Homework : Travelling Salesman Problem

I. Mezghani, J.-C. Delvenne & J. Hendrickx

Main objectives of the homework

1. Implement different formulations of the same problem.
2. Become familiar with what you can do with solvers.
3. Explore the techniques at your disposal to improve performances.

All the code requested is due in AMPL. I strongly advise you to use Gurobi as a solver. All the information about the options on this solver can be found in :

— Gurobi : `www.gurobi.com/documentation/7.5/ampl-gurobi/parameters.html`

The homework can be done **alone or by groups of two**. The deadline is on **December, 22nd**.

General suggestions

— Try to be as concise as possible. Summarizing results in a table is most of the time more helpful than a long text.
— Use relevant names for parameters, variables etc.. in your code. Do not hesitate to comment it also.
— You will not always be able to obtain the optimal solution depending on the formulation, the instance, the solver, the options you choose and your computer. It will not be penalized. It is strongly encouraged to limit the executing time for solving the problems and to report the relevant information in order to present the results.

Travelling Salesman Problem (TSP)

The TSP is one of the most famous problems in operations research. This problem is NP-hard and is a particular case of the Vehicle Routing Problem.

The problem can be stated as follows : given n cities, a cost of transportation matrix between the cities $c_{ij} \geq 0$, one looks for the cheapest travel through all the cities. In combinatorial optimization, this is the problem of finding the cheapest Hamiltonian cycle in a complete graph $G = (V, E) = K_n$.

1 Formulations of the TSP

Q1. Classical formulation

- **Remember the formulation of the TSP** that you've seen in class with subtour elimination constraints written this way :

$$\forall \emptyset \subsetneq S \subsetneq V : \quad \sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad (1 \leq |S| \leq n-1)$$

- **How many constraints and variables do we have in this formulation ?**

Other formulations of the problem exist, we'll explore some of them in the next questions.

Q2. Sequential formulation
The idea in this formulation is to introduce an integer variable u_i representing the the order of visit of city $i \in V$. For example, if the optimal subtour in a 4-node graphe is 1-3-4-2-1, then $u_1 = 1$, $u_3 = 2$, $u_4 = 3$ and $u_2 = 4$ (since the beginning of the tour is arbitrary, you can consider that $u_1 = 1$).

With this extra variable, you should be able to replace the subtour elimination constraint by a constraint expressing the fact that if the edge ij is chosen, i should be visited before j.

- **Write the sequential formulation of the TSP**.
  You should have $n^2 - n + 2$ constraints, $n(n-1)$ binary variables and $(n-1)$ integer variables in this formulation.

Q3. Flow formulation
Now, in this formulation, we try to see the TSP as a flow problem. Indeed, one

could imagine that an company needs to inject $n-1$ units of a product in the network and assign one unit of the product at each node by making the cheapest tour.

- **Write the flow formulation of the TSP**. To model the flow formulation, we introduce the variable y_{ij} representing the flow between nodes i and j. We will replace the subtour elimination contraints by three different type of constraints :
  1. We inject $n-1$ units in node 1.
  2. Each node (or city) keeps exactly one unit of the product (except for node 1).
  3. There can be units flowing through edge ij only if edge ij is chosen in the tour.

  You should have $n(n+2)$ constraints, $n(n-1)$ binary variables and $n(n-1)$ integer variables in this formulation.

2 First implementation

Q4.

- **Implement the three formulations presented. Test them on the furnished TSP instances and report the performances**.

You have to define yourself what can be relevant to report (execution time, bounds, etc..) and you are encouraged to synthesize the results with tables or graphs. In this question, no improvements should be done, so try to limit the automatic improvements that solvers apply when solving a MIP (see options `presolve, heurfrac, cuts and mipstart` with Gurobi solver for example).

The TSP instances you should test in this question are available in the folder `TSP_instances`. The size of the instances are from 14 to 52 cities. You are also not forced to let the computation terminates to report results : in that case, it might be interesting to report the guarantee you have on your solution for example. The computational time for solving one instance should not exceed 5 or 10 minutes (even if you do not reach optimality).

An example of how to write the subtour elimination constraints is given in `SubtourElim.mod`.

Explaining what you tested and presenting your results should not exceed 1 page.

3 Improvements

Q5.

(a) **Write one heuristic of the TSP and implement it in AMPL**.

(b) **Improve what you implemented in Q4 for all three formulations.** The heuristic you implemented should be used as first solution at the root of the branch and bound (you just need to fix the variables of your problem with the `let` command in AMPL). Also, you should add relevant cuts for each formulation (see `cuts` in Gurobi options for example). Report the results in the same way you did it in Q4 and underline the improvements.

Like in Q4, **explaining what improvements you made and presenting your results should not exceed 1 page.**

Q6. A test-case with 131 cities is also furnished, `xqf131.dat`.

- **Propose an implementation that will make the computation as fast as possible.** The idea is to use the formulation that you think is the best in Q5.(b), and maybe continue the tuning on different options of the solver or on the formulation itself.

Monitoring & Deliverable

Appointments upon request by email : `ilyes.mezghani@uclouvain.be`
The homework must be sent by email to Ilyes. The deliverable is one pdf file reporting what you've done and AMPL code files. The whole folder has to be compressed before being sent (format : `.zip`)