Project of Operations Research

Prof. Maurizio Bruglieri

Academic year 2021/2022

Solution of some instances of the Flying Sidekick Traveling Salesman Problem

---

Giorgio Lerario, 10800092

Contents

1 Introduction

The potential role that drones can play in the transportation and logistics field has been under intense study over the past decade, during which many models and heuristic algorithms have been proposed to deal with freshly defined problems [1]. The aim of this project is to solve a problem based on the concept of Drone Truck Combined Operation (DTCO), the Flying Sidekick Traveling Salesman Problem (FSTSP). The model adopted is based on Mixed Integer Linear Programming (MILP) and has been introduced by Chase C. Murray and Amanda G. Chu [3].

2 Flying Sidekick Traveling Salesman Problem

The Flying Sidekick Traveling Salesman Problem is a delivery problem in which both a truck and a drone are employed. The drone is carried by the truck during its route and it can be launched from any delivery location. Furthermore, the drone is able to deliver a package and return to the truck without any human intervention. Once the drone returns to the truck, the driver will easily replace its battery making it ready for the next flight.

Both vehicles are bound to depart and end their trip from a single location and, in the problem most general formulation, the vehicles have to travel on the existing road network. A simple graph representation of the problem can be adopted: the nodes represent the locuses of the customers while the arcs represent the travel of one vehicle from one node to another (Figure 2).

The FSTSP includes the Traveling Salesman Problem as a particular case and therefore is an NP-hard problem. Its solution through branch and bound algorithm is feasible only for a fairly small number of customers.

In order to create and solve relevant instances of the FSTSP both from a numerical and realistic point of view, the following scenario is introduced.

Polimi card delivery problem

The Polytechnic School of Milan wants to deliver its student ID card to all its new students. In order to achieve this a truck and a drone are employed. The delivery area loosely corresponds to the urban territory of Milan, where most of the students are clustered. Both vehicles depart from Bovisa's campus and they are bound to return there at the end of the route. The truck and the drone are constrained to operate on Milan's road network and the objective function to be minimized is the time at which the vehicles return back to the campus.

3 MILP formulation

The formulation used for this study presents no simplifications whatsoever with respect to the reference paper [3]. The set $C = \{1, 2, ...c\}$ represents the set of all customers, while $C' \subseteq C$ denotes the set of drone eligible customers. $N_0 = \{0, 1, ..., c\}$ and $N_+ = \{1, 2, ..., c + 1\}$ represent respectively the set of nodes from which the drone may depart and may visit. The parameter c is the total number of customers, both the nodes 0 and c+1 correspond to the depot location, so that the vehicles are bound to start at node 0 and end at node c+1.

This formulation exploits the concept of sorties: since the drone has to start at a node $i \subseteq N_0$, visit a node $j \subseteq C'$ and meet the truck at a node $k \subseteq N_+$, the set of all possible combinations is represented from the set of tuples $P = \{\langle i, j, k\rangle : i \in N_0, j \in j \subseteq C', k \subseteq N_+, i \neq j, j \neq k\}$.

The parameters τ_{ij} and τ'_{ij} represent respectively the time for the truck and the time for the drone to go from i to j. Parameters s_L and s_R represent the times to prepare the drone for launch and to recover the drone after a sortie. Parameter e represents the endurance of the drone in units of time.

The decision variable $x_{ij} \in \{0, 1\}$ equals one if the truck travels from node i to node j, zero otherwise. $y_{ijk} \in \{0, 1\}$ is defined over P and it is equal one if the drone performs the sortie $i \to j \to k$, zero otherwise. The variables t_i and t'_i represent respectively the time at which the truck and the drone arrive at node i. The auxiliary variable $1 \leq u_i \leq c + 2$ specifies the position of node i in the truck path, this variable is necessary to prevent truck subtours. Another auxiliary variable is defined, $p_{ij} \in \{0, 1\}$, which equals one if node i is visited at some time before node j, zero otherwise.

The objective function to be minimized is the final time at which the truck returns to the depot t_{c+1}. The truck and the drone have to arrive at the depot at the same time, this is expressed in the model from constraints (14) and (15), that can be seen in appendix A.

An accurate description of all the constraints and the assumptions can be found in the reference paper, while the AMPL mod and dat files are presented in appendix B. Because the original statement of the problem requires the vehicles to operate on the road network the creation of an instance is not trivial and it is worth to discuss it as a problem on its own.

4 Instance generation

All the instances of the FSTSP used here have been generated through a self-developed python code. The tasks that the code accomplishes can be summarized as follows:

1. generation of random positions inside a user defined area (circle or square areas are available as default).
2. conversion of the positions into geographical (latitude/longitude) coordinates

by taking as reference one central coordinate.

3. addition of the depot coordinate and other non mandatory user defined coordinates.
4. generation of the distances and times matrixes, this is done by calling the Google Maps API [2].
5. printing the matrixes in an AMPL readable dat file.
6. printing the matrixes and coordinates of all locations in a txt file.

The geographical coordinates uniquely correspond to the nodes. The numbering starts from the depot, node 0, and then it follows the order of generation of the random positions. The use of the Google Maps API implies that the road network is always taken into account, furthermore even traffic conditions can enter the model implicitly by considering them in the generation of the times and distances matrixes. The formulas used to transpose relative positions into coordinates are presented in appendix C, while the txt output of the code can be found in appendix D.

Description of the employed instances parameters

The center of the area can be any coordinate around the world, for the present study it corresponds approximately to the geographical center of Milan. The characteristic length of the area is 14 kilometers. The depot location, as stated before, is fixed for all instances and corresponds to Polimi Bovisa's campus. The total number of customers chosen for the instances is nine so that the total number of nodes is ten (eleven if the depot is considered two times). All customers are considered drone eligible, the time parameters s_L and s_R are 1 minute each. The truck speed and the drone speed chosen are the ones in the reference paper, 25 mph (670.6 m/min) and 35 mph (938.6 m/min), the endurances tested are many and are described in the results.

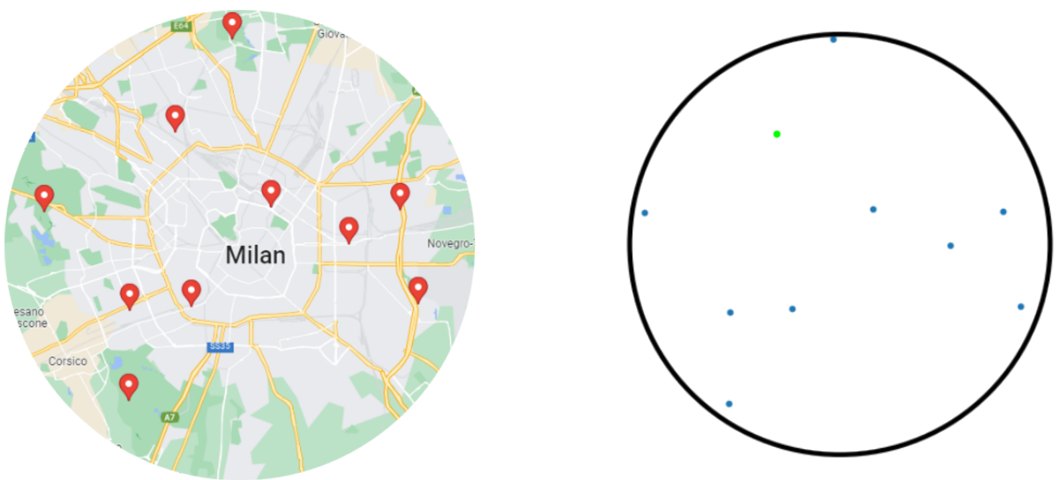

Figure 1: Plot of the positions on the right and their transposition into google maps on the left. The green point is the depot coordinate.

5 Results

The MILP formulation has been implemented in AMPL and can be found in appendix B. Since the number of variables and constraints, for nine customers, exceeds by far the maximum number set for the AMPL free version, the solutions have been computed with the one month trial student version of AMPL. The solver adopted is CPLEX. The running time for all the instances is in a range of 5-20 minutes and, as an observation, it increases if the endurance parameter increases.

The representations of the solutions in Figure 2 and 3 are reliable in terms of position of the nodes, but it must be emphasized that the arcs do not show the real path between one node to another since in reality both drone and truck work on the road network.

Solution reconstruction method and representation

The solution pictures are constructed by taking the reference coordinates, that are an output of the Python code (appendix D), and positioning them on the reference area. Then, by looking at the output solution of the instance taken from AMPL (appendix E), the route of the truck can be retrieved by looking at variable x starting from 0 and ending to node 10:

```
x [*,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   0   0   0   0   1   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   1   0   0   0   0   0   0
4   0   0   0   0   0   0   1   0   0   0
5   0   0   0   0   0   0   0   0   0   1
6   0   0   0   0   1   0   0   0   0   0
7   0   0   0   0   0   1   0   0   0   0
8   0   0   0   0   0   0   0   0   1   0
9   0   0   1   0   0   0   0   0   0   0
;
```

Then, for the generic node j that is not visited by the truck, the matrix $y[*, j, *]$ can be consulted, the element different from zero of said matrix will have as row i the starting node of the drone and as column k the ending node of the drone:

```
y [*,1,*]
:   1   2   3   4   5   6   7   8   9   10
0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   1
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0
```

```
 [*,2,*]
:   1   2   3   4   5   6   7   8   9   10
0   0   0   0   1   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   0
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0
```

in this way the sortie $\langle i, j, k \rangle$ performed by the drone is retrieved.

First test configuration

For this configuration four drone endurances have been tested: 0, 12, 15 and 25 minutes. The zero endurance case is perfectly equivalent to a Traveling Salesman Problem. Increasing the endurance the objective function, expressed in minutes, decreases and the number of drone sorties increases as it should from intuition. This case is very sensitive to an endurance variation: varying it just of 3 minutes, from twelve to fifteen, results in a relevant improvement of the total time. The objective function, from zero to twenty-five minutes endurance, improves of -29%.

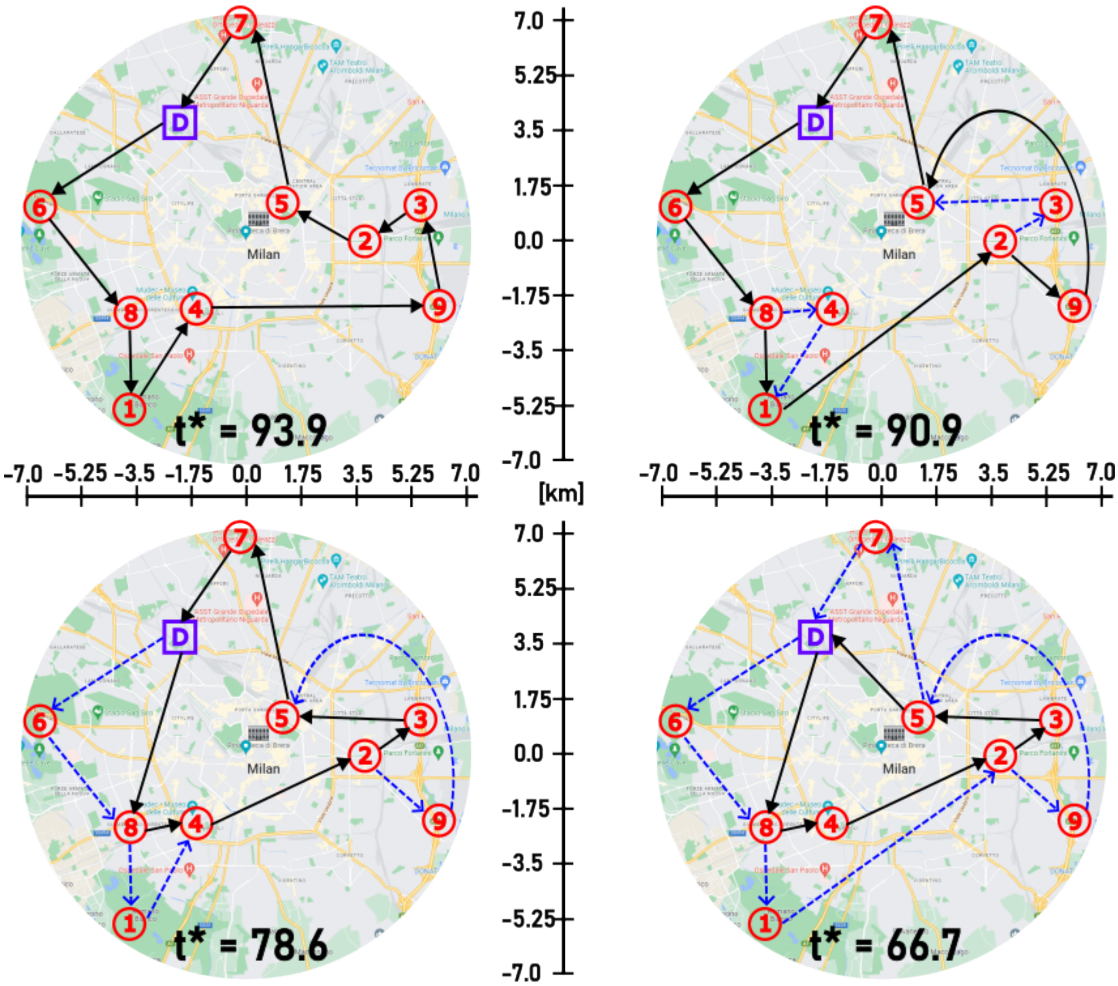

Figure 2: First test configuration solutions, from left to right and from top to bottom the endurance increases.

Second test configuration

Also for this configuration four drone endurances have been tested: 0, 15, 25 and 60 minutes. From the solutions it appears clearly that this particular configuration is quite less sensitive to endurance variation with respect to the first one: from zero to sixty minutes of endurance the optimal solution improves just of -15%, 18 minutes in total.

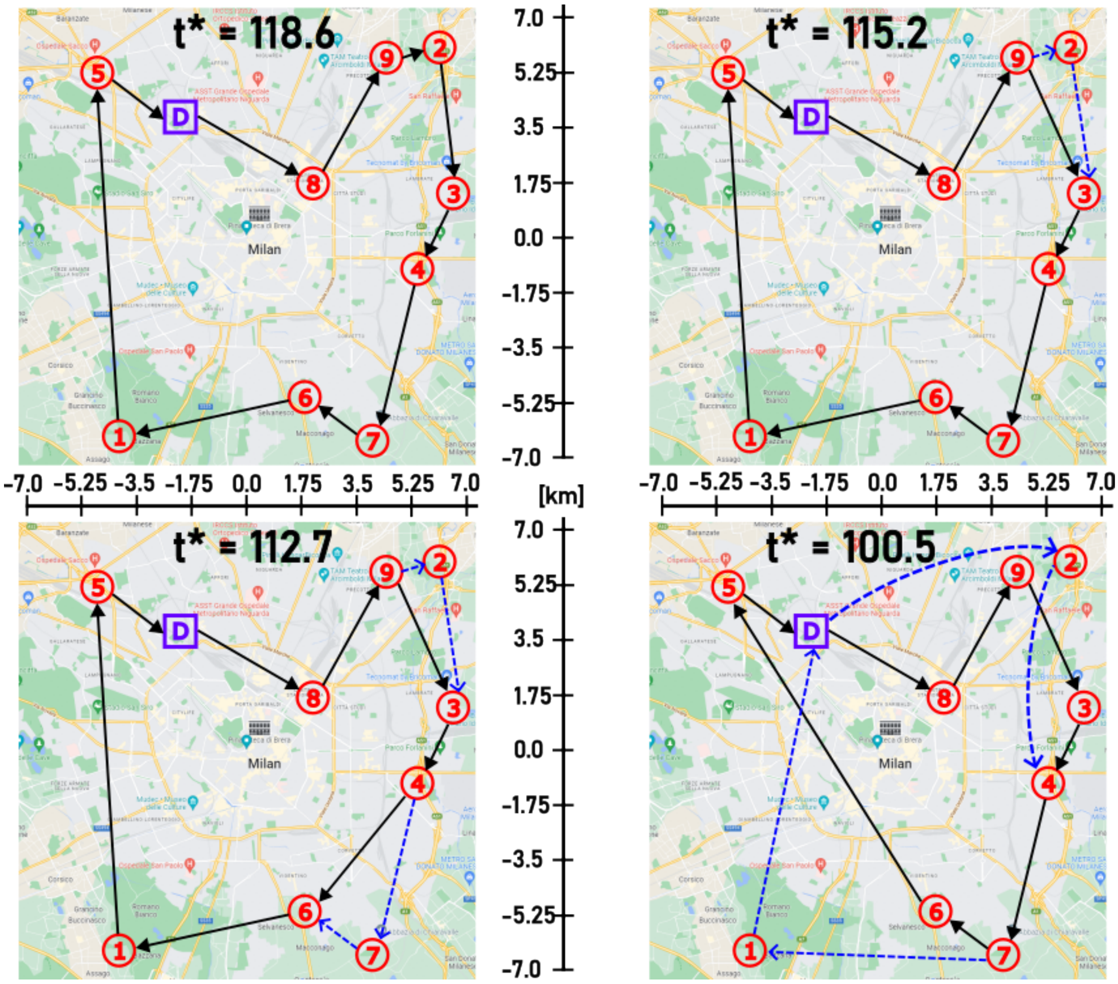

Figure 3: Configurations and total number of panels.

Traffic influence on solution

For all the previous instances the time to get from one node to another (τ and τ') has been calculated as the distance between the nodes, obtained with the Google Maps API, over the speed of the vehicle considered. In this last test only τ' has been computed as distance over velocity, τ on the other hand corresponds to the times matrix, which comes directly from Google Maps and considers the real estimated velocity of the truck and the actual traffic conditions. The departure time is set at 8:00 AM of the 3^{rd} of August 2022 and the endurance of the drone is set to 60 minutes.

The values of the optimal solutions should be compared carefully, since the speed of the truck is different between the two cases and also the routes between couples of nodes found by Google Maps are different. Anyway, it appears clearly from Figure 4 that the use of the drone, assumed it has the green lights to fly over the streets of the city avoiding traffic, can be much helpful in a real city with realistic traffic conditions.

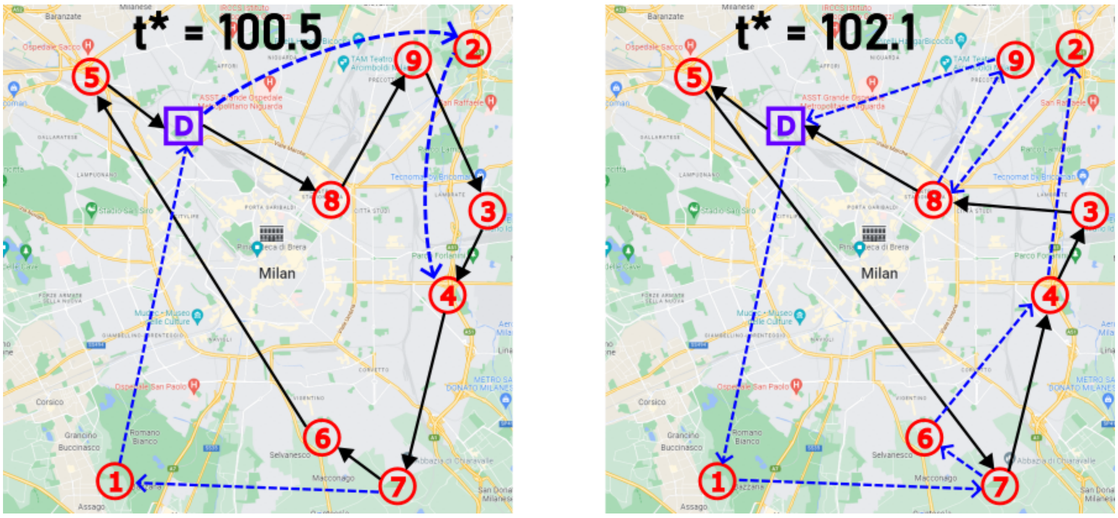

Figure 4: Comparison between the second test configuration and the test with traffic conditions. The nodes coordinates are the same. The endurance is 60 minutes for both cases.

6 Conclusions

The AMPL language has proven to be flexible, fast to code and easy to use even for complex formulations. The use of a drone in simple delivery operations can greatly decrease the total time of the route, as proven with the test problems here, and it can also bring advantages in terms of fuel consumption since the efficiency of a drone is superior to that of the truck. For this reason, the study of the FSTSP problem and in general of DTCO problems assumes an important value for industries and universities all around the world.

The use of geolocation services, such as Google Maps, can act as an important link between a simple test case and a more realistic one. In the present case geolocalization services have proven to be reliable and easy to use.

References

[1] Sung Chung, Bhawesh Sah, and Jinkun Lee. "Optimization for Drone and Drone-truck Combined Operations: A Review of the State of the Art and Future Directions". In: *Computers Operations Research* 123 (June 2020), p. 105004. DOI: 10.1016/j.cor.2020.105004.

[2] Google Inc. *Python Client for Google Maps Services*. URL: https://github.com/googlemaps/google-maps-services-python.

[3] Chase Murray and Amanda Chu. "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54 (May 2015). DOI: 10.1016/j.trc.2015.03.005.

[4] Wikipedia. *Mercator projection*. URL: https://en.wikipedia.org/wiki/Mercator_projection.

A MILP formulation

$$\text{Min } t_{c+1} \tag{1}$$

$$\text{s.t.} \sum_{\substack{i\in N_0\\ i\neq j}} x_{ij} + \sum_{\substack{i\in N_0\\ i\neq j}} \sum_{\substack{k\in N_+\\ \langle i,j,k\rangle \in P}} y_{ijk} = 1 \quad \forall j \in C \tag{2}$$

$$\sum_{j\in N_+} x_{0j} = 1 \tag{3}$$

$$\sum_{i\in N_0} x_{i,c+1} = 1 \tag{4}$$

$$u_i - u_j + 1 \leqslant (c+2)\,(1 - x_{ij}) \quad \forall i \in C, j \in \{N_+ : j \neq i\} \tag{5}$$

$$\sum_{\substack{i\in N_0\\ i\neq j}} x_{ij} = \sum_{\substack{k\in N_+\\ k\neq j}} x_{jk} \quad \forall j \in C \tag{6}$$

$$\sum_{\substack{j\in C\\ j\neq i}} \sum_{\substack{k\in N_+\\ \langle i,j,k\rangle \in P}} y_{ijk} \leqslant 1 \quad \forall i \in N_0 \tag{7}$$

$$\sum_{\substack{i\in N_0\\ i\neq k}} \sum_{\substack{j\in C\\ \langle i,j,k\rangle \in P}} y_{ijk} \leqslant 1 \quad \forall k \in N_+ \tag{8}$$

$$2y_{ijk} \leqslant \sum_{\substack{h\in N_0\\ h\neq k}} x_{hi} + \sum_{\substack{l\in C\\ l\neq k}} x_{lk}$$

$$\forall i \in C, j \in \{C : j \neq i\}, k \in \{N_+ : \langle i,j,k\rangle \in P\} \tag{9}$$

$$y_{0jk} \leqslant \sum_{\substack{h\in N_0\\ h\neq k}} x_{hk} \quad \forall j \in C, k \in \{N_+ : \langle 0,j,k\rangle \in P\} \tag{10}$$

$$u_k - u_i \geqslant 1 - (c+2)\left(1 - \sum_{\substack{j\in C\\ \langle i,j,k\rangle \in P}} y_{ijk}\right)$$

$$\forall i \in C, k \in \{N_+ : k \neq i\} \tag{11}$$

$$t'_{\mathrm{i}} \geqslant t_{\mathrm{i}} - M\left(1 - \sum_{\substack{j\in C\\ j\neq i}} \sum_{\substack{k\in N_+\\ \langle i,j,k\rangle \in P}} y_{ijk}\right) \quad \forall i \in C \tag{12}$$

$$t'_i \leqslant t_i + M\left(1 - \sum_{\substack{j\in C\\ j\neq i}} \sum_{\substack{k\in N_+\\ \langle i,j,k\rangle \in P}} y_{ijk}\right) \quad \forall i \in C \tag{13}$$

$$\text{s.t. } t'_k \geqslant t_k - M\left(1 - \sum_{\substack{i\in N_0\\ i\neq k}} \sum_{\substack{j\in C\\ \langle i,j,k\rangle\in P}} y_{ijk}\right) \quad \forall k \in N_+ \tag{14}$$

$$t'_k \leqslant t_k + M\left(1 - \sum_{\substack{i\in N_0\\ i\neq k}} \sum_{\substack{j\in C\\ \langle i,j,k\rangle\in P}} y_{ijk}\right) \quad \forall k \in N_+ \tag{15}$$

$$t_k \geqslant t_h + \tau_{hk} + s_L\left(\sum_{\substack{l\in C\\ l\neq k}} \sum_{\substack{m\in N_+\\ \langle k,l,m\rangle\in P}} y_{klm}\right) + s_R\left(\sum_{\substack{i\in N_0\\ i\neq k}} \sum_{\substack{j\in C\\ \langle i,j,k\rangle\in P}} y_{ijk}\right) - M\left(1 - x_{hk}\right)$$

$$\forall h \in N_0, k \in \{N_+ : k \neq h\} \tag{16}$$

$$t'_j \geqslant t'_i + \tau'_{ij} - M\left(1 - \sum_{\substack{k\in N_+\\ \langle i,j,k\rangle\in P}} y_{ijk}\right) \quad \forall j \in C', i \in \{N_0 : i \neq j\} \tag{17}$$

$$t'_k \geqslant t'_j + \tau'_{jk} + S_R - M\left(1 - \sum_{\substack{i\in N_0\\ \langle i,j,k\rangle\in P}} y_{ijk}\right)$$

$$\forall j \in C', k \in \{N_+ : k \neq j\} \tag{18}$$

$$t'_k - \left(t'_j - \tau'_{ij}\right) \leqslant e + M\left(1 - y_{ijk}\right)$$

$$\forall k \in N_+, j \in \{C : j \neq k\}, i \in \{N_0 : \langle i,j,k\rangle \in P\} \tag{19}$$

$$u_i - u_j \geqslant 1 - (c+2)p_{ij} \quad \forall i \in C, j \in \{C : j \neq i\} \tag{20}$$

$$u_i - u_j \leqslant -1 + (c+2)\left(1 - p_{ij}\right) \quad \forall i \in C, j \in \{C : j \neq i\} \tag{21}$$

$$p_{ij} + p_{ji} = 1 \quad \forall i \in C, j \in \{C : j \neq i\} \tag{22}$$

$$t'_l \geqslant t'_k - M\left(3 - \sum_{\substack{j\in C\\ \langle i,j,k\rangle\in P\\ j\neq l}} y_{ijk} - \sum_{\substack{m\in C\\ m\neq i\\ m\neq k\\ m\neq l}} \sum_{\substack{n\in N_+\\ \langle l,m,n\rangle\in P\\ n\neq i\\ n\neq k}} y_{lmn} - p_{il}\right)$$

$$\forall i \in N_0, k \in \{N_+ : k \neq i\}, l \in \{C : l \neq i, l \neq k\} \tag{23}$$

$$t_0 = 0 \tag{24}$$

$$t'_0 = 0 \tag{25}$$

$$p_{0j} = 1 \quad \forall j \in C \tag{26}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in N_0, j \in \{N_+ : j \neq i\} \tag{27}$$

$$\begin{aligned}
\text{s.t. } & y_{ijk} \in \{0,1\} \quad \forall i \in N_0, j \in \{C : j \neq i\}, k \in \{N_+ : \langle i,j,k \rangle \in P\} && (28)\\
& 1 \leqslant u_i \leqslant c+2 \quad \forall i \in N_+ && (29)\\
& t_i \geqslant 0 \quad \forall i \in N && (30)\\
& t'_i \geqslant 0 \quad \forall i \in N && (31)\\
& p_{ij} \in \{0,1\} \quad \forall i \in N_0, j \in \{C : j \neq i\}. && (32)
\end{aligned}$$

B AMPL files

FSTSP.mod

```
param c;    #total number of customers
param c1;   #total number of drone eligible customers

param M;    #large enough scalar

set C := 1..c;
set C1 := 1..c; #all customers are drone eligible

set N0 := 0..c;
set Nplus := 1..c+1;

set P within {N0,C1,Nplus} = {i in N0,j in C1, k in Nplus: i!=j and j!= k};

param sl;   #time to launch drone
param sr;   #time to recover drone

param e;    #drone endurance

#parameters not present in the model but useful
param truck_speed;
param drone_speed;
param distances{N0,Nplus};

param tau{i in N0, j in Nplus} = (1/truck_speed) * distances[i,j];
param tau1{i in N0, j in Nplus} = (1/drone_speed) * distances[i,j];


#set of tuples a drone can perform

var x{N0,Nplus} binary;
var y{N0,C1,Nplus} binary;

var t{0..c+1} >= 0; #time at wich the truck arrives at node j
var t1{0..c+1} >= 0;    #time at wich the drone arrives at node j

var p{i in 0..c,j in C: j!=i} integer;
var u{Nplus} integer >=1 <=c+2;

minimize total_time: t[c+1];

s.t. one_visit{j in C}: (sum{i in N0: i!=j} x[i,j] + sum{(i,j,k) in P} y[i,j,k]) = 1;
s.t. start_truck: sum{j in Nplus} x[0,j] = 1;
s.t. end_truck: sum{i in N0} x[i,c+1] = 1;
s.t. prevent_subtours{i in C, j in Nplus: i!=j}: u[i] - u[j] + 1 <= (c + 2)*(1 - x[i,j]);
#if truck arrives at location j then it has to start from there.
#If it doesn't arrive at j it can't arrive at another node from there.
s.t. path_consistence{j in C}: sum{i in N0} x[i,j] = sum{k in Nplus} x[j,k];
s.t. drone_start{i in N0}: sum{(i,j,k) in P} y[i,j,k] <=1;
s.t. drone_end{k in Nplus}: sum{(i,j,k) in P} y[i,j,k] <=1;
```

```
s.t. truck_drone_consistency{i in C, j in C, k in Nplus: k!=j and j!=i}:
                                                 2*y[i,j,k] <=
                                                 sum{h in N0: h!=i} x[h,i] +
                                                 sum{l in N0: l!=k} x[l,k];
s.t. drone_start_2{j in C, k in Nplus: j!=k}:
                                                 y[0,j,k] <=
                                                 sum{h in N0: h!=k} x[h,k];
s.t. drone_visit_order{i in C, k in Nplus: i!=k}:
                                 u[k] - u[i] >=
                                 1 - (c + 2)*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. truck_drone_coordination_1{i in C}: t1[i] >= t[i] - M*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. truck_drone_coordination_2{i in C}: t1[i] <= t[i] + M*(1-sum{(i,j,k) in P} y[i,j,k]);

s.t. truck_drone_coordination_3{k in Nplus}: t1[k] >= t[k]-M*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. truck_drone_coordination_4{k in Nplus}: t1[k] <= t[k]+M*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. time_between_nodes_truck{h in N0, k in Nplus: k!=h}:
                                                 t[k] >=
                                                 t[h] + tau[h,k] + sl*(sum{(k,l,m) in P}
                                                 y[k,l,m]) +
                                                 sr*(sum{(i,j,k) in P: i!=k} y[i,j,k])
                                                 - M*(1-x[h,k]);
s.t. time_between_nodes_drone_1{j in C1, i in N0: i!=j}:
                                                 t1[j] >=
                                                 t1[i] + tau1[i,j] -
                                                 M*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. time_between_nodes_drone_2{j in C1, k in Nplus: k!=j}:
                                                 t1[k] >= t1[j] + tau1[j,k] + sr -
                                                 M*(1-sum{(i,j,k) in P} y[i,j,k]);
s.t. drone_endurance{k in Nplus, j in C, i in N0: i!=j and j!=k}:
                                                 t1[k] - (t1[j] - tau1[i,j]) <=
                                                 e + M*(1 - y[i,j,k]);
s.t. determine_p_1{i in C, j in C: j!=i}: u[i] - u[j] >= 1 - (c + 2)*p[i,j];
s.t. determine_p_2{i in C, j in C: j!=i}: u[i] - u[j] <= -1 + (c + 2)*(1 - p[i,j]);
s.t. determine_p_3{i in C, j in C: j!=i}: p[i,j] + p[j,i] = 1;
s.t. prevent_early_launch{i in N0, k in Nplus, l in C: l!=i and l!=k and k!=i}:
                                                 t1[l] >=
                                                 t1[k] -
                                                 M*(3 - sum{(i,j,k) in P: j!=l} y[i,j,k] -
                                                 sum{(l,m,n) in P: n!=l and m!=i and m!=k
                                                 and n!=i and n!=k} y[l,m,n] - p[i,l]);


s.t. initial_cond_1: t[0] = 0;
s.t. initial_cond_2: t1[0] = 0;
s.t. initial_cond_3{j in C}: p[0,j] = 1;
```

FSTSP_circle.dat

sample dat file of the first test problem

```
#all times parameters are in minutes,
#the velocities are in m/min, the distance matrix is in meters


param c := 9;        #number of customers
param c1 := 9;       #number of customers drone eligible
param M := 10000;    #large enough number
param e := 25;       #endurance
param sl := 1;
param sr := 1;
param truck_speed := 670.6;
param drone_speed := 938.8;


param distances: 1 2 3 4 5 6 7 8 9 10  :=
0   12423.0 9792.0  27462.0 8140.0  6190.0  8073.0  7224.0  8194.0  30590.0 0.0
1   0.0 12175.0 28820.0 6572.0  12203.0 22411.0 33365.0 5919.0  24847.0 13211.0
2   12039.0 0.0 2955.0  9090.0  3344.0  12410.0 9788.0  10418.0 4254.0  9276.0
3   28619.0 2570.0  0.0 15909.0 5914.0  38769.0 22410.0 17435.0 4748.0  28638.0
4   5855.0  8418.0  16483.0 0.0 8395.0  7070.0  17099.0 3036.0  12510.0 10100.0
5   12289.0 3795.0  6370.0  6456.0  0.0 9765.0  7679.0  7481.0  7707.0  6632.0
6   10355.0 40233.0 33848.0 6859.0  9782.0  0.0 17465.0 4996.0  36503.0 8767.0
7   36067.0 9728.0  22501.0 16545.0 7507.0  19044.0 0.0 16599.0 25630.0 7184.0
8   5430.0  10074.0 18527.0 2642.0  8658.0  5436.0  16353.0 0.0 14555.0 9238.0
9   26813.0 4367.0  4610.0  14103.0 7812.0  36964.0 24797.0 15630.0 0.0 31025.0
;
```

FSTSP_square.dat

sample dat file of the second test problem

```
#all times parameters are in minutes, the velocities are in m/min,
#the distance matrix is in meters


param c := 9;        #number of customers
param c1 := 9;       #number of customers drone eligible
param M := 10000;    #large enough number
param e := 60;       #endurance
param sl := 1;
param sr := 1;
param truck_speed := 670.6;
param drone_speed := 938.8;


param distances: 1 2 3 4 5 6 7 8 9 10  :=
0   25361.0 21025.0 27711.0 28756.0 4289.0  34409.0 33873.0 7126.0  22362.0 0.0
1   0.0 33832.0 27408.0 25783.0 14347.0 11650.0 13882.0 15780.0 33349.0 28479.0
2   32526.0 0.0 9108.0  10153.0 20927.0 19813.0 17672.0 9364.0  3564.0  22450.0
3   26271.0 9881.0  0.0 3899.0  28099.0 13559.0 11418.0 5787.0  9398.0  29623.0
4   23195.0 14187.0 7763.0  0.0 32405.0 10483.0 8342.0  8248.0  13703.0 33928.0
5   22787.0 23273.0 29958.0 31003.0 0.0 31835.0 31299.0 20172.0 24609.0 11798.0
6   10274.0 20486.0 14062.0 12436.0 14304.0 0.0 5206.0  8709.0  20002.0 34548.0
7   13402.0 20203.0 13779.0 12154.0 38421.0 4935.0  0.0 10169.0 19720.0 39945.0
8   15017.0 9493.0  5533.0  5510.0  10139.0 8309.0  10006.0 0.0 6185.0  7773.0
9   32225.0 3542.0  8807.0  9852.0  22047.0 19512.0 17371.0 6371.0  0.0 13352.0
;
```

C Position to coordinates formulas

Google Maps adopts a Mercator projection for its maps, which is is a cylindrical map projection presented by Flemish geographer and cartographer Gerardus Mercator in 1569 [4]. As a consequence of the projection, a certain area at a latitude far from the equator line is inflated. For this reason, if we want to go from relative coordinates to geographical ones, we have to correct the relative positions to make them fall into the correct real area of interest. the scale factor is a function of the latitude and have to be multiplied for the x and y coordinate values to get the correct latitude and longitude afterwards:

$$k = \frac{1}{cos(\varphi_0)}$$
$$x' = k * x$$
$$y' = k * y$$

φ_0 represents the reference latitude, taken to be $45,4685°$ for all the instances of this project. Once scaled, the coordinates are used in the formulas of latitude and longitude of the Mercator projection:

$$y_0 = R \ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi_0}{2}\right)\right]$$
$$\lambda = \lambda_0 + \frac{x'}{R}, \qquad \varphi = 2\tan^{-1}\left[\exp\left(\frac{y' - y_0}{R}\right)\right] - \frac{\pi}{2}$$

φ_0 and λ_0 indicate the reference latitude and longitude, that are respectively $45,4685°$ and $9,1827°$ for all the instances of this project. φ and λ are the coordinates of the generic point [x,y]. The radius R is the earth radius and the value used here is 6371 km.

The python function that implements all the above formulas is:

```
R = 6371.0 #earth radius [km],very high accuracy not important

def latlonfromdistance(self,x,y,lat0,lon0):

        scalefactor = 1/np.cos(np.deg2rad(lat0))
        x,y = [x*scalefactor,y*scalefactor]
        y0 = self.R*np.log(np.tan(np.pi/4 + np.deg2rad(lat0)/2))
        lon  = long0 + np.rad2deg(x/self.R)  #degrees
        lat   = np.rad2deg(2*np.arctan(np.exp((y0 + y) / self.R))
              - np.pi/2)
        coords = np.array([lat,lon]).transpose()
        return coords
```

D Python code output

output.txt

Output generated by the python code relative to the second test configuration. The coordinates are in the order latitude/longitude, the first one represents the depot location. The coordinates are expressed in degrees.

```
times
0.0 1716.0  1438.0  1623.0  1685.0  508.0   1914.0  1848.0  1195.0  1689.0
1799.0  0.0 1812.0  1455.0  1401.0  1536.0  1022.0  1030.0  2059.0  2010.0
1594.0  1886.0  0.0 739.0   801.0   1225.0  1555.0  1315.0  1244.0  560.0
1787.0  1373.0  634.0   0.0 288.0   1419.0  1042.0  802.0   945.0   832.0
1974.0  1198.0  820.0   463.0   0.0 1605.0  867.0   627.0   1301.0  1019.0
865.0   1326.0  1216.0  1401.0  1464.0  0.0 1524.0  1459.0  1517.0  1467.0
2090.0  920.0   1369.0  1012.0  958.0   1776.0  0.0 522.0   1508.0  1568.0
2381.0  1304.0  1227.0  870.0   817.0   2012.0  567.0   0.0 1549.0  1426.0
1294.0  1934.0  1090.0  940.0   1001.0  1314.0  1348.0  1475.0  0.0 1144.0
1909.0  1982.0  505.0   835.0   897.0   1439.0  1651.0  1411.0  1141.0  0.0


distances
0.0 25361.0 21025.0 27711.0 28756.0 4289.0  34409.0 33873.0 7126.0  22362.0
28479.0 0.0 33832.0 27408.0 25783.0 14347.0 11650.0 13882.0 15780.0 33349.0
22450.0 32526.0 0.0 9108.0  10153.0 20927.0 19813.0 17672.0 9364.0  3564.0
29623.0 26271.0 9881.0  0.0 3899.0  28099.0 13559.0 11418.0 5787.0  9398.0
33928.0 23195.0 14187.0 7763.0  0.0 32405.0 10483.0 8342.0  8248.0  13703.0
11798.0 22787.0 23273.0 29958.0 31003.0 0.0 31835.0 31299.0 20172.0 24609.0
34548.0 10274.0 20486.0 14062.0 12436.0 14304.0 0.0 5206.0  8709.0  20002.0
39945.0 13402.0 20203.0 13779.0 12154.0 38421.0 4935.0  0.0 10169.0 19720.0
7773.0  15017.0 9493.0  5533.0  5510.0  10139.0 8309.0  10006.0 0.0 6185.0
13352.0 32225.0 3542.0  8807.0  9852.0  22047.0 19512.0 17371.0 6371.0  0.0


coordinates
45.501913216243466  9.155222881632804
45.413149379990955  9.13102880553654
45.52193684380405   9.25895923951282
45.48130281094056   9.264593427685938
45.46015219293106   9.250173237436732
45.514980154058655  9.122147903431854
45.42412063949612   9.205127171173423
45.41191649326344   9.232036984811892
45.484108039820775  9.208482454165662
45.519310780983155  9.237730721196524
```

E Solution plots

solution_first.txt

Plot of the objective function and variables of the first test configuration, with endurance e = 25 min

```
CPLEX 20.1.0.0: optimal integer solution within mipgap or absmipgap; objective 66.73100871
17970528 MIP simplex iterations
854037 branch-and-bound nodes
absmipgap = 0.00568811, relmipgap = 8.52393e-05
 tau [*,*]
:      1          2          3          4          5          6          7     :=
0    18.5252    14.6018    40.9514    12.1384     9.23054   12.0385    10.7724
1     0         18.1554    42.9764     9.80018   18.1971    33.4193    49.754
2    17.9526     0          4.4065    13.555      4.98658   18.5058    14.5959
3    42.6767     3.83239    0         23.7235     8.81897   57.8124    33.4178
4     8.73099   12.5529    24.5795     0         12.5186    10.5428    25.4981
5    18.3254     5.65911    9.49896    9.6272     0         14.5616    11.4509
6    15.4414    59.9955    50.4742    10.2282    14.5869     0         26.0438
7    53.7832    14.5064    33.5535    24.6719    11.1945    28.3984     0
8     8.09723   15.0224    27.6275     3.93976   12.9108     8.10617   24.3856
9    39.9836     6.51208    6.87444   21.0304    11.6493    55.1208    36.9773

:      8          9          10        :=
0    12.2189    45.6159     0
1     8.82642   37.0519    19.7003
2    15.5353     6.34357   13.8324
3    25.9991     7.08023   42.705
4     4.52729   18.6549    15.0611
5    11.1557    11.4927     9.88965
6     7.45004   54.4333    13.0734
7    24.7525    38.2195    10.7128
8     0         21.7044    13.7757
9    23.3075     0         46.2645
;

tau1 [*,*]
:      1          2          3          4          5          6          7     :=
0    13.2329    10.4303    29.2522     8.67064    6.59352    8.59928    7.69493
1     0         12.9687    30.6988     7.00043   12.9985    23.872     35.5401
2    12.8238     0          3.14764    9.68257    3.56199   13.219     10.4261
3    30.4847     2.73754    0         16.9461     6.29953   41.2963    23.8709
4     6.23669    8.96677   17.5575     0          8.94227    7.53089   18.2137
5    13.0901     4.04239    6.78526    6.87686    0         10.4016     8.17959
6    11.03      42.8558    36.0545     7.30614   10.4197     0         18.6035
7    38.4182    10.3622    23.9678    17.6236     7.99638   20.2855     0
8     5.78398   10.7307    19.7348     2.81423    9.22241    5.79037   17.419
9    28.5609     4.65168    4.91052   15.0224     8.32126   39.3737    26.4135

:      8          9          10        :=
0     8.72816   32.5841     0
1     6.30486   26.4668    14.0722
2    11.0971     4.53132    9.8807
3    18.5716     5.05752   30.5049
4     3.23392   13.3255    10.7584
5     7.96868    8.20942    7.06434
6     5.32169   38.8826     9.33852
7    17.6811    27.3008     7.65232
8     0         15.5038     9.84022
9    16.6489     0         33.0475
;

x [*,*]
:   1   2   3   4   5   6   7   8   9   10     :=
0   0   0   0   0   0   0   0   1   0   0
```

```
1    0    0    0    0    0    0    0    0    0    0
2    0    0    1    0    0    0    0    0    0    0
3    0    0    0    0    1    0    0    0    0    0
4    0    1    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    1
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    1    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0
;

y [*,1,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    1    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,2,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,3,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,4,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,5,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
```

```
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,6,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    1    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,7,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    1
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,8,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,9,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    1    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0
;

u [*] :=
 1   1
 2   8
```

```
 3   9
 4   7
 5  10
 6   6
 7  11
 8   5
 9   4
10  11
;

p [*,*]
:   1   2   3   4   5   6   7   8   9     :=
0   1   1   1   1   1   1   1   1   1
1   .   1   1   1   1   1   1   1   1
2   0   .   1   0   1   0   1   0   0
3   0   0   .   0   1   0   1   0   0
4   0   1   1   .   1   0   1   0   0
5   0   0   0   0   .   0   1   0   0
6   0   1   1   1   1   .   1   0   0
7   0   0   0   0   0   0   .   0   0
8   0   1   1   1   1   1   1   .   0
9   0   1   1   1   1   1   1   1   .
;

t [*] :=
 0   0
 1   0
 2  34.6736
 3  39.0801
 4  18.8607
 5  49.8991
 6   0
 7   0
 8  14.921
 9   0
10  66.731
;

t1 [*] :=
 0   0
 1  20.7049
 2  34.6736
 3   0
 4   0
 5  49.8991
 6   8.59928
 7  58.0787
 8  14.921
 9  39.2049
10  66.731
;
```

solution_second.txt

Plot of the objective function and variables of the second test configuration, with endurance e = 60 min

```
CPLEX 20.1.0.0: optimal integer solution within mipgap or absmipgap; objective 100.51864
19050855 MIP simplex iterations
1193850 branch-and-bound nodes
absmipgap = 0.00674008, relmipgap = 6.7053e-05
x [*,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   0   0   0   0   1   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   1   0   0   0   0   0   0
4   0   0   0   0   0   0   1   0   0   0
5   0   0   0   0   0   0   0   0   0   1
6   0   0   0   0   1   0   0   0   0   0
7   0   0   0   0   0   1   0   0   0   0
8   0   0   0   0   0   0   0   0   1   0
9   0   0   1   0   0   0   0   0   0   0
;

y [*,1,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   1
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0

 [*,2,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   1   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   0
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0

 [*,3,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   0
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0

 [*,4,*]
:   1   2   3   4   5   6   7   8   9   10    :=
0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0
2   0   0   0   0   0   0   0   0   0   0
```

```
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,5,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,6,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,7,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,8,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0
5    0    0    0    0    0    0    0    0    0    0
6    0    0    0    0    0    0    0    0    0    0
7    0    0    0    0    0    0    0    0    0    0
8    0    0    0    0    0    0    0    0    0    0
9    0    0    0    0    0    0    0    0    0    0

 [*,9,*]
:    1    2    3    4    5    6    7    8    9   10     :=
0    0    0    0    0    0    0    0    0    0    0
1    0    0    0    0    0    0    0    0    0    0
2    0    0    0    0    0    0    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0
```

```
4   0   0   0   0   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0
7   0   0   0   0   0   0   0   0   0   0
8   0   0   0   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0   0
;

u [*] :=
 1   5
 2   4
 3   6
 4   7
 5  10
 6   9
 7   8
 8   1
 9   3
10  11
;

p [*,*]
:   1   2   3   4   5   6   7   8   9    :=
0   1   1   1   1   1   1   1   1   1
1   .   0   1   1   1   1   1   0   0
2   1   .   1   1   1   1   1   0   0
3   0   0   .   1   1   1   1   0   0
4   0   0   0   .   1   1   1   0   0
5   0   0   0   0   .   0   0   0   0
6   0   0   0   0   1   .   0   0   0
7   0   0   0   0   1   1   .   0   0
8   1   1   1   1   1   1   1   .   1
9   1   1   1   1   1   1   1   0   .
;

:       t          t1        :=
0      0          0
1      0         67.5119
2      0         22.3956
3     32.9824     0
4     39.7966    39.7966
5     81.9254     0
6     60.5953     0
7     53.2362    53.2362
8     10.6263     0
9     19.8494     0
10   100.519    100.519
;
```