Project: Travelling Salesman Problem (TSP)

The optimisation problem

The travelling salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?".

Material available

- a collection of instances (asymetric TSP)
- a parser for the provided instances

Activities

- Using the parser provided, load in memory an instance.

- Write a greedy algorithm (HEU) based on the nearest neighbor for computing a sub-optimal solution;
- Solve a given instance; report the best tour obtained and the CPUtime consumed.

- Using the Miller–Tucker–Zemlin (MTZ) formulation, write the corresponding model with JuMP;
- Solve a given instance; report the optimal tour obtained and the CPUtime consumed.

- Using the Dantzig–Fulkerson–Johnson (DFJ) formulation, write the corresponding model with JuMP;
- Solve a given instance; report the optimal tour obtained and the CPUtime consumed.

- Write a program which perform a numerical experiment using all instances provided;
- Report a textual and graphical synthesis of the results collected.

Didactic example

```
 0 786 549 657 331 559 250
786 0 668 979 593 224 905
549 668 0 316 607 472 467
657 979 316 0 890 769 400
331 593 607 890 0 386 559
559 224 472 769 386 0 681
250 905 467 400 559 681 0
```

HEU: greedy Heuristic (🌶️)

Solution obtained for the didactic example :

Starting from the city n°1 :

```
Heuristic solution of TSP according the nearest neighbor :
  departure location  = 1
  resolution time     = 0.0
  length of the TSP   =  2586
  sequence of visited locations :
    1 -> 7
    7 -> 4
    4 -> 3
    3 -> 6
    6 -> 2
    2 -> 5
    5 -> 1
```

```
Produce the program in Julia
```

MTZ formulation (🌶️)

The mathematical formulation proposed

$$
\begin{array}{rclcll}
\min z & = & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & & \\
s.c. & & \displaystyle\sum_{j\in\{1,\ldots,n\}\setminus\{i\}} x_{ij} & = & 1 & \forall i \in \{1,\ldots,n\} \quad (1') \\
& & \displaystyle\sum_{i\in\{1,\ldots,n\}\setminus\{j\}} x_{ij} & = & 1 & \forall j \in \{1,\ldots,n\} \quad (2') \\
& & t_i - t_j + n.x_{ij} & \leq & n-1 & \forall i,j \in \{2,\ldots,n\} \quad (3'') \\
& & x_{ij} \in \{0,1\} & & & \forall i,j \in \{1,\ldots,n\} \\
& & 0 \leq t_j \leq n & & & \forall j \in \{2,\ldots,n\}
\end{array}
$$

This first model integrates a notion of time (date $t_j \geq 0$ when a city is visited, with $j \in \{2,\ldots,n\}$)

Solution obtained for the didactic example :

```
Optimal solution of TSP according MTZ :
  resolution time     = 0.0013
  length of the TSP   = 2575
  sequence of visited locations :
    1 -> 7
    7 -> 4
    4 -> 3
```

```
    3 -> 2
    2 -> 6
    6 -> 5
    5 -> 1
```

Produce the program in Julia and JuMP.

DFJ formulation (🌶️🌶️🌶️🌶️)

The mathematical formulation proposed

$$
\begin{array}{lclllll}
\min\ z & = & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & & & \\
& s.c. & \sum_{j\in\{1,\ldots,n\}\setminus\{i\}} x_{ij} & = & 1 & \forall i \in \{1,\ldots,n\} & (1') \\
& & \sum_{i\in\{1,\ldots,n\}\setminus\{j\}} x_{ij} & = & 1 & \forall j \in \{1,\ldots,n\} & (2') \\
& & \sum_{i,j\in S} x_{ij} & \leq & |S|-1 & \forall S \text{ avec } 2 \leq |S| \leq n-1 & (3') \\
& & x_{ij} \in \{0,1\} & & & \forall i,j \in \{1,\ldots,n\} &
\end{array}
$$

This model is solved iteratively. First the linear assignment problem -constraints $(1')$ and $(2')$)- is solved, and while subtours appear in the optimal solution, one additionnal constraint -among $(3')$- is added to eliminate the subtours. The added constraint added has the role of breaking the shortest subtour.

Solution obtained for the didactic example :

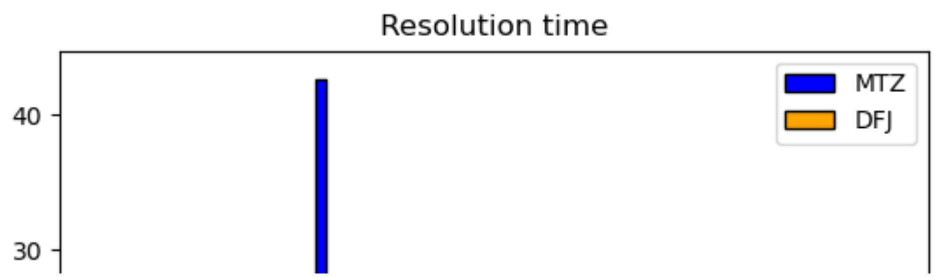
```
Optimal solution of TSP according DFJ :
  resolution time     = 0.0004
  length of the TSP   = 2575
  sequence of visited locations :
    1 -> 7
    7 -> 4
    4 -> 3
    3 -> 2
    2 -> 6
    6 -> 5
    5 -> 1
  nbCstAdded : 3
    cstSsTour1 : x[4,3] + x[3,4] ≤ 1
    cstSsTour2 : x[5,1] + x[1,5] ≤ 1
    cstSsTour3 : x[6,2] + x[2,6] ≤ 1
```

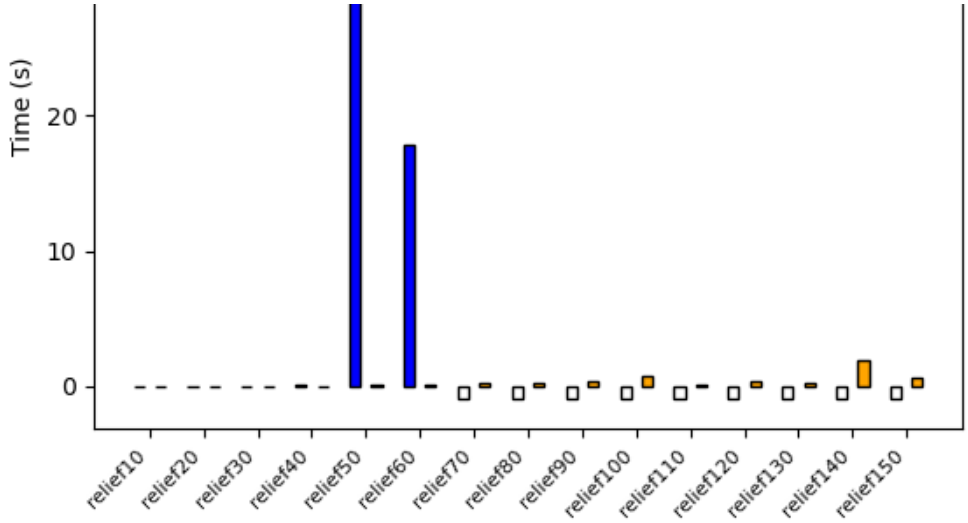
Produce the program in Julia and JuMP.

The numerical experiment (🌶️🌶️)

The results obtained for the collection of instances provided :

fname	MTZ(d)	MTZ(t)	DFJ(d)	DFJ(t)	#csts	HEU(d)	HEU(t)
relief10	198	0.001	198	0.001	2	241	0.000
relief20	147	0.009	147	0.002	1	187	0.000
relief30	116	0.009	116	0.002	0	223	0.000
relief40	105	0.028	105	0.009	2	153	0.000
relief50	155	42.545	155	0.117	9	291	0.000
relief60	136	17.812	136	0.149	8	207	0.000
relief70	-1	-1.000	115	0.208	8	216	0.000
relief80	-1	-1.000	99	0.220	6	298	0.000
relief90	-1	-1.000	118	0.303	5	268	0.000
relief100	-1	-1.000	103	0.699	12	207	0.000
relief110	-1	-1.000	113	0.138	1	279	0.000
relief120	-1	-1.000	103	0.311	3	297	0.000
relief130	-1	-1.000	107	0.171	1	309	0.000
relief140	-1	-1.000	111	1.960	22	337	0.000
relief150	-1	-1.000	100	0.660	7	294	0.000




Time (s)
0
10
20
relief10
relief20
relief30
relief40
relief50
relief60
relief70
relief80
relief90
relief100
relief110
relief120
relief130
relief140
relief150

Entrée [ ]: