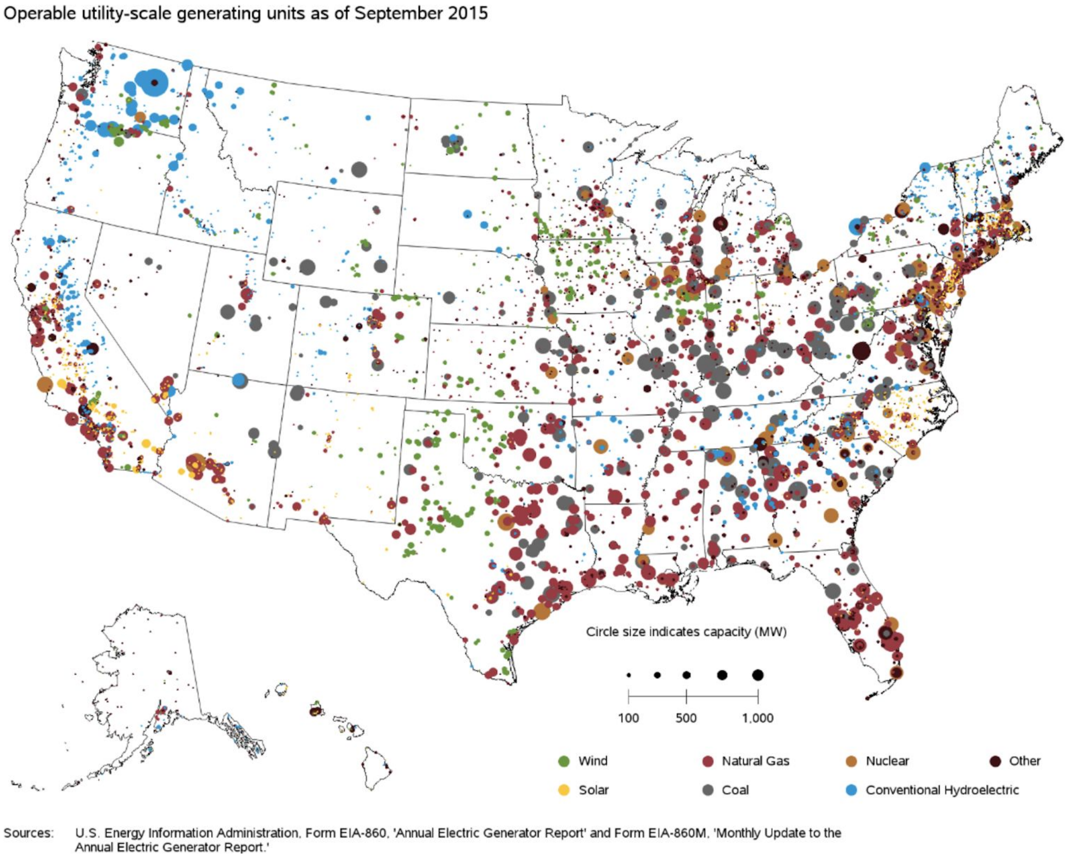
Building Mixed Integer Programs with Gurobipy in Jupyter

Disclaimer: Gurobi Optimization LLC is a private, for profit company, that makes the Gurobi solver, the Gurobipy API, and many other products. The Gurobipy API is their intellectual property and I am not affiliated with Gurobi Optimization LLC in any way. For references to official documentation visit www.gurobi.com

Motivating Problem 1 Power Plant Commitment/Dispatch



Motivating Problem 1 Power Plant Commitment/Dispatch

Let’s say that 1% of transmission infrastructure is constraining

65,520 x .01 = 655 miles

Let’s also say that each generator has 3 output levels

3 x 1700 = 5,100 levels to set

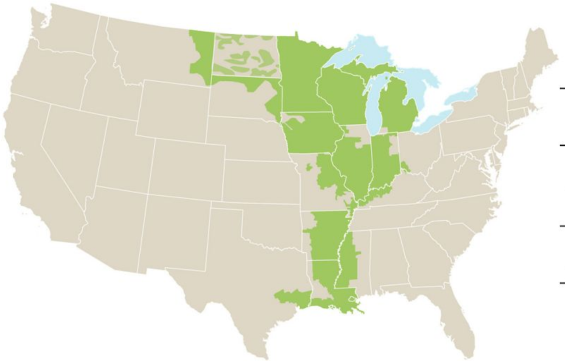
Lastly let’s say we want to develop an hourly schedule for a week out

24 x 7 = 168 hours to set

We already have 168 x 655 x 5,100 **expensive** decisions to make! **561,204,000**

We did not include
Generators can take multiple fuels, and can mix fuels
Required maintenance
Ramp Rates - rates at which generators can be brought to different levels
Starting States (cold, warm)

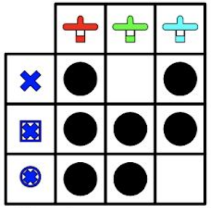
MISO is the Grid/Market Operator for 15 states in the Central U.S. plus the Canadian Provence of Manitoba



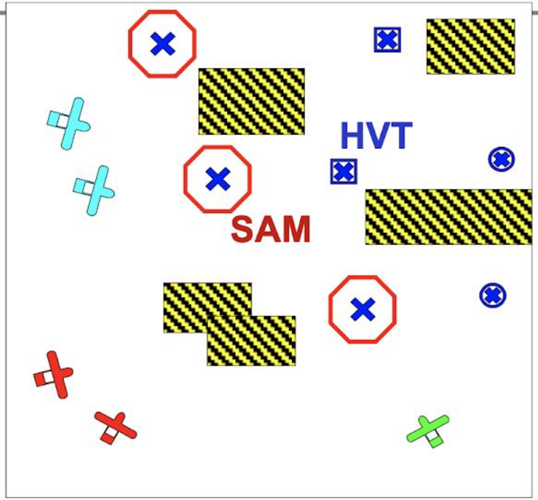
	MISO	U.S.
High Voltage Transmission - miles	65,520	200,000 +
Installed Generation - GW	195	1,009
Installed Generation – # of Units	1700	19,000
Annual System Load – Billion kWH	670	3,900
People Served - Millions	42	320

Motivating Problem 2 Attack Planning

- Different types of UAV's
- Different types of targets



- Example timing constraints
  - BDA after Strike after Recon.
  - SAM site visited before HVT
  - Synchronized strike of HVT
  - → All can be expressed in a **canonical form**



- Select assignments & trajectories to optimize desired mission objectives and satisfy dynamic/timing constraints
  - Costs for demonstration based on time, but extends to scores / risks
  - Hard because problems are tightly coupled

Source: [MIT Aerospace Controls Lab]

What is a MIP?

A Mixed Integer Program is a mathematical model of a system where variables can take integer or continuous values.

What is a MIP?

A Mixed Integer Program is a mathematical model of a system where variables can take integer or continuous values.

Examples

G1 can be generating at level 0, 1, or 2 => Integer Variable

G2 can be performing maintenance 0, 1 => Binary Variable

What is a MIP?

A Mixed Integer Program is a mathematical model of a system where variables can take integer or continuous values.

Examples

G1 can be generating at level 0, 1, or 2 => Integer Variable

G2 can be performing maintenance 0, 1 => Binary Variable

TL1 is rated to handle power from -3.0 to 3.0 MW => Continuous Variable

What is MIPing?

Mixed Integer Programming is the use of a Mixed Integer Program(**MIP**) and **solution algorithms** to find variable values that result in feasible, and optionally, optimal solutions.

By the end of this we'd like to...

1. See practical MIP examples from industry.
2. Have a fuzzy idea of what MIPing is, build one of our own.
3. Implement and solve a MIP in Anaconda using Gurobipy.
4. Learn a few tips and tricks to use when working with MIPs.

Summary,

...be able to **hack together a MIP using current DS/Analytics tools** and add it to our toolbelt.

Lets Build One

The Knapsack problem

Imagine you are a thief, and you are very deliberate about your thieving. You always know what is available to be stolen, and where it's at. When you steal, you put things in your bag. However, since you know you can't carry everything to be stolen, you must make some **DECISIONS** about what to take, and what to leave behind.

You are limited by the fact that your bag can only hold **20** pounds and that you must be in and out in **10** minutes.

The Knapsack problem

Imagine you are a thief, and you are very deliberate about your thieving. You always know what is available to be stolen, and where it's at. When you steal, you put things in your bag. However, since you know you can't carry everything to be stolen, you must make some **DECISIONS** about what to take, and what to leave behind.

You are limited by the fact that your bag can only hold **20** pounds and that you must be in and out in **10** minutes. Also, the African Grey and the Calico Cat can not be put in the bag together.

1 African Grey	2 Neighbors Calico Cat	3 Clock	4 Laptop	5 Painting	6 TV	7 Keurig Machine	8 Miniature Pincher
5lb	8lb	3lb	3lb	1lb	7lb	5lb	7lb
$ 1200	$900	$100	$800	$400	$325	$250	$400
2 min	6 min	1 min	1 min	3 min	4 min	2 min	1 min

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data =

1 African Grey	2 Neighbors Calico Cat	3 Clock	4 Laptop	5 Painting	6 TV	7 Keurig Machine	8 Miniature Pincher
6lb	12lb	3lb	3lb	1lb	7lb	5lb	7lb
$ 1200	$700	$100	$1100	$400	$325	$250	$400
2 min	6 min	1 min	1 min	3 min	4 min	2 min	1 min

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$

$v_1, v_2, \ldots, v_8$

$t_1, t_2, \ldots, t_8$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$
$v_1, v_2, \ldots, v_8$
$t_1, t_2, \ldots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$

$v_1, v_2, \ldots, v_8$

$t_1, t_2, \ldots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \dots, w_8$

$v_1, v_2, \dots, v_8$

$t_1, t_2, \dots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1 x_1 + v_2 x_2 + \cdots + v_8 x_8)$

Constraints: $w_1 x_1 + w_2 x_2 + \cdots + w_8 x_8 \leq 20$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$

$v_1, v_2, \ldots, v_8$

$t_1, t_2, \ldots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

Constraints: $w_1x_1 + w_2x_2 + \cdots + w_8x_8 \leq 20$

$t_1x_1 + t_2x_2 + \cdots + t_8x_8 \leq 10$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$

$v_1, v_2, \ldots, v_8$

$t_1, t_2, \ldots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

Constraints: $w_1x_1 + w_2x_2 + \cdots + w_8x_8 \leq 20$

$t_1x_1 + t_2x_2 + \cdots + t_8x_8 \leq 10$

$x_1 = (1 - x_2)$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \ldots, w_8$

$v_1, v_2, \ldots, v_8$

$t_1, t_2, \ldots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

Constraints: $w_1x_1 + w_2x_2 + \cdots + w_8x_8 \leq 20$

$t_1x_1 + t_2x_2 + \cdots + t_8x_8 \leq 10$

$\cancel{x_1 = (1 - x_2)}$

$x_1 + x_2 \leq 1$

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2, \dots, w_8$

$v_1, v_2, \dots, v_8$

$t_1, t_2, \dots, t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

$x_i = 1 \; if \; item \; i \; is \; taken, 0 \; otherwise$

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

Constraints: $w_1x_1 + w_2x_2 + \cdots + w_8x_8 \leq 20$

$t_1x_1 + t_2x_2 + \cdots + t_8x_8 \leq 10$

$\cancel{x_1 = (1 - x_2)}$

$x_1 + x_2 \leq 1$

http://cb.mty.itesm.mx/materias/tc3001/materiales/lsb-chap04.pdf

The Knapsack problem

Now what do we do? We will express the problem as a complete formulation in Jupyter using the gurobipy API.

Completed notebook on github here

The Knapsack problem formulation

1. **Data**
2. **Decision Variables**
3. **Objective Function**
4. **Constraints**

Data : $w_1, w_2,..., w_8$
$v_1, v_2,..., v_8$
$t_1, t_2,..., t_8$

Decision Variables : $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$
$x_i = 1$ *if item i is taken*, 0 *otherwise*

Objective Function: $Max(v_1x_1 + v_2x_2 + \cdots + v_8x_8)$

Constraints: $w_1x_1 + w_2x_2 + \cdots + w_8x_8 \leq 20$

$t_1x_1 + t_2x_2 + \cdots + t_8x_8 \leq 10$

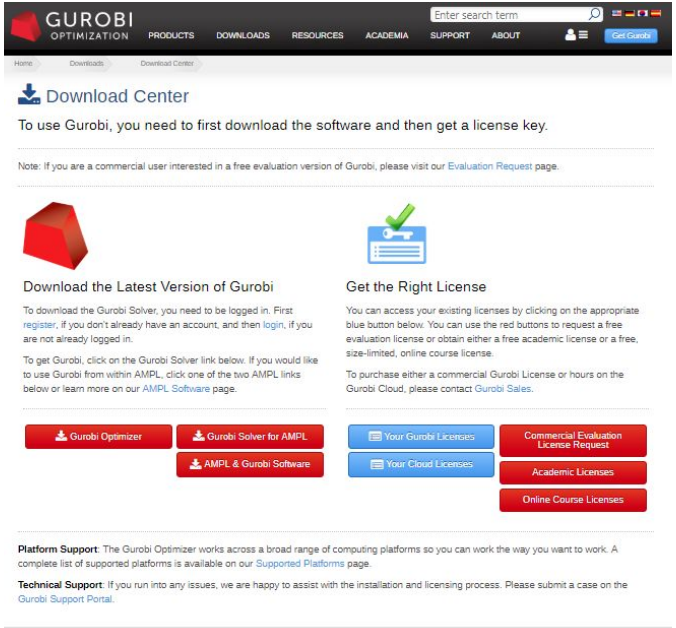
$\cancel{x_1 = (1 - x_2)}$

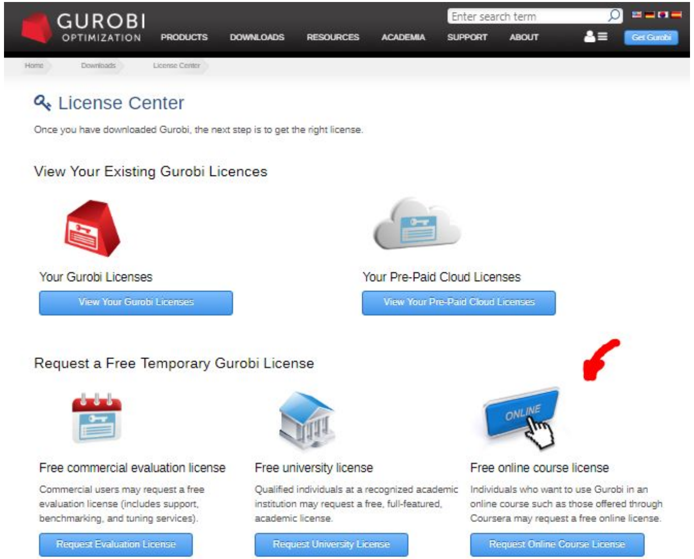
$x_1 + x_2 \leq 1$

```python
# imports
from gurobipy import *
m = Model('knapsack')

# data
item = ["AfricanGrey", "CalicoCat", "Clock", "Laptop",
        "Painting", "TV", "KeurigMachine", "MiniaturePincher"]
w= dict(zip(item,[5,8,3,3,1,7,5,7]))
v=dict(zip(item, [1200, 900, 100, 800, 400, 325, 250, 400]))
t=dict(zip(item, [2,6,1,1,3,4,2,1]))

# decision variables
x = m.addVars(item, vtype=GRB.BINARY, name="Steal")

# objective function
m.setObjective(x.prod(v), sense=GRB.MAXIMIZE)

# constraints
m.addConstr(x.prod(w)<=20)
m.addConstr(x.prod(t)<=10)
m.addConstr(x["AfricanGrey"] + x["CalicoCat"]<=1)
```

Note: To run the notebook you must have done the following

1. Installed Gurobi solver (Gurobi Optimizer). It's [here]
2. Installed a valid Gurobi License (the 'Online Course' version is the free version with limited functionality). It's [here]
3. The Gurobipy package/library installed (conda package is available thru anaconda)





Overall modeling process

Model

1. Variables for data
2. Mathematical relationships that need to be modeled
3. Objective function

Data

1. Data
   A. Costs
   B. Time
   C. Physical parameters on items
2. Decisions already made

Formulation

1. Data
2. Decision Variables
3. Objective Function
4. Constraints

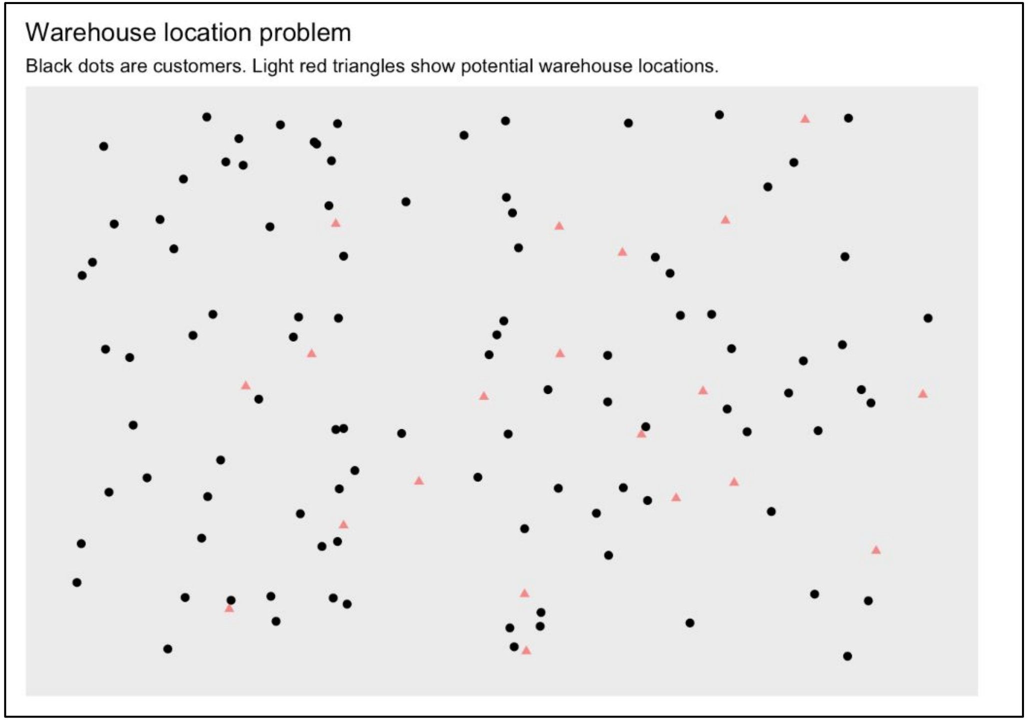
Solver

Solution Algorithm (branch and bound)

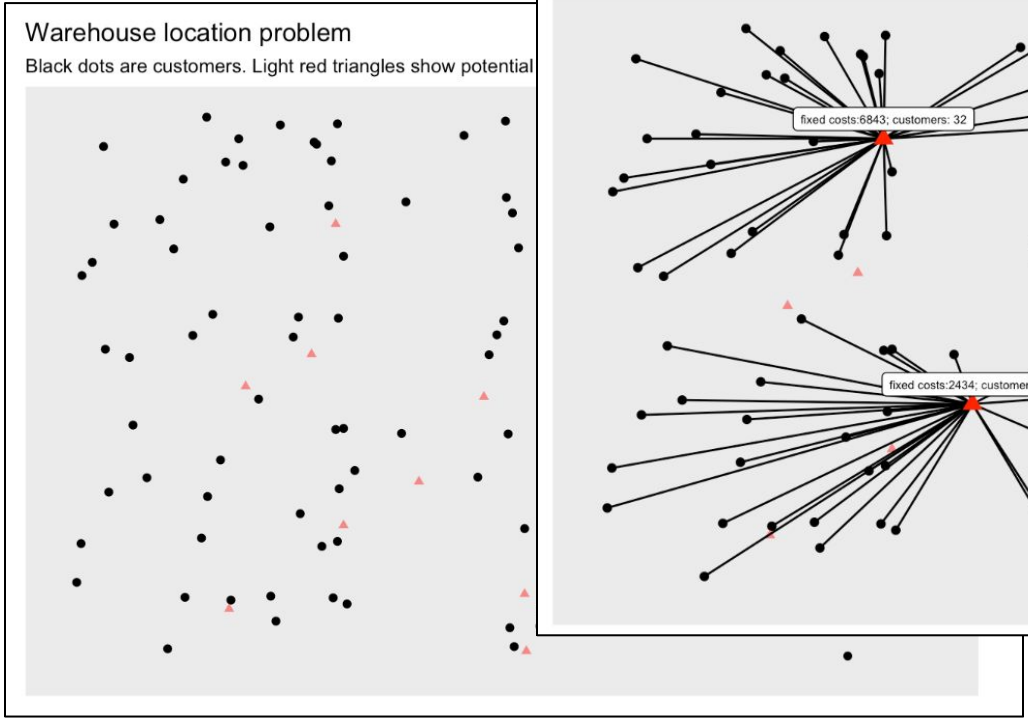
Solution

Values of the decision variables and value of objective function

Relevant Last Mile Optimization Model - Warehouse Location



Relevant Last Mile Optimization Model - Warehouse Location



Relevant Last Mile Optimization Model - Warehouse Location

We start with a set of customers $C = \{1 \dots n\}$ and a set of possible warehouses $W = \{1 \dots m\}$ that could be built. In addition we have a cost function giving us the transportation cost from a warehouse to a customer. Furthermore there is a fixed cost associated with each warehouse if it will be built. This basic version of the warehouse location problem is adapted from the German Wikipedia page about the problem.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} \text{transportcost}_{i,j} \cdot x_{i,j} + \sum_{j=1}^{m} \text{fixedcost}_j \cdot y_j & \\
\text{subject to} & \displaystyle\sum_{j=1}^{m} x_{i,j} = 1 & i = 1, \dots, n \\
& x_{i,j} \leq y_j, & i = 1, \dots, n \quad j = 1, \dots, m \\
& x_{i,j} \in \{0, 1\} & i = 1, \dots, n, \quad j = 1, \dots, m \\
& y_j \in \{0, 1\} & j = 1, \dots, m
\end{array}
$$

https://www.r-orms.org/mixed-integer-linear-programming/practicals/warehouse-location-problem/

Tips and Tricks (Testing)

Test Data Sets	Good For	Bad For
Prototype Data(small)	Validating Functionality Keeping Functionality Up(regression testing) Being Productive	Performance Testing Assessing Fit for use
Real World Data	Identifying enhancements, especially in usability Tracking Performance	Being Productive

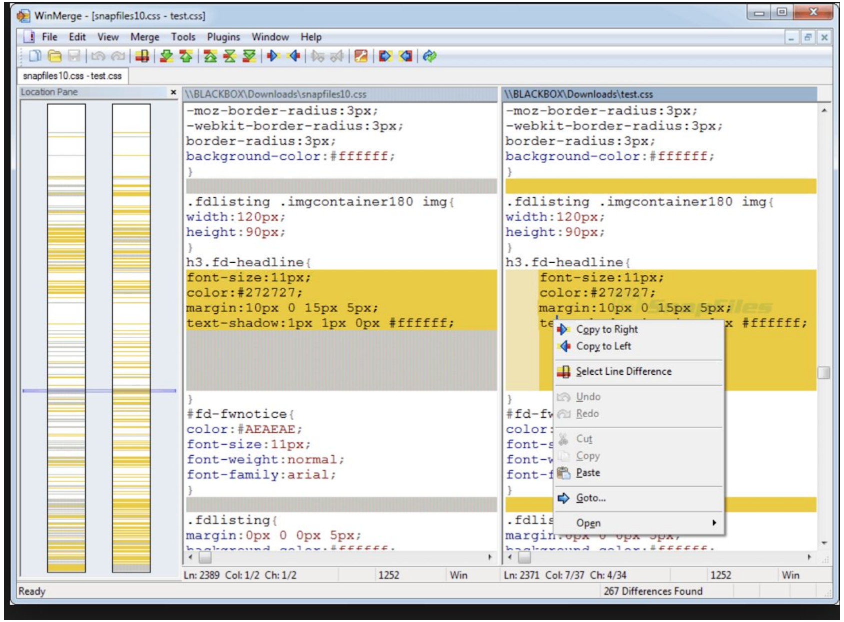
Tips and Tricks

Print out formulation (winmerge)

Version control (winmerge)

Results File (print out variable values that correspond to test data, compare with assert and winmerge)

Optimal value used in conjunction with results file (a good way to validate large changes in conjunction with a results file)



Additional examples can be found at Gurobi.com

To obtain a license for the Gurobi Solver and get up and running with the Gurobipy API, I recommend starting at the Gurobi documentation page.

https://www.gurobi.com/documentation/