Gurobi Workshop - Part I

Lan Peng, PhD Candidate

Department of Industrial & Systems Engineering
University at Buffalo, SUNY

October 26, 2021

Overview

- Part I - 26th Oct.
  - Introduction and Installation
  - Manufactures Problem - Small Example of LP
  - Parallel Machine Scheduling Problem - Simple Example of IP
  - Traveling Salesman Problem - Lazy Constraints and Callback
- Part II - 2nd Nov.
  - Vehicle Routing Problem with Time Windows - Column Generation
  - Facility Location Problem - Benders Decomposition

Introduction

- Gurobi $\rightarrow$ Dr. Gu, Dr. Rothberg, Dr. Bixby.
- The best MIP solver (for now).
- Can be integrated into Python, C++, Java, Matlab, etc.
- Homepage: https://www.gurobi.com/

Installation

- 1. To download gurobi, we will need to sign up for an account on `https://pages.gurobi.com/registration`. Remember that we need to use the .edu email address to get the free education license.
- 2. Then, we can find the distribution on the following link: `https://www.gurobi.com/downloads/gurobi-software/` (login required). This page has many different versions for different OS.
- 3. The next step will be getting a license, which we can get from `https://www.gurobi.com/downloads/end-user-license-agreement-academic/` (login required).
- 4. After we retrieved the license, we can use the terminal command to execute the ‘`grbgetkey YOUR_LICENSE`‘ to activate the license.
- 5. In terminal, go to the root direction of Gurobi, run ‘`python setup.py install`‘.
- 6. If all those steps are done, we can now try to import gurobi in python using ‘`import gurobipy`‘.

Manufactures Problem

Manufactures Problem Giapetto's Woodcarving, Inc., manufactures two types of wooden toys: soldiers and trains. A soldier sells for $3 of profit and a train sells for $2 of profit. The manufacture of wooden soldiers and trains requires two types of skilled labor: carpentry and finishing. A soldier requires 2 horus of finishing labor and 1 hour of carpentry labor. A train requires 1 hour of finishing and 1 hour of carpentry labor. Each week, Giapetto can obtain all the needed raw materials but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldiers are bought each week. Giapetto wants to maximie weekly profit. Formulate a mathematical model to Fiapetto's situation that can be used to maximize Giapetto's weekly profit.

Manufactures Problem

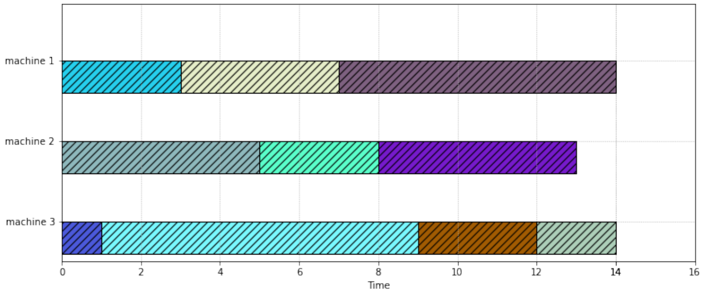
- **Decision Variable:**
  - x_1 - Number of soldiers
  - x_2 - Number of trains
- **Linear Programming Model:**

$$\begin{aligned} \max \quad & 3x_1 + 2x_2 && (1) \\ \text{s.t.} \quad & 2x_1 + x_2 \leq 100 && (2) \\ & x_1 + x_2 \leq 80 && (3) \\ & x_1 \leq 40 && (4) \\ & x_1, x_2 \geq 0 && (5) \end{aligned}$$

- **Objective Function:** (1) - maximize profit
- **Constraints:**
  - (2) - Total finishing labor hours
  - (3) - Total carpentry labor hours
  - (4) - at most 40 soldiers are brought
  - (5) - Non-negativity constraints.

Parallel Machine Scheduling Problem

Given n jobs $j_1, j_2, \cdots, j_n$ of varying processing times, which need to be scheduled on m machines while trying to minimize the makespan



Parallel Machine Scheduling Problem

- **Sets and Parameters**
  - M Set of identical machines.
  - J Set of jobs, each has a length of process time.
  - $p_j, \forall j \in J$ The process time for job j.
- **Decision Variables:**
  - $x_{ij}, \forall j \in J, i \in M$, Binary, $x_{ij} = 1$ if job j is assigned to machine m.
  - z Time span.

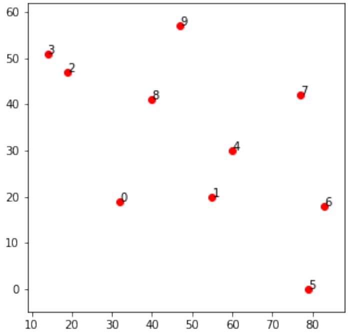
- **Integer Programming Model**

$$\begin{aligned} \min \quad & z && (6)\\ \text{s.t.} \quad & z \geq \sum_{j \in J} p_j x_{ij} \quad \forall i \in M && (7)\\ & \sum_{i \in M} x_{ij} = 1 \quad \forall j \in J && (8)\\ & x_{ij} \in \{0, 1\} \quad \forall j \in J, \forall i \in M && (9)\\ & z \geq 0 && (10) \end{aligned}$$

- **Constraints** Each job can assign to one machine once. All jobs needs to be assigned.

Traveling Salesman Problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



Dantzig-Fulkerson-Johnson Formulation

let $G = (V, A)$ be a graph where V is a set of n vertices, and A is a set of arcs (or edges). Let $C = c_{ij}$ be a cost (distance) matrix associated with A. The TSP consists of determining a minimum cost (distance) Hamiltonian circle (or cycle) that visits each vertex once and only once. If for all $i, j \in V, c_{ij} = c_{ji}$, then the TSP is symmetrical, otherwise is asymmetrical.
(We use the well-known DFJ formulation)
- **Decision variables**

$$x_{ij} = \begin{cases} 1, & \text{if goes from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}, \quad (i, j) \in A \tag{11}$$

- **Objective function**

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{12}$$

Dantzig-Fulkerson-Johnson Formulation

- **Constraints**

$$\sum_{j\in V,(i,j)\in A} x_{ij} = 1, \quad \forall i \in V \tag{13}$$

$$\sum_{i\in V,(i,j)\in A} x_{ij} = 1, \quad \forall j \in V \tag{14}$$

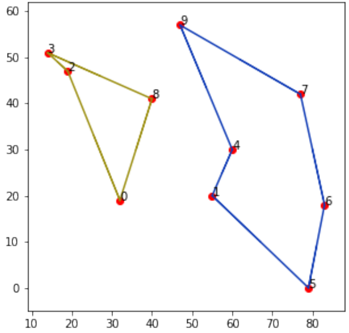
$$\sum_{i,j\in S,(i,j)\in A} x_{ij} \leq |S| - 1, \quad \forall S \subset V, 2 \leq |S| \leq n-1 \tag{15}$$

If we list all sub-tour constraints in DFJ, there will be $O(2^n)$ constraints and $O(n^2)$ binary variables. The exponential number of constraints makes it impractical to solve directly. Instead, lazy constraints are usually implemented for the sub-tour elimination constraints (15)

Subtours

In this case, two subtour constraints are identified as

$$x_{14} + x_{49} + x_{97} + x_{76} + x_{65} + x_{51} \leq 5$$
$$x_{08} + x_{83} + x_{32} + x_{20} \leq 3$$



Find subtours

Algorithm Sub-tour Searching Algorithm

```
$K = \emptyset$
$d_i = 0, \forall i \in N$
for $i \in N$ do
    $C = \emptyset$
    $Q = \emptyset$
    if $d_i == 0$ then
        $d_i = 1$
        $C = C \cup \{i\}$
        Q.append(i)
        while $Q \neq \emptyset$ do
            v = Q.pop()
            for $u \in FS(v)$ do
                if $d_u == 0$ then
                    $d_u = 1$
                    $C = C \cup \{u\}$
                    Q.append(u)
                end if
            end for
        end while
    end if
    $K = K \cup C$
end for
```