Homework 1, LINMA 2450
Nicolas Stevens
13/10/2021

1 The integer knapsack problem

In this homework, you are asked to solve the integer knapsack problem

$$
\begin{aligned}
z = \max \ & \sum_{i=1}^{n} x_i c_i \\
& \sum_{i=1}^{n} x_i a_i \leq b \\
& x \in \mathbb{Z}_+^n
\end{aligned}
$$

where b, c_i and a_i are all positive integers.

Your are asked to solve *four instances* with *three different algorithms* implemented in Julia (see section 2). The four instances are attached to this assignment (in .json files) and are named *small* ($n = 10$), *medium* ($n = 100$), *large1* and *large2* ($n = 1000$). The three algorithms you are asked to implement are:

1. *Commercial solver such as Gurobi*: implement the problem in Julia (JuMP) and solve it with Gurobi. Do not forget to turn off automatic *heuristic*, *presolve*, and *cut generation* — detailed explanation in the next page.
2. *Greedy algorithm*: propose and implement a greedy algorithm that retrieves a feasible solution of the problem.
3. *Dynamic programming*: implement a DP algorithm that solves the problem.

You will have to submit both your code and a written report (max 2 pages). The code of the three algorithms should output (i) a solution x^*, (ii) the associated objective z^* and (iii) the run time. In your report, you are asked to:

- briefly describe the algorithms 2 (greedy) and 3 (dynamic programming) you have chosen to implement and provide their complexity (and justify it) ;
- check the coherence of the algorithms results on the *small* instance, report the x^* solution of this *small* instance and demonstrate, with one argument, why this x^* is a *feasible* and *optimum* solution ;
- report (ii) and (iii) for the four instances and the three algorithms and briefly discuss the results.

2 Julia

In this semester, the students need to use Julia for implementation. Julia is an open source high-level programming language optimized for numerical analysis and computations. A domain specific language JuMP embedded in Julia is designed specifically for mathematical programming. JuMP is fast, solver independent, and has a syntax structure which mimics natural mathematical expressions. Starting from the USA, the Julia community is growing fast especially for mathematical programming community.

You need to prepare four things before start coding.

- Julia Language
- JuMP Manual
- Optimization Solver (Gurobi, CPLEX, etc) Gurobi / Gurobi JuMP Manua
- Choose your favourite IDE (e.g. Atom, Jupyter notebook, etc.).

For the optimization solver, you can use something else, but we highly recommend Gurobi. You can get a student license for free.

3 Parameter Settings for Gurobi

For comparing different IP formulations numerically, it is very important to turn off some settings of the solver. For Gurobi, you need to set following three parameters to zero: Presolve, Heuristics, and Cuts. One way to set the parameters in Gurobi is using set optimizer attribute (e.g. `set_optimizer_attribute( myModel, "Presolve", 0)`). You can check the details in the Gurobi JuMP Manual above.

4 Submission

This homework should be performed by groups of two students. You should submit both code and report in a zip file until 23:59 3/11/2021. No late acceptance. The report and the code should be the *personal* work of each group.

- Code (in julia)
- Report (.pdf file within 2 pages)
- File Name (include FirstName LastName for each group members)