POLITECNICO DI TORINO
NUMERICAL OPTIMIZATION FOR LARGE SCALE PROBLEMS AND STOCHASTIC OPTIMIZATION

Problem list, stochastic optimization a.y. 2022/23

Please choose one of the following problems. Whatever problem you choose, I expect you to submit a single pdf file, and a python folder. The folder must contain a commented script that implements an algorithm and tests it on instances randomly generated (see the problem list below). Please make sure to use sensible names for the variables and functions, and to provide enough comments and explanations to render the code readable to a non expert of the specific language.

The report must contain a description of the methodology used (e.g. describing the operators that you used) and a summary of the results you got from running your script. Please explain and comment if you are satisfied with the results you got or if it seems that something went wrong. Be aware that I may not be able to execute the code, therefore include all the numbers and figures that you obtain as an output, before commenting on them.

Please make every effort to make the whole report understandable.

NB: Some exercises are marked as difficult because they are more demanding than the others, for these ones I suggest to work in group of 3 people. Recall that:

"The greater the difficulty, the more glory in surmounting it."
Epictetus

1. The Capacitated Clustering Problem

Given a set I of nodes, each one characterized by a position $\bar{x}_i \in \mathbb{R}^2$ and a weight w_i, the goal of the Capacitated Clustering Problem (CCP) is to divide them into several disjoint clusters so that the sum of the node weights in each cluster meets a given capacity limit. The mathematical model describing the aforementioned problem consider the following parameters:

- d_{ij} is the distance between point i and j,
- C_i is the capacity of cluster i,
- $w_i(\omega)$ is the weight of point i (it is a random variable),

and the following decision variables:

- x_{ij} is 1, if point i is assigned to cluster j and 0 otherwise,
- y_i is 1, if node i is the center of the cluster and 0 otherwise.

The mathematical model is:

$$\min . \sum_{i\in I}\sum_{j\in I} d_{ij}x_{ij} + \lambda\mathbb{E}\left[\sum_{j\in I}[C_j y_j - \sum_{i\in I} w_i(\omega)x_{ij}]^+\right] \tag{1}$$

$$\text{s.t.} \sum_{j\in I} x_{ij} = 1, \quad \forall i \in I \tag{2}$$

$$\sum_{j\in I} y_j \leq p, \tag{3}$$

$$x_{ij} \leq y_j, \quad \forall i \in I, \quad \forall j \in J \tag{4}$$

$$x_{ij}, y_j \in \{0,1\}, \quad \forall i \in I, \quad \forall j \in J$$

The objective function (1) aims at minimizing the distance of each point from the center of the cluster plus a quantity that depends by the amount of weight not met. The parameter λ weights the two terms of the objective function. Constraints (2) impose that each point must be assigned and constraint (3) that the maximum number of clusters is p. Finally, constraints (4) force the x_{ij} to be one, only if the point j has been chosen as a cluster center.

If you consider each point to be a physical location and the weight to be a demand for some service, this problem can be applied to the design of garbage collection zones, defining salesmen areas, hospital location, etc.

You are requested to:

(1) create a script generating instances of the problem. You can choose the assumptions that you want (e.g. consider a mix of multivariate normal distribution for the points and a normal demand for each point);
(2) create a script generating the random realizations of the weights. Again, you can choose the assumptions that you want;

(3) solve the model by using gurobi
(4) study the in sample, out of sample stability and the VSS;
(5) use heuristic algorithm to solve the problem and compare (in terms of computational time and objective function) the solutions with the ones of gurobi (customized heuristics are welcomed).

2. The Stochastic Generalized Bin Packing Problem [DIFFICULT]

Let us consider a logistic provider that has to ship a set of items (with volume and profit) and may use a set of bins (with capacity and cost). Items can be compulsory (i.e., mandatory to load) or non-compulsory, while bins are classified in bin types. Bins belonging to the same bin type have the same capacity and cost (e.g. trucks, vans, bikes, etc). Moreover, a maximum number of bins can be used for each bin type. The aim of the problem is to accommodate all the compulsory items and possible non-compulsory items into appropriate bins in order to minimize the overall cost, given by the difference between the costs of the selected bins and the profits of the loaded non-compulsory items. Let us call:

- a set J of bins
- a set I of items divided in compulsory item I^C and non compulsory item I^{NC};

Moreover, let us call:

- C_j the cost of each bin;
- p_i the profit of item i;
- w_i the weight of item i;
- W_j the capacity of bin j;
- B is the maximum budget.

The mathematical model of the problem is:

$$\min . \sum_{j \in J} C_j y_j - \sum_{i \in I^{NC}} p_i \sum_{j \in J} x_{ij} \tag{5}$$

$$\text{s.t.} \sum_{i \in I} w_i x_{ij} \leq W_j y_j, \quad \forall j \in J \tag{6}$$

$$\sum_{j \in I} x_{ij} = 1, \forall i \in I^C \tag{7}$$

$$\sum_{j \in I} x_{ij} \leq 1, \forall i \in I^{NC} \tag{8}$$

$$\sum_{j \in J} C_j y_j \leq U, \tag{9}$$

$$x_{ij}, y_j \in \{0, 1\}, \quad \forall i \in I, \quad \forall j \in J$$

The objective function (5) minimizes the total net cost of the packing, given by the difference between the total cost of the used bins and the total profit of the selected non-compulsory items. The profit of the compulsory items is not considered in the objective function because it is a constant. Constraints (6) force not to exceed the bins capacities, constraints (7) and (8) force to select all the compulsory items and

to allocate each item into one bin. Finally, constraint (9) limits the budget that can be used. You are requested to:

(1) create a script generating instances of the problem. You can choose the assumptions that you want (better if they are realistic, see here for ideas);
(2) solve the model by using gurobi;
(3) create at least two customized heuristics to solve the problem and compare the solutions with the ones provided by gurobi (in terms of computational time and objective function).

[Code available on the website]

3. Reinforcement learning

We consider the DLSP on parallel machines. Let I be the number of items, M the number of machines, and T the number of time periods. We call:

- $[I]$: the set of items.
- $[M]$: the set of machines.
- $[T]$: the set of time periods.
- $\mathcal{I}(m)$ the set of items that can be produced by machine m.

Moreover, we define the following parameters:

- $d_{i,t}$: demand of item i in period t.
- $p_{i,m}$: production of the item i for the machine m.
- $f_{i,m}$: setup cost matrix when changing the machine setup of the machine m to item i.
- h_i: inventory cost vector of the item i.
- c_i: setup loss matrix when changing the machine setup to produce item i.
- l_i: lost sales cost of item i.

The decision is to select the best item for every machine in order to minimize the total cost. We define the decision variable $x_{i,m,t}$ which is equal to 1 if machine m is producing item i during t.

The state is composed by the item produced by each machine and by the inventory. While the evolution of the first is straightforward, the evolution of the second is:

$$I_{i,t+1} - z_{i,t} = I_{i,t} + \sum_{m=1}^{M}(p_{i,m}x_{i,m,t} - c_i\delta_{i,m,t}) - d_{i,t}, \qquad \forall i \in [I],\ t \in [T]$$

where

- $I_{i,t}$: inventory of item i at time t.
- $z_{i,t}$: the lost sales of item i at time t.
- $\delta_{i,m,t}$: binary variable equal to 1 if machine m has done a setup between time $t-1$ and time t.

Finally the cost that we pay in each time step is

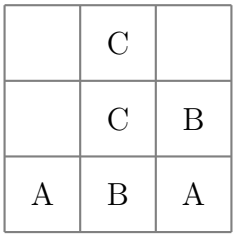
$$\sum_{i=1}^{n}(h_iI_{i,t} + l_iz_{i,t} + \sum_{m=1}^{M} f_m\delta_{i,m,t})$$

You are requested to develop a methodology to provide solution to the problem.

[Code available on the website]

4. Reinforcement learning [DIFFICULT]

Warehouse premarshalling (also pre-marshalling or remarshalling) is the activity of reordering items in a storage location so that subsequent retrieval orders can be serviced with little or no need for further relocations. It has deep impact on warehouse efficiency. We are interested in a stochastic case, where pickup and deliveries become known only at the moment when they are to be retrieved. Consider a 2D warehouse (as in the image).



We call:

- I different types of items;
- $W \times H$ slots.

Note that the problem is trivial if $W \times H > I$ (you allocate one column for each type of item) while it is difficult for $W \times H < I$. In each time step:

(1) a possible demand for a given item contained in the warehouse (we know $\mathbb{P}[\text{new delivery} = i]$);
(2) a possible new item to allocate (we know $\mathbb{P}[\text{new pick-up} = i]$);
(3) you have to allocate the new item and re-arrange the warehouse (if needed).

Each movement has a fixed cost c, the goal is to fulfill all the orders by the least possible amount of movements. If needed you can use an extra column to temporary locate the items to satisfy the demand. But it must be voided before the end of the time step.

You are requested to develop a methodology to provide solution to the problem.

[Code available on the website (maybe some bug)]

Suggestions:

- start with easy instance and plot the grid to see if the approach proposed is working;
- care must be taken for unfeasible operations (we suggest to take that into account at the agent level).