Home Assignment 2

CS 787 – Decision Guidance Systems – Professor Alex Brodsky

Consider the posted folder **cs787_ha2_supp_manuf_transp_sn_template** folder. Install Pyomo package:
https://pyomo.readthedocs.io/en/stable/index.html

Problem 1

Instructions

Under the subfolder **solution,** duplicate the Python module **ams_template.py** into **ams.py** Implement the required analytic models (see below) by filling out **ams.py** template. Example inputs and corresponding outputs are given in the folder **example_input_output.** To run the implemented functions in Problem 1 below, use **solution/main.py**

a. In terminal, make the downloaded folder your current folder
b. In **main.py** uncomment invocation of the function you want to run, e.g. **answer = ams.supplierMetrics(input)** and comment other invocations
c. Run **main.py** in Python w/ stdin being the model input, e.g., **example_input_output/supplier_in.json** and indicate the file in stdout

Example: **python solution/main.py < example_input_output/supplier_in.json > answers/out.json**

to get answers in **out.json** which you can compare with the correct answer, e.g., in **example_input_output/supplier_out.json**

questions

Implement the following analytic models as Python functions in the module **ams.py**

1. **supplierMetrics(input)**
   a. example input: **supplier_in.json**
   b. example output: **supplier_out.json**
   c. **cost** in the output structure is the total cost of procurement from the supplier in the **input** based on **ppu** (price-per-unit) and purchased **qty.**
   d. **constraints** in the output structure are computed to **True** iff, for every key in **outflow,** the purchased **qty** is non-negative, and is greater than or equal to **lb** (lower bound)
2. **manufMetrics(input)**
   a. example input: **manuf_in.json**
   b. example output: **manuf_out.json**
   c. **cost** in the output structure is the total cost of manufacturer in the **input** computed based on **ppu** (price-per-unit) and ordered **qty,** for all ordered products, which are represented as keys in **outflow**

d. **qty** of each key in **inFlow** is computed based on **qty**’s of ordered items in **input[“outFlow”]** and on **input[“qtyInPer1out”].** For example, if **input[“outFlow”][“table”] = 100,** and **input[“qtyInPer1out”][“table”][“table_leg”] = 4,** then **output[“inFlow”][“table_leg”]** should be 100 * 4 = 400.

e. **constraints** in the output structure are computed to **True** iff, for every key in **inflow** and in **outFlow,** the **qty** is non-negative, and is greater than or equal to **lb** (lower bound)

3. **transportMetrics(input,shared)**

   a. example input: **transp_in.json**

   b. example output: **transp_out.json**

   c. **input[“orders”]** contains transportation orders. Each order is described using (**in, out, sender, recipient, qty**) where **in** and **out** indicate item/loc being sent and received; **sender** and **recipient** are business entities (described in **shared**), which have an associated location.

   d. **Cost** in the output is computed as follows. For every pair of (source, destination) locations, compute the total weight of all shipments (in **orders**). Then, the cost for a (source, destination) is total weight times price-per-lb (in **input[“pplbFromTo”])** Finally, the total cost is aggregation of cost for all (source, destination) pairs.

   e. **inFlow** and **outFlow** quantities in the output, for every item/loc key, is computed from the corresponding orders

   f. **constraints** in the output structure are computed to **True** iff, for every key in **inflow** and in **outFlow,** the **qty** is non-negative, and is greater than or equal to **lb** (lower bound)

4. **combinedSupply(input)**

   a. example input: **combined_supply_in.json**

   b. example output: **combined_supply_out.json**

   c. Computed **cost** is the summation of costs of suppliers in **input[“services”][“combinedSupply”][“subServices”]**

   d. **outFlow** quantities in the output are computed by aggregating **outFlow** quantities of the corresponding suppliers

   e. **inFlow** in the model output is empty

   f. **constraints** in the output is a Boolean value that is **True** iff all quantities in **inFlow** and **outFlow** are non-negative and greater-than-or-equal their corresponding lower bounds (**lb)**

5. **combinedManuf(input)**

   a. example input: **combined_manuf_in.json**

   b. example output: **combined_manuf_out.json**

   c. Computed **cost** is the summation of costs of manufacturers in **input[“services”][“combinedManuf”][“subServices”]**

   d. **outFlow** quantities in the output are computed by aggregating **outFlow** quantities of the manufacturers in **input[“services”][“combinedManuf”][“subServices”]**

   e. **inFlow** quantities in the output are computed by aggregating **outFlow** quantities of the manufacturers in **input[“services”][“combinedManuf”][“subServices”]**

   f. **constraints** in the output is a Boolean value that is **True** iff the following holds:
      i. **keys(input["services"]["tier1manuf"]["outFlow"] = keys(input["services"]["tier2manuf"]["inFlow"])**
      ii. Let **S** be the set of keys in (i). Then, for every key **k** in **S,** the quantities for **k** in **input["services"]["tier1manuf"]["outFlow"][k]** and **input["services"]["tier2manuf"]["inFlow"][k]** are equal.
      iii. Constraints computed for **tier1manuf** and **tier2manuf** are both satisfied.
      iv. Quantities in **inFlow** and **outFlow** are non-negative and greater-than-or-equal their corresponding lower bounds (**lb)**

6. **combinedTransp(input)**
   a. example input: **combined_transp_in.json**
   b. example output: **combined_transp_out.json**
   c. Computed **cost** is the summation of costs of transportation services in **input["services"]["combinedSupply"]["subServices"]**
   d. **outFlow** quantities in the output are computed by aggregating **outFlow** quantities of the transportation services in **input["services"]["combinedTransp"]["subServices"]**
   e. **inFlow** quantities in the output are computed by aggregating **inFlow** quantities of the transportation services in **input["services"]["combinedSupply"]["subServices"]**
   f. **constraints** in the output is a Boolean value that is **True** iff all quantities in **inFlow** and **outFlow** are non-negative and greater-than-or-equal their corresponding lower bounds (**lb)**

Problem 2

instructions

In this problem, you need to construct and solve an optimization problem for the analytic models of **combinedSupply, combinedManuf** and **combinedTransp**. To construct and solve optimization problems:

Under the folder **solution,** duplicate the files **optSupply_template.py, optManuf_template.py** and **optTransp_template.py** into **optSupply.py, optManuf.py** and **optTransp.py,** respectively

Questions

Implement the followins:

1. In **optSupply.py** module, implement the function **constraints(o)** that is used in **dgal.min()** optimization invocation. The function **constraints(o),** where **o** is of the form of the **combinedSupply** model output, needs to express the following Boolean value:
   a. Model constraints and
   b. The total supplied amount of mat1 (mat1_sup1 + mat1_sup2) is at least 1000 & the total amount of mat2 is 2000.

   To optimize, from the root folder, run: **python solution/optSupply.py**

2. In **optManuf.py** module, implement the function **constraints(o)** that is used in **dgal.min()** optimization invocation. The function **constraints(o),** where **o** is of the form of the **combinedManuf** model output, needs to express the following Boolean value:
    a. Model constraints and
    b. The total produced amount of product1 (prod1_manuf2) is at least 1000, and of product2 (prod2_manuf2) is at least 2000.

   To optimize, from the root folder, run: **python solution/optManuf.py**
3. In **optTransp.py** module, implement the function **constraints(o)** that is used in **dgal.min()** optimization invocation. The function **constraints(o),** where **o** is of the form of the **combinedTransp** model output, needs to express the following Boolean value:
    a. Model constraints and
    b. The total amount of delivered mat1 (mat1_manuf1) is at least 1000 & of mat2 (mat2_manuf1) is at least 2000.

   To optimize, from the root folder, run: **python solution/optTransp.py**

Play with constants in constraints and in the model input, such as prices, so that you can predict the optimal values that optimization should choose. See if this is indeed the case. If not, try to debug the model etc.

To submit

Upload the files **ams.py, combinedSupply.py, combinedManuf.py, combineTransp.py**
Nothing else!