Project – III
Supply Chain Management

CSCI 2951-O: Foundations of Prescriptive Analytics

Due Date: 10 PM March 18, 2022

1 Problem Statement

One of the most important strategic decisions in supply chain management is concerned with finding the right location of facilities and allocation of customers. This is a classical problem which deals with selecting the best locations for opening facilities and assigning customers to open locations while optimizing an objective value such as the cost of opening facilities and/or transportation costs.

To bring the problem closer to the real-life, the decision making process of location allocation is combined with vehicle management in order to provide service to customers in round-trips.

A multi-billion dollar business client of Strategic Consulting Group (SCG) would like to estimate the cost of operating such a business. This is a highly critical project! Based on the estimated cost, the client will make a final decision on whether such a venture would be profitable, hence they should enter the market, or they should consider alternative investment strategies.

The stakes are high and SCG is tasked with finding a lower bound on the total cost of this supply-chain problem for their client. Inspired by their recent success stories, SCG, rightly-so, turns to CSCI 2951-O elites!

2 Location/Allocation

Figure 1 presents an example of the integrated facility location customer allocation problem.

In this setting, we are given a set of potential locations to open facilities (the gray nodes in the network) and a set of customers to serve using a fleet of identical vehicles. The cost of opening a facility depends on the particular location. Each facility has a maximum capacity to serve its customers. Each customer is served by exactly one open facility (the buildings in Figure 1) using a vehicle (the trucks next to each building). Customers are served in round-trips meeting their demand in full. That means, in the worst case, a facility can serve each customer with a different vehicle, hence, the total number of vehicles needed at the location is bounded by the number of customers allocated. Ideally, the same vehicle can

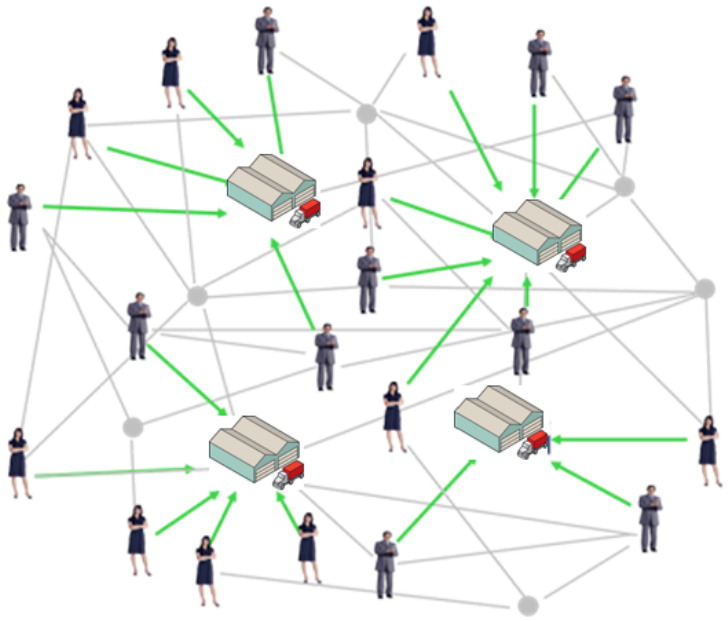

Figure 1: Example of facility location, customer allocation, and vehicle management.

be used to serve multiple customers as long as its workload does not exceed a given total driving distance. When in use, each vehicle incurs a fixed rental cost. The decision of which facility serves which customer affects the corresponding service costs and driving distances.

The ultimate goal of the problem is to decide:

1. which facilities to open (the buildings in Figure 1).
2. which vehicles to deploy in each facility.
3. which facility and vehicle to use when serving each customer.

Notice that each customer is served by exactly one open facility (the green edges), and facility capacities and vehicle driving limits should not be exceeded.

Overall, the objective is to minimize the total cost which is the sum of the costs of i) opening facilities, ii) service/allocation cost incurred at each visit between customers and facilities, and iii) fixed vehicle usage cost.

3 Your Task

This is an inherently discrete optimization problem. For example, it does not make sense to assign 3.3 vehicles to a facility. That means to find a feasible/optimal solution to this problem we need solve an Integer Programming model. As an CS2951-O elite, you know by-heart that while solving Integer Programming (IP) problems is NP-hard in general, solving Linear Programming (LP) problems is in polynomial time! Not only that, but LPs provide bounds on the objective value of the corresponding IP formulation. This is exactly the fact

you will exploit in this problem since SCG is only interested in finding a lower bound on the estimated business cost. Knowing the lower bound on the cost to operate this business at the minimum will help making the investment decision. Your task is to build a constraint model for this problem using **continuous variables**[1] and solve the Linear Relaxation to find a lower bound on the cost quickly.

Specifically, you will employ IBM Optimization Studio to create the constraint model in CPLEX Solver and solve the linear programming problem to generate a lower bound.

Notice that when an optimal solution to the LP formulation is found, it might still include variables taking on fractional values. That means the decisions suggested by this solution might not be immediately applicable in real-life (e.g., assigning 3.3 vehicles to a facility or a customer served by 0.7 facilities each providing some portion of the demand and so on). Nevertheless, in this project we are interested in finding a lower bound on the cost function to help the client of SCG.

4 Input

The input files specify the parameters used for the supply-chain problem such as the number of possible locations together with their corresponding capacity, the number of customer, traveling distances between locations, and so on.

5 Support Code

We provide you with simple LP instance object that parses the data files (LPInstance.java) so that you can immediately start modeling the problem. The compile and run scripts take care of exporting a license and linking to the constraint solver. The support comes with the LP model of the Diet problem from the lecture to get you up to speed with the solver.

The support code is written in Java. You are free to use the support code as is, tweak the data structures as you see fit, or discard it completely. If your choice of programming language is different, the support code can serve you as a reference. CP Optimizer allows C++ and C# bindings as well. However we cannot provide support on how to run the constraint solver in those languages.

Assuming you are in the `project` directory,

To compile the support code, type:

```
> ./compile.sh
```

To run the support code in a given instance, type:

```
> ./run.sh simple.scm
```

[1] That means variables should be created using:
cplex.numVar(lowerBound, upperBound, **IloNumVarType.Float**)

6 Output

Given an input file as its first command line argument via the `run` script, your solution should print out the result on its **last line**. You can print other statements before the result but they will be discarded. We suggest you to keep such debug information to a minimum since writing I/O is expensive. We use `stdout` for output and ignore other streams. Your submission will be tested on department's linux machines and it should work without any other software package installation. CPLEX solver is already installed on department machines.

The expected output is a JSON object with "Instance" mapped to the instance name, "Time" to the number of seconds it takes your program to solve this instance (in 2-digit precision), "Result" to the *ceiling* of the objective value of the linear program, and "Solution" to "OPT" to denote optimality.

```
./run.sh simple.sched
..running..bla..
..bla..
{"Instance": "simple.scm", "Time": 1.23, "Result": 10, "Solution": "OPT"}
```

In the above example, the LP is solved to optimality with a round-up objective value equals to 10.

Finally, to run a benchmark on all instances in a given input folder using a fixed time limit while collecting the result (i.e. the last line) to a log file, type:

```
> ./runAll.sh input/ 300 results.log
```

Please make sure to follow this convention *exactly*! Failing to do so will cause evaluating your submission incorrectly.

7 Submission

```
/solution
|_ src/
|_ compile.sh
|_ run.sh
|_ runAll.sh
|_ results.log
|_ report.pdf
|_ team.txt
```

- Submit all your source code under the (`/src`) folder. Remember that commenting your code is a good practice so that one can follow the flow of your program at a high-level.
- Submit your (possible updated) `compile` and `run` scripts which compiles your solution and runs it on a given instance respectively. Make sure that the *last line* of your solution output follows the specification above.
- Submit the last line found for all the instances in the input directory in the `results.log`.
- Submit a short report in pdf format (1 or 2 pages max) that contains a brief discussion of your constraint model, decision variables and the constraints used. Don't forget to write name(s) of the team members and the screenname(s) corresponding to this project. Please also include a note on how much time you spend on this project. This information is completely irrelevant to evaluation and kept solely for statistical purposes.
- Submit the cs login(s) of the team in `team.txt` listing one member per line in lowercase.

Have questions? and/or need clarifications? Please do not hesitate to contact us, your instructor and TAs are here to help. Good luck on your quest for helping out the business analysts at SCG!