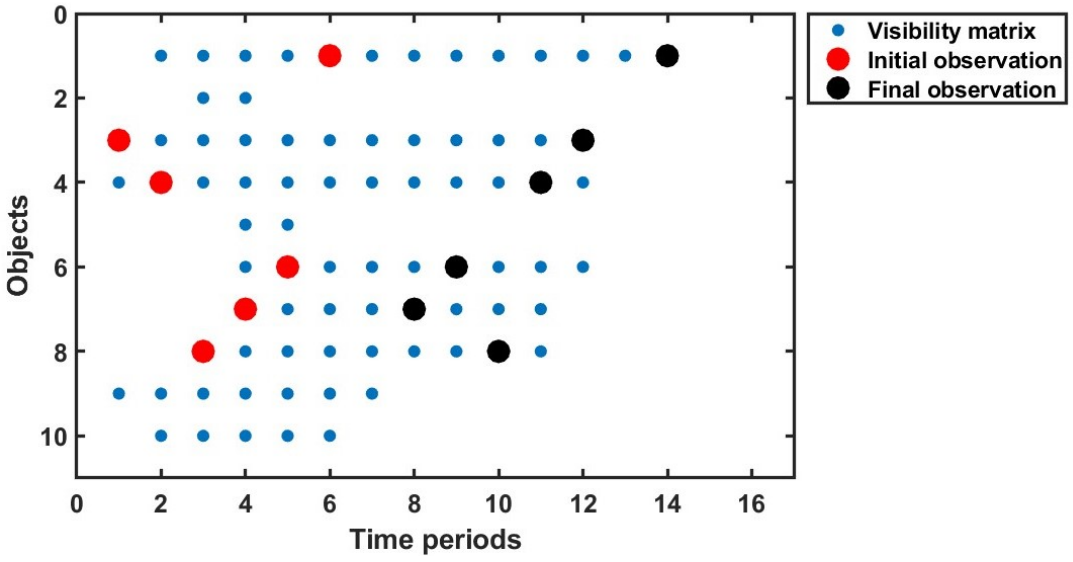


Figure 1: Results of the first round of optimization, model 1

Given a certain visibility matrix we can see that the first model was able to choose which objects should be observed and which should be avoided. Some objects were in fact not selected.

The total arc length is maximized but no attention is paid to the variance of the single ones.

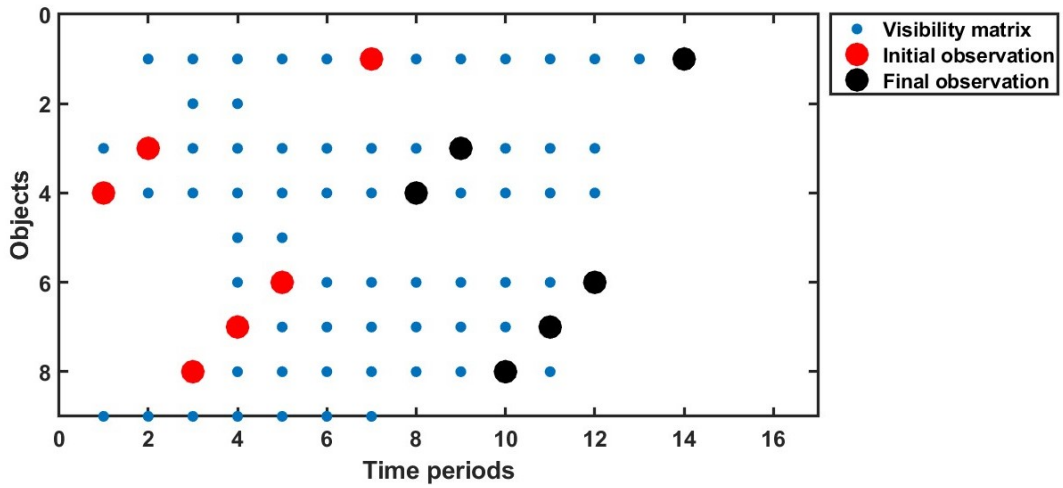


Figure 2: Results of the second round of optimization, model 2

Following a second optimization round (by using the second model) the objects are already selected, and a variance based optimization is performed on the chosen satellites. We can clearly see that the arc lengths are now more similar.

The total arc length dropped from 49 to 48, as the main objective of this optimization isn't anymore the total arc length but the minimization of the variance.

The total arc length is still good as it was the objective of the first optimization.

This is preferable as it avoids the selection of short arcs that might lead to an imprecise orbit determination.

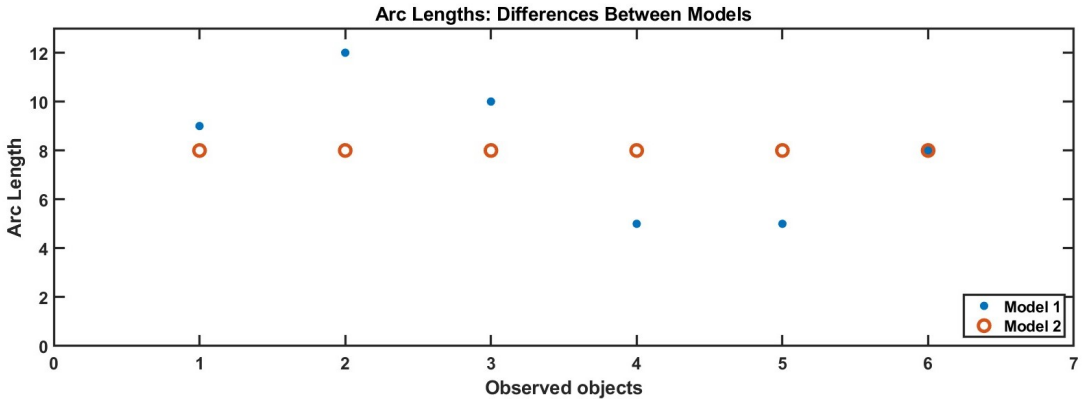


Figure 3: Relative results between first round and second round of optimization

As we can clearly see, the final chosen arcs after two rounds of optimization are very similar in length (in this case equal).

# Conclusion

The two models are working as expected and the two-round optimization is beneficial to the intended objective of the problem (the selection of the best possible arcs for multiple orbit characterizations). While this was tested on a small sample size (due to the free **AMPL** limitations) the models are general and they works. On a bigger data set a two-rounds optimization will prove very efficient computationally as the heavy variance minimization is performed only on the objects previously selected with the fast first-round optimization.

# Appendix

## MATLAB code

### Generating visibility matrix for model 1

```
function V = generate_sat_data(TS, TP, init_range, span_range)
%GENERATE_SAT_DATA generate data for the high_alt_obj_opt.mod
% PROTOTYPE:
%     V = generate_sat_data(TS, TP, init_range, span_range)
%
% INPUT:
%    TS[1]                    number of satellites [-]
%    TP[1]                    number of periods [-]
%    init_range[1]            % of the period in which a initial visibility happen[-]
%    span_range[1]            % of the period which can be visible at max[-]
%
% OUTPUT:
%    V[TS,TP]                 visibility matrix [-]
%
% CONTRIBUTORS:
%   Davide Zamblera - Politecnico di Milano - Space eng.
%
% VERSIONS
% 2022-09-10: First version
%

if nargin < 3
    init_range = 0.3;
    span_range = 0.7;

end
max_init = ceil(TP*init_range);
max_span = ceil(TP*span_range);

rng(17);
row_start = randi(max_init,TS,1);
row_span = randi(max_span,TS,1);
row_end = row_start + row_span;
row_end(row_end > TP) = TP;

V = zeros(TS,TP);
for j = 1:TS
    V(j,row_start(j):row_end(j)) = ones(1,row_end(j)-row_start(j)+1);
end


end
```

### Generating visibility matrix for model 2

```
function V3D = generate_visibility_3d(V)
%GENERATE_VISIBILITY_3D Summary of this function goes here
%   Detailed explanation goes here
[n,m] = size(V);
```

```
V3D = zeros(m,m,n);
for i = 1:n
    ind_i = find(V(i,:) == 1);
    V3D(ind_i,ind_i,i) = 1;
end
```

## Runs the first model

```
clear;
close all;
clc;

%% Initiate Engine and Load Model
setupOnce;
ampl = AMPL;

ampl.read('model1.mod');
%% Generate Data
TS = 5;
TP = 16;
OBJ = cell(1,TS);
%OBJ(:) = {'SAT'};
for i = 1:TS
    %OBJ{i} = OBJ{i} + string(i);
    OBJ{i} = string(i);
end
TIME = cell(1,TP);
for i = 1:TP
    TIME{i} = string(i);
end

initSets(ampl,{'OBJ','TIME'},OBJ,TIME);
for i = 1:TP
    TIME{i} = i;
end
for i = 1:TS
    OBJ{i} = i;
end

V = generate_sat_data(TS,TP);
spy(V);
hold on;
df = DataFrame(2,'OBJ','TIME','V');
df.setMatrix(V,OBJ,TIME);
df.toTable();
% Set the values to the parameter
ampl.setData(df);

ampl.display('V') % Check that assignment worked correctly (delete ; to show)

%% Solve Problem
ampl.setOption('solver', 'cplex');
ampl.solve;

%% Post Processing
YS = ampl.getVariable('YS');
YE = ampl.getVariable('YE');
df_YS = table2array(YS.getValues.toTable());
df_YE = table2array(YE.getValues.toTable());


index_S = [];
index_E = [];
tol = 1e-3;
for i = 1:size(df_YE,1)
    if abs(df_YS{i,3} - 1) < tol
        index_S = [index_S, i];
    end
    if abs(df_YE{i,3} - 1) < tol
        index_E = [index_E, i];
```

```
    end
end

VsolB = zeros(TS,TP)';
VsolB(index_S) = 1;
VsolB = VsolB';
VsolE = zeros(TS,TP)';
VsolE(index_E) = 1;
VsolE = VsolE';
spy(VsolB,'ro');
spy(VsolE,'ko');
VsolOLD = VsolB + VsolE;

chosen_sats = [df_YE{[df_YE{:,3}] == 1,1}];
n_chosen = length(chosen_sats);

MEAN = 0;
for i = 1:n_chosen
    min_ind = find(VsolB(chosen_sats(i),:) == 1);
    max_ind = find(VsolE(chosen_sats(i),:) == 1);
    MEAN = MEAN + (max_ind - min_ind + 1);
end
mean = MEAN/n_chosen;

save('first_round','V','TP','TS','n_chosen','chosen_sats','mean','VsolOLD');

ampl.close;
```

## Runs the second model

```
clear;
close all;
clc;

%% Initiate Engine and Load Model
setupOnce;
ampl = AMPL;

ampl.read('model1.mod');
%% Generate Data
TS = 5;
TP = 16;
OBJ = cell(1,TS);
%OBJ(:) = {'SAT'};
for i = 1:TS
    %OBJ{i} = OBJ{i} + string(i);
    OBJ{i} = string(i);
end
TIME = cell(1,TP);
for i = 1:TP
    TIME{i} = string(i);
end

initSets(ampl,{'OBJ','TIME'},OBJ,TIME);
for i = 1:TP
    TIME{i} = i;
end
for i = 1:TS
    OBJ{i} = i;
end

V = generate_sat_data(TS,TP);
spy(V);
hold on;
df = DataFrame(2,'OBJ','TIME','V');
df.setMatrix(V,OBJ,TIME);
df.toTable();
% Set the values to the parameter
ampl.setData(df);

```

```
ampl.display('V') % Check that assignment worked correctly (delete ; to show)

%% Solve Problem
ampl.setOption('solver', 'cplex');
ampl.solve;

%% Post Processing
YS = ampl.getVariable('YS');
YE = ampl.getVariable('YE');
df_YS = table2array(YS.getValues.toTable());
df_YE = table2array(YE.getValues.toTable());


index_S = [];
index_E = [];
tol = 1e-3;
for i = 1:size(df_YE,1)
    if abs(df_YS{i,3} - 1) < tol
        index_S = [index_S, i];
    end
    if abs(df_YE{i,3} - 1) < tol
        index_E = [index_E, i];
    end
end

VsolB = zeros(TS,TP)';
VsolB(index_S) = 1;
VsolB = VsolB';
VsolE = zeros(TS,TP)';
VsolE(index_E) = 1;
VsolE = VsolE';
spy(VsolB,'ro');
spy(VsolE,'ko');
VsolOLD = VsolB + VsolE;

chosen_sats = [df_YE{[df_YE{:,3}] == 1,1}];
n_chosen = length(chosen_sats);

MEAN = 0;
for i = 1:n_chosen
    min_ind = find(VsolB(chosen_sats(i),:) == 1);
    max_ind = find(VsolE(chosen_sats(i),:) == 1);
    MEAN = MEAN + (max_ind - min_ind + 1);
end
mean = MEAN/n_chosen;

save('first_round','V','TP','TS','n_chosen','chosen_sats','mean','VsolOLD');

ampl.close;
```

## Analyses the output of the optimization

```
function displayLengthsPlot(VsolO,VsolN)
%DISPLAYLENGTHSPLOT Summary of this function goes here
%   Detailed explanation goes here
    ll_old = length_arcs(VsolO);
    ll_new = length_arcs(VsolN);
    figure;
    plot(ll_old,'.');
    hold on;
    plot(ll_new,'o');
    xlim([0, max(xlim)+1]);
    ylim([0, max(ylim)+1]);
    curtick = get(gca, 'xTick');
    xticks(unique(round(curtick)));
    title('Arc Lengths: Differences Between Models')
    xlabel('Satellites')
    ylabel('Arc Length')
    legend({'Model 1','Model 2'},'Location','southeast')
end
```

```
function ll = length_arcs(V)
    ind = find(V');
    ll = diff(ind);
    ll = ll(1:2:end)+1;
end
```

# AMPL code

## Model 1

```
reset;
set OBJ;
set TIME;

var YS {OBJ, TIME} binary;
var YE {OBJ, TIME} binary;

param V {OBJ, TIME};

maximize TotalArcLength:
sum{i in OBJ, j in TIME} (j*YE[i,j] - (j - 1)*YS[i,j]);

subject to UniqueStart {i in OBJ}:
sum {j in TIME} YS[i,j] <= 1;

subject to UniqueEnd {i in OBJ}:
sum {j in TIME} YE[i,j] <= 1;

subject to Consecutivity {i in OBJ}:
sum {j in TIME} j*(YE[i,j] - YS[i,j]) >= 0;

subject to ObservationAgreement {i in OBJ}:
sum {j in TIME} (YE[i,j] - YS[i,j]) = 0;

subject to ObjectLimit {j in TIME}:
sum {i in OBJ} (YE[i,j] + YS[i,j]) <= 1;

subject to ObjectVisibility {i in OBJ, j in TIME}:
YE[i,j] + YS[i,j] <= V[i,j];
```

## Model 2

```
reset;
set OBJ;
set TIME;

var Y {TIME, TIME, OBJ} binary;

param V {TIME, TIME, OBJ};
param n; #final element of time
param MEAN; #mean of observed satellite arcs of model I


minimize Variance:
sum {i in OBJ, j in TIME, k in TIME} ((k-j+1-MEAN)^2)*Y[j,k,i];

subject to PlaceUnicity {i in OBJ}:
sum {j in TIME, k in TIME} Y[j,k,i] = 1;
```

```
subject to Consecutivity:
sum {i in OBJ, j in 2..n, k in 1..j} Y[j,k,i] = 0;

subject to OneObjectAtATime {j in TIME}:
sum {i in OBJ} (sum {k in 1..j} Y[k,j,i] + sum {k in j+1..n} Y[j,k,i]) <= 1;

subject to Visibility {i in OBJ, j in TIME, k in TIME}:
    Y[j,k,i] <= V[j,k,i];
```