15. Exponential Size Formulations

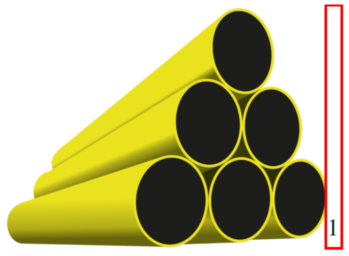
Although typically models need to be a reasonable size in order for us to code them and send them to a solver, there are some ways that we can allow having models of exponential size. The first example here is the cutting stock problem, where we will model with exponentially many variables. The second example is the traveling salesman problem, where we will model with exponentially many constraints. We will also look at some other models for the traveling salesman problem.

15.1 Cutting Stock

This is a classic problem that works excellent for a technique called *column generation*. We will discuss two versions of the model and then show how we can use column generation to solve the second version more efficiently. First, let's describe the problem.

Cutting Stock:

You run a company that sells of pipes of different lengths. These lengths are $L_1, \dots, L_k$. To produce these pipes, you have one machine that produces pipes of length L, and then cuts them into a collection of shorter pipes as needed.



You have an order come in for d_i pipes of length i for $i = 1, \dots, k$. How can you fill the order while cutting up the fewest number of pipes?

Example 15.1: Cutting stock with pipes

A plumber stocks standard lengths of pipe, all of length 19 m. An order arrives for:

- *12 lengths of 4m*
- *15 lengths of 5m*
- *22 lengths of 6m*

How should these lengths be cut from standard stock pipes so as to minimize the number of standard *pipes used?*

An initial model for this problem could be constructed as follows:

- Let N be an upper bound on the number of pipes that we may need.
- Let $z_j = 1$ if we use pipe i and $z_j = 0$ if we do not use pipe j, for $j = 1,\dots,N$.
- Let x_{ij} be the number of cuts of length L_i in pipe j that we use.

Then we have the following model

$$\begin{aligned}
\min \quad & \sum_{j=1}^{N} z_j \\
\text{s.t.} \quad & \sum_{i=1}^{k} L_i x_{ij} \le L z_j \ \text{ for } j = 1,\dots,N \\
& \sum_{j=1}^{N} x_{ij} \ge d_i \ \text{ for } i = 1,\dots,k \\
& z_j \in \{0,1\} \text{ for } j = 1,\dots,N \\
& x_{ij} \in \mathbb{Z}_+ \text{ for } i = 1,\dots,k,\ j = 1,\dots,N
\end{aligned} \tag{15.1}$$

Exercise 15.2: Show Bound

In the example above, show that we can choose $N = 16$.

For our example above, using $N = 16$, we have

$$\begin{aligned}
\min \quad & \sum_{j=1}^{16} z_j \\
\text{s.t.} \quad & 4x_{1j} + 5x_{2j} + 6x_{3j} \le 19 z_j \\
& \sum_{j=1}^{16} x_{1j} \ge 12 \\
& \sum_{j=1}^{16} x_{2j} \ge 15 \\
& \sum_{j=1}^{16} x_{3j} \ge 22 \\
& z_j \in \{0,1\} \text{ for } j = 1,\dots,16 \\
& x_{ij} \in \mathbb{Z}_+ \text{ for } i = 1,\dots,3,\ j = 1,\dots,16
\end{aligned} \tag{15.2}$$

Additionally, we could break the symmetry in the problem. That is, suppose the solution uses 10 of the 16 pipes. The current formulation does not restrict which 10 pipes are used. Thus, there are many possible solutions. To reduce this complexity, we can state that we only use the first 10 pipes. We can write a constraint that says *if we don't use pipe j, then we also will not use any subsequent pipes*. Hence, by not using pipe 11, we enforce that pipes 11, 12, 13, 14, 15, 16 are not used. This can be done by adding the constraints

$$z_1 \geq z_2 \geq z_3 \geq \cdots \geq z_N. \tag{15.3}$$

See **??** for code for this formulation.

Unfortunately, this formulation is slow and does not scale well with demand. In particular, the number of variables is $N + kN$ and the number of constraints is N (plus integrality and non-negativity constraints on the variables). The solution times for this model are summarized in the following table:

INPUT TABLE OF COMPUTATIONS

15.1.1. Pattern formulation

We could instead list all patterns that are possible to cut each pipe. A pattern is an vector $a \in \mathbb{Z}^k_+$ such that for each i, a_i lengths of L_i can be cut from a pipe of length L. That is

$$\begin{aligned} &\sum_{i=1}^k L_i a_i \leq L \\ &a_i \in \mathbb{Z}_+ \text{ for all } i = 1, \ldots, k \end{aligned} \tag{15.4}$$

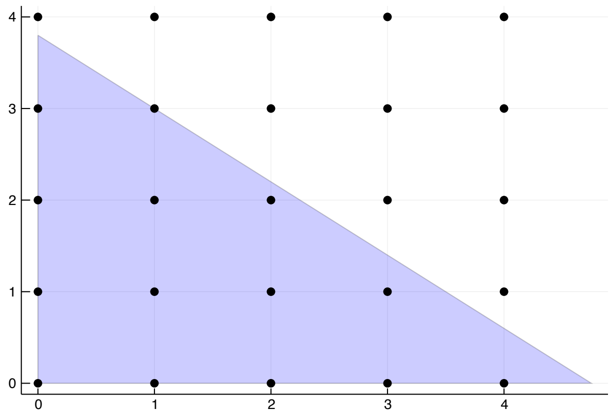
In our running example, we have

$$\begin{aligned} &4a_1 + 5a_2 + 6a_3 \leq 19 \\ &a_i \in \mathbb{Z}_+ \text{ for all } i = 1, \ldots, 3 \end{aligned} \tag{15.5}$$

For visualization purposes, consider the patterns where $a_3 = 0$. That is, only patterns with cuts of length 4m or 5m. All patterns of this type are represented by an integer point in the polytope

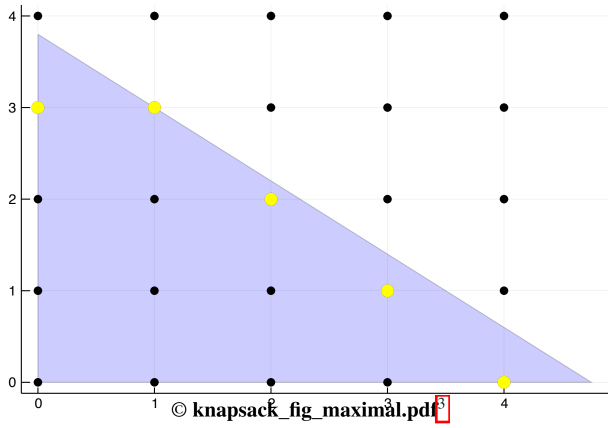
$$P = \{(a_1, a_2) : 4a_1 + 5a_2 \leq 19, a_1 \geq 0, a_2 \geq 0\} \tag{15.6}$$

which we can see here:



© **knapsack_fig.pdf**[2]

where P is the blue triangle and each integer point represents a pattern. Feasible patterns lie inside the polytope P. Note that we only need patterns that are maximal with respect to number of each type we cut. Pictorially, we only need the patterns that are integer points represented as yellow dots in the picture below.



© **knapsack_fig_maximal.pdf**[3]

For example, the pattern $[3,0,0]$ is not needed (only cut 3 of length 4m) since we could also use the patten $[4,0,0]$ (cut 4 of length 4m) or we could even use the pattern $[3,1,0]$ (cut 3 of length 4m and 1 of length 5m).

Example 15.3: Pattern Formulation

Let's list all the possible patterns for the cutting stock problem:

	Patterns									
Cuts of length 4m	0	0	1	0	2	1	2	3	4	1
Cuts of length 5m	0	1	0	2	1	2	2	1	0	3
Cuts of length 6m	3	2	2	1	1	0	0	0	0	0

We can organize these patterns into a matrix.

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 2 & 1 & 2 & 3 & 4 & 1 \\ 0 & 1 & 0 & 2 & 1 & 2 & 2 & 1 & 0 & 3 \\ 3 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{15.7}$$

Let p be the number of patterns that we have. We create variables $x_1, \dots, x_p \in \mathbb{Z}_+$ that denote the number of times we use each pattern.
Now, we can recast the optimization problem as

[2]**knapsack_fig.pdf**, from **knapsack_fig.pdf**. **knapsack_fig.pdf**, **knapsack_fig.pdf**.
[3]**knapsack_fig_maximal.pdf**, from **knapsack_fig_maximal.pdf**. **knapsack_fig_maximal.pdf**, **knapsack_fig_maximal.pdf**.

$$\min \sum_{i=1}^{p} x_i \tag{15.8}$$

$$\textit{such that } Ax \geq \begin{bmatrix} 12 \\ 15 \\ 22 \end{bmatrix} \tag{15.9}$$

$$x \in \mathbb{Z}_+^p \tag{15.10}$$

15.1.2. Column Generation

Consider the linear program(**??**), but in this case we are instead minimizing.

Thus we can write it as

$$\begin{aligned} \min \quad & (c_N - c_B A_B^{-1} A_N) x_N + c_B A_B^{-1} b \\ \text{s.t.} \quad & x_B + A_B^{-1} A_N x_N = A_B^{-1} b \\ & x \geq 0 \end{aligned} \tag{15.11}$$

In our LP we have $c = 1$, that is, $c_i = 1$ for all $i = 1, \dots, k$. Hence, we can write it as

$$\begin{aligned} \min \quad & (1_N - 1_B A_B^{-1} N) x_N + 1_B A_B^{-1} b \\ \text{s.t.} \quad & x_B + A_B^{-1} A_N x_N = A_B^{-1} b \\ & x \geq 0 \end{aligned} \tag{15.12}$$

Now, if there exists a non-basic variable that could enter the basis and improve the objective, then there is one with a reduced cost that is negative. For a particular non-basic variable, the coefficient on it is

$$\left(1 - 1_B A_B^{-1} A_N^i\right) x_i \tag{15.13}$$

where A_N^i is the i-th column of the matrix A_N. Thus, we want to look for a column a of A_N such that

$$1 - 1_B A_B^{-1} a < 0 \quad \Rightarrow \quad 1 < 1_B A_B^{-1} a \tag{15.14}$$

Pricing Problem:

(knapsack problem!)

Given a current basis B of the *master* linear program, there exists a new column to add to the basis that improves the LP objective if and only if the following problem has an objective value strictly larger than 1.

$$\begin{aligned} \max \quad & 1_B A_B^{-1} a \\ \text{s.t.} \quad & \sum_{i=1}^{k} L_i a_i \leq L \\ & a_i \in \mathbb{Z}_+ \text{ for } i = 1, \dots, k \end{aligned} \tag{15.15}$$

Example 15.4: Pricing Problem

Let's make the initial choice of columns easy. We will do this by selecting columns

	Patterns		
Cuts of length 4m	*4*	*0*	*0*
Cuts of length 5m	*0*	*3*	*0*
Cuts of length 6m	*0*	*0*	*3*

So our initial A matrix is

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{15.16}$$

Notice that there are enough patterns in the initial A matrix to produce feasible solution. Let's also append an arbitrary column to the A matrix as a potential new pattern.

$$A = \begin{pmatrix} 4 & 0 & 0 & a_1 \\ 0 & 3 & 0 & a_2 \\ 0 & 0 & 3 & a_3 \end{pmatrix} \tag{15.17}$$

Now, let's solve the linear relaxation and compute the tabluea.

$$\begin{aligned} \max \quad & ....a \\ s.t. \quad & 4a_1 + 5a_2 + 6a_3 \leq 19 \\ & a_i \in \mathbb{Z}_+ \text{ for } i = 1, \ldots, k \end{aligned} \tag{15.18}$$

15.1.3. Cutting Stock - Multiple widths

Resources

Gurobi has as excellent demonstration application to look at: Gurobi - Cutting Stock Demo Gurobi - Multiple Master Rolls

Here are some solutions:

- `https://github.com/fzsun/cutstock-gurobi`.
- `http://www.dcc.fc.up.pt/~jpp/mpa/cutstock.py`

Here is an AIMMS description of the problem: AIMMS Cutting Stock

15.2 Spanning Trees

Resources

*See [**Abdelmaguid2018**] for a list of 11 models for the minimum spanning tree and a comparison using CPLEX.*

15.3 Traveling Salesman Problem

Resources

See math.waterloo.ca for excellent material on the TSP.
See also this chapter A Practical Guide to Discrete Optimization.
Also, watch this excellent talk by Bill Cook "Postcards from the Edge of Possibility": Youtube!

Google Maps!

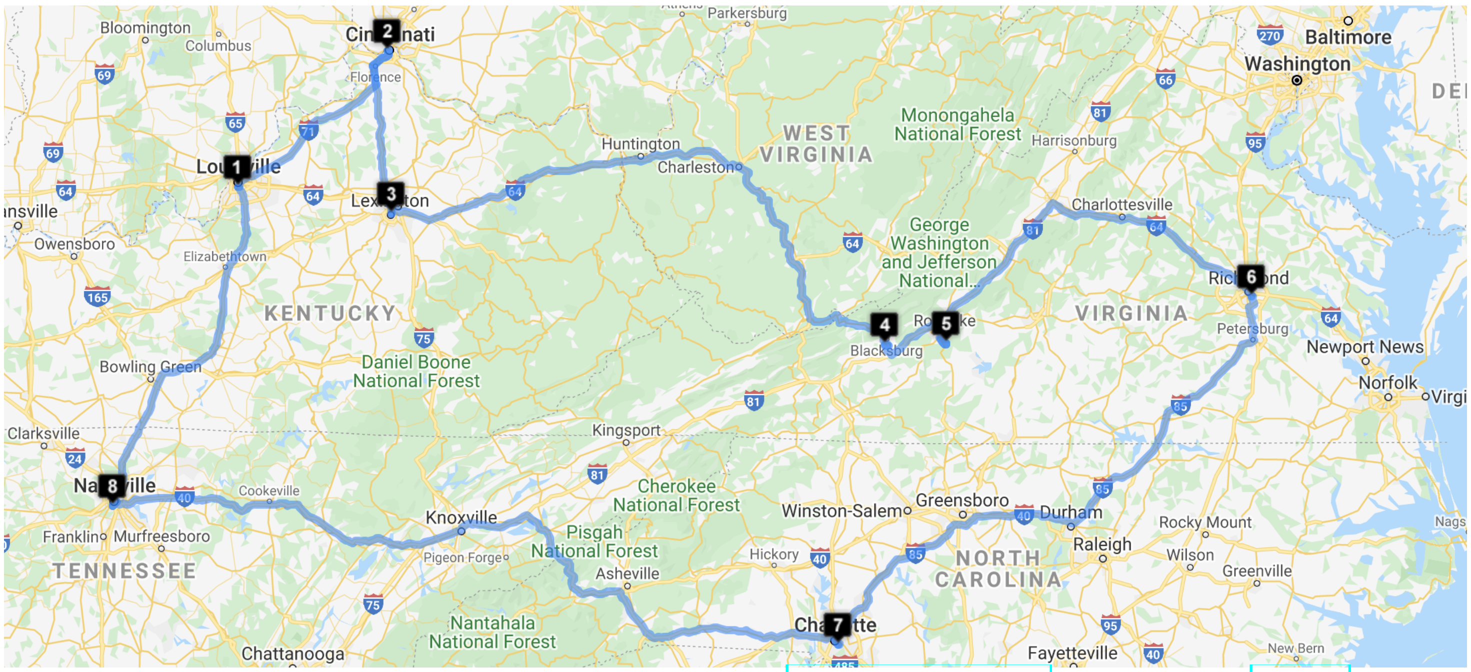


Figure 15.1: Optimal tour through 8 cities. Generated by Gebweb - Optimap. See it also on Google Maps!.

We consider a directed graph, graph $G = (N, A)$ of nodes N and arcs A. Arcs are directed edges. Hence the arc (i, j) is the directed path $i \to j$.

© wiki/File/William_Rowan_Hamilton_painting.jpg[4]

Figure 15.2: wiki/File/William_Rowan_Hamilton_painting.jpg

A *tour*, or Hamiltonian cycle (see Figure 15.2), is a cycle that visits all the nodes in N exactly once and returns back to the starting node.

Given costs c_{ij} for each arc $(i,j) \in A$, the goal is to find a minimum cost tour.

Traveling Salesman Problem:

NP-Hard

Given a directed graph $G = (N,A)$ and costs c_{ij} for all $(i,j) \in A$, find a tour of minimum cost.

ADD TSP FIGURE

In the figure, the nodes N are the cities and the arcs A are the directed paths city$i \to$ city j.

[4]**wiki/File/William_Rowan_Hamilton_painting.jpg**, from **wiki/File/William_Rowan_Hamilton_painting.jpg**. **wiki/File/William_Rowan_Hamilton_painting.jpg**, **wiki/File/William_Rowan_Hamilton_painting.jpg**.

MODELS When constructing an integer programming model for TSP, we define variables x_{ij} for all $(i,j) \in A$ as

$$x_{ij} = 1 \text{ if the arc } (i,j) \text{ is used and } x_{ij} = 0 \text{ otherwise.}$$

We want the model to satisfy the fact that each node should have exactly one incoming arc and one leaving arc. Furthermore, we want to prevent self loops. Thus, we need the constraints:

$$\sum_{j \in N} x_{ij} = 1 \qquad \text{for all } i \in N \quad \text{[outgoing arc]} \tag{15.1}$$

$$\sum_{i \in N} x_{ij} = 1 \qquad \text{for all } j \in N \quad \text{[incoming arc]} \tag{15.2}$$

$$x_{ii} = 0 \qquad \text{for all } i \in N \quad \text{[no self loops]} \tag{15.3}$$

Unfortunately, these constraints are not enough to completely describe the problem. The issue is that *subtours* may arise. For instance

ADD SUBTOURS FIGURE

15.3.1. Miller Tucker Zemlin (MTZ) Model

The Miller-Tucker-Zemlin (MTZ) model for the TSP uses varibles to mark the order for which cities are visited. This model introduce general integer variables to do so, but in the process, creates a formulation that has few inequalities to describe.

Some feature of this model:

- This model adds variables $u_i \in \mathbb{Z}$ with $1 \leq u_i \leq n$ that decide the order in which nodes are visited.
- We set $u_1 = 1$ to set a starting place.
- Crucially, this model relies on the following fact

Let x be a solution to (15.1)-(15.3) with $x_{ij} \in \{0,1\}$. If there exists a subtour in this solution that contains the node 1, then there also exists a subtour that does not contain the node 1.

The following model adds constraints

$$\text{If } x_{ij} = 1, \text{ then } u_i + 1 \leq u_j. \tag{15.4}$$

This if-then statement can be modeled with a big-M, choosing $M = n$ is a sufficient upper bound. Thus, it can be written as

$$u_i + 1 \leq u_j + n(1 - x_{ij}) \tag{15.5}$$

Setting these constraints to be active enforces the order $u_i < u_j$.

Consider a subtour now $2 \to 5 \to 3 \to 2$. Thus, $x_{25} = x_{53} = x_{32} = 1$. Then using the constraints from (15.5), we have that

$$u_2 < u_5 < u_3 < u_2, \tag{15.6}$$

but this is infeasible since we cannot have $u_2 < u_2$.

As stated above, if there is a subtour containing the node 1, then there is also a subtour not containing the node 1. Thus, we can enforce these constraints to only prevent subtours that don't contain the node 1. Thus, the full tour that contains the node 1 will still be feasible.

This is summarized in the following model:

Traveling Salesman Problem - MTZ Model:

$$\min \sum_{i,j \in N} c_{ij} x_{ij} \tag{15.7}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \text{for all } i \in N \quad \text{[outgoing arc]} \tag{15.8}$$

$$\sum_{i \in N} x_{ij} = 1 \qquad \text{for all } j \in N \quad \text{[incoming arc]} \tag{15.9}$$

$$x_{ii} = 0 \qquad \text{for all } i \in N \quad \text{[no self loops]} \tag{15.10}$$

$$u_i + 1 \leq u_j + n(1 - x_{ij}) \qquad \text{for all } i, j \in N, i, j \neq 1 \quad \text{[prevents subtours]} \tag{15.11}$$

$$u_1 = 1 \tag{15.12}$$

$$2 \leq u_i \leq n \qquad \text{for all } i \in N, i \neq 1 \tag{15.13}$$

$$u_i \in \mathbb{Z} \qquad \text{for all } i \in N \tag{15.14}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i, j \in N \tag{15.15}$$

Example 15.5: TSP with 4 nodes

Distance Matrix:

```
A \ B | 1  2  3  4
----------------------
1     | 0  1  2  3
2     | 1  0  1  2
3     | 2  1  0  4
4     | 3  2  4  0
```

Example 15.6: MTZ model for TSP with 4 nodes

Here is the full MTZ model:

$$\begin{aligned} \min \quad & x_{1,2} + 2x_{1,3} + 3x_{1,4} + x_{2,1} + x_{2,3} + 2x_{2,4} + \\ & 2x_{3,1} + x_{3,2} + 4x_{3,4} + 3x_{4,1} + 2x_{4,2} + 4x_{4,3} \end{aligned}$$

Subject to

$$\begin{aligned}
& x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1 && \textit{outgoing from node 1} \\
& x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1 && \textit{outgoing from node 2} \\
& x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1 && \textit{outgoing from node 3} \\
& x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} = 1 && \textit{outgoing from node 4} \\
\\
& x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} = 1 && \textit{incoming to node 1} \\
& x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} = 1 && \textit{incoming to node 2} \\
& x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} = 1 && \textit{incoming to node 3} \\
& x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} = 1 && \textit{incoming to node 4} \\
\\
& x_{1,1} = 0 && \textit{No self loop with node 1} \\
& x_{2,2} = 0 && \textit{No self loop with node 2} \\
& x_{3,3} = 0 && \textit{No self loop with node 3} \\
& x_{4,4} = 0 && \textit{No self loop with node 4} \\
\\
& u_1 = 1 && \textit{Start at node 1} \\
& 2 \leq u_i \leq 4, \quad \forall i \in \{2,3,4\} \\
& u_2 + 1 \leq u_3 + 4(1 - x_{2,3}) \\
& u_2 + 1 \leq u_4 + 4(1 - x_{2,4}) \leq 3 \\
& u_3 + 1 \leq u_2 + 4(1 - x_{3,2}) \leq 3 \\
& u_3 + 1 \leq u_4 + 4(1 - x_{3,4}) \leq 3 \\
& u_4 + 1 \leq u_2 + 4(1 - x_{4,2}) \leq 3 \\
& u_4 + 1 \leq u_3 + 4(1 - x_{4,3}) \leq 3 \\
& x_{i,j} \in \{0,1\} \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\} \\
& u_i \in \mathbb{Z}, \quad \forall i \in \{1,2,3,4\}
\end{aligned}$$

Example 15.7: MTZ model for TSP with 5 nodes

$$
\begin{aligned}
\min\quad & x_{1,2}+2x_{1,3}+3x_{1,4}+4x_{1,5}+x_{2,1}+x_{2,3}+2x_{2,4}+2x_{2,5}+2x_{3,1}+\\
& x_{3,2}+4x_{3,4}+x_{3,5}+3x_{4,1}+2x_{4,2}+4x_{4,3}+2x_{4,5}+\\
& 4x_{5,1}+2x_{5,2}+x_{5,3}+2x_{5,4}\\
\textit{Subject to}\quad & x_{1,1}+x_{1,2}+x_{1,3}+x_{1,4}+x_{1,5}=1\\
& x_{2,1}+x_{2,2}+x_{2,3}+x_{2,4}+x_{2,5}=1\\
& x_{3,1}+x_{3,2}+x_{3,3}+x_{3,4}+x_{3,5}=1\\
& x_{4,1}+x_{4,2}+x_{4,3}+x_{4,4}+x_{4,5}=1\\
& x_{5,1}+x_{5,2}+x_{5,3}+x_{5,4}+x_{5,5}=1\\
\\
& x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1}+x_{5,1}=1\\
& x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{5,2}=1\\
& x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{5,3}=1\\
& x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{5,4}=1\\
& x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}+x_{5,5}=1\\
\\
& x_{1,1}=0\\
& x_{2,2}=0\\
& x_{3,3}=0\\
& x_{4,4}=0\\
& x_{5,5}=0\\
\\
& u_1=1\\
& 2\le u_i\le 5\quad \forall i\in\{1,2,3,4,5\}\\
& u_2+1\le u_3+5(1-x_{2,3})\\
& u_2+1\le u_4+5(1-x_{2,4})\\
& u_2+1\le u_5+5(1-x_{2,5})\\
& u_3+1\le u_2+5(1-x_{3,2})\\
& u_3+1\le u_4+5(1-x_{3,4})\\
& u_4+1\le u_2+5(1-x_{4,2})\\
& u_4+1\le u_3+5(1-x_{4,3})\\
& u_3+1\le u_5+5(1-x_{3,5})\\
& u_4+1\le u_5+5(1-x_{4,5})\\
& u_5+1\le u_2+5(1-x_{5,2})\\
& u_5+1\le u_3+5(1-x_{5,3})\\
& u_5+1\le u_4+5(1-x_{5,4})\\
& x_{i,j}\in\{0,1\}\quad \forall i\in\{1,2,3,4,5\}, j\in\{1,2,3,4,5\}\\
& u_i\in\mathbb{Z},\quad \forall i\in\{1,2,3,4,5\}
\end{aligned}
$$

PROS OF THIS MODEL

- Small description
- Easy to implement

CONS OF THIS MODEL

- Linear relaxation is not very tight. Thus, the solver may be slow when given this model.

Example 15.8: Subtour elimation constraints via MTZ model

Consider the subtour $2 \to 4 \to 5 \to 2$.
For this subtour to exist in a solution, we must have

$$\begin{aligned} x_{2,4} &= 1 \\ x_{4,5} &= 1 \\ x_{5,2} &= 1. \end{aligned}$$

Consider the three corresponding inequalities to these variables:

$$\begin{aligned} u_2 + 1 &\leq u_4 + 5(1 - x_{2,4}) \\ u_4 + 1 &\leq u_5 + 5(1 - x_{4,5}) \\ u_5 + 1 &\leq u_2 + 5(1 - x_{5,2}). \end{aligned}$$

Since $x_{2,4} = x_{4,5} = x_{5,2} = 1$, *these reduce to*

$$\begin{aligned} u_2 + 1 &\leq u_5 \\ u_4 + 1 &\leq u_5 \\ u_5 + 1 &\leq u_2. \end{aligned}$$

Now, lets add these inequalities together. This produces the inequality

$$u_2 + u_4 + u_5 + 3 \leq u_2 + u_4 + u_5,$$

which reduces to

$$3 \leq 0.$$

This inequality is invalid, and hence no solution can have the values $x_{2,4} = x_{4,5} = x_{5,2} = 1$.

Example 15.9: Weak Model

Consider again the same tour in the last example, that is, the subtour $2 \to 4 \to 5 \to 2$. We are interested to know how strong the inequalties of the problem description are if we allow the variables to be continuous variables. That is, suppose we relax $x_{ij} \in \{0,1\}$ to be $x_{ij} \in [0,1]$.
Consider the inequalities related to this tour:

$$\begin{aligned} u_2 + 1 &\leq u_4 + 5(1 - x_{2,4}) \\ u_4 + 1 &\leq u_5 + 5(1 - x_{4,5}) \\ u_5 + 1 &\leq u_2 + 5(1 - x_{5,2}). \end{aligned}$$

A valid solution to this is

$$\begin{aligned} u_2 &= 2 \\ u_4 &= 3 \\ u_5 &= 4 \end{aligned}$$

$$\begin{aligned} 3 &\leq 3 + 5(1 - x_{2,4}) \\ 4 &\leq 4 + 5(1 - x_{4,5}) \\ 5 &\leq 2 + 5(1 - x_{5,2}). \end{aligned}$$

$$\begin{aligned} 0 &\leq 1 - x_{2,4} \\ 0 &\leq 1 - x_{4,5} \\ 3/5 &\leq 1 - x_{5,2}. \end{aligned}$$

$$\begin{aligned} 2 + 1 &\leq 3 + 5(1 - x_{2,4}) \\ 3 + 1 &\leq 4 + 5(1 - x_{4,5}) \\ 4 + 1 &\leq 2 + 5(1 - x_{5,2}). \end{aligned}$$

15.3.2. Dantzig-Fulkerson-Johnson (DFJ) Model

Resources

- *Gurobi Modeling Example: TSP*

This model does not add new variables. Instead, it adds constraints that conflict with the subtours. For instance, consider a subtour

$$2 \to 5 \to 3 \to 2. \tag{15.16}$$

We can prevent this subtour by adding the constraint

$$x_{25} + x_{53} + x_{32} \leq 2 \tag{15.17}$$

meaning that at most 2 of those arcs are allowed to happen at the same time. In general, for any subtour S, we can have the *subtour elimination constraint*

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1 \qquad \text{Subtour Elimination Constraint.} \tag{15.18}$$

In the previous example with $S = \{(2,5),(5,3),(3,2)\}$ we have $|S| = 3$, where $|S|$ denotes the size of the set S.

This model suggests that we just add all of these subtour elimination constraints.

Traveling Salesman Problem - DFJ Model:

$$\min \sum_{i,j\in N} c_{ij}x_{ij} \tag{15.19}$$

$$\sum_{j\in N} x_{ij} = 1 \qquad \text{for all } i \in N \quad \text{[outgoing arc]} \tag{15.20}$$

$$\sum_{i\in N} x_{ij} = 1 \qquad \text{for all } j \in N \quad \text{[incoming arc]} \tag{15.21}$$

$$x_{ii} = 0 \qquad \text{for all } i \in N \quad \text{[no self loops]} \tag{15.22}$$

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1 \qquad \text{for all subtours } S \quad \text{[prevents subtours]} \tag{15.23}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i,j \in N \tag{15.24}$$

Distance Matrix:

```
A \ B | 1  2  3  4
----------------------
1     | 0  1  2  3
2     | 1  0  1  2
3     | 2  1  0  4
4     | 3  2  4  0
```

Example 15.10: DFJ Model for $n = 4$ nodes

$$\begin{aligned}
\min \quad & x_{1,2} + 2x_{1,3} + 3x_{1,4} + x_{2,1} + x_{2,3} + 2x_{2,4} \\
& + 2x_{3,1} + x_{3,2} + 4x_{3,4} + 3x_{4,1} + 2x_{4,2} + 4x_{4,3}
\end{aligned}$$

Subject to

$$\begin{aligned}
& x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1 && \textit{outgoing from node 1} \\
& x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1 && \textit{outgoing from node 2} \\
& x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1 && \textit{outgoing from node 3} \\
& x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} = 1 && \textit{outgoing from node 4} \\
\\
& x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} = 1 && \textit{incoming to node 1} \\
& x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} = 1 && \textit{incoming to node 2} \\
& x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} = 1 && \textit{incoming to node 3} \\
& x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} = 1 && \textit{incoming to node 4} \\
\\
& x_{1,1} = 0 && \textit{No self loop with node 1} \\
& x_{2,2} = 0 && \textit{No self loop with node 2} \\
& x_{3,3} = 0 && \textit{No self loop with node 3} \\
& x_{4,4} = 0 && \textit{No self loop with node 4} \\
\\
& x_{1,2} + x_{2,1} \leq 1 && S = [(1,2),(2,1)] \\
& x_{1,3} + x_{3,1} \leq 1 && S = [(1,3),(3,1)] \\
& x_{1,4} + x_{4,1} \leq 1 && S = [(1,4),(4,1)] \\
& x_{2,3} + x_{3,2} \leq 1 && S = [(2,3),(3,2)] \\
& x_{2,4} + x_{4,2} \leq 1 && S = [(2,4),(4,2)] \\
& x_{3,4} + x_{4,3} \leq 1 && S = [(3,4),(4,3)] \\
& x_{2,1} + x_{1,3} + x_{3,2} \leq 2 && S = [(2,1),(1,3),(3,2)] \\
& x_{1,2} + x_{2,3} + x_{3,1} \leq 2 && S = [(1,2),(2,3),(3,1)] \\
& x_{3,1} + x_{1,4} + x_{4,3} \leq 2 && S = [(3,1),(1,4),(4,3))] \\
& x_{1,3} + x_{3,4} + x_{4,1} \leq 2 && S = [(1,3),(3,4),(4,1)] \\
& x_{2,1} + x_{1,4} + x_{4,2} \leq 2 && S = [(2,1),(1,4),(4,2)] \\
& x_{1,2} + x_{2,4} + x_{4,1} \leq 2 && S = [(1,2),(2,4),(4,1)] \\
& x_{3,2} + x_{2,4} + x_{4,3} \leq 2 && S = [(3,2),(2,4),(4,3)] \\
& x_{2,3} + x_{3,4} + x_{4,2} \leq 2 && S = [(2,3),(3,4),(4,2)] \\
\\
& x_{i,j} \in \{0,1\} \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}
\end{aligned}$$

Example 15.11

Consider a graph on 5 nodes.
Here are all the subtours of length at least 3 and also including the full length tours.
Hence, there are many subtours to consider.

PROS OF THIS MODEL

- Very tight linear relaxation

CONS OF THIS MODEL

- Exponentially many subtours S possible, hence this model is too large to write down.

SOLUTION: ADD SUBTOUR ELIMINATION CONSTRAINTS AS NEEDED. WE WILL DISCUSS THIS IN A FUTURE SECTION ON *cutting planes* .

15.3.3. Traveling Salesman Problem - Branching Solution

We will see in the next section

1. That the constraint (15.1)-(15.3) always produce integer solutions as solutions to the linear relaxation.

2. A way to use branch and bound (the topic of the next section) in order to avoid subtours.

15.3.4. Traveling Salesman Problem Variants

15.3.4.1. Many salespersons (m-TSP)

m-Traveling Salesman Problem - DFJ Model:

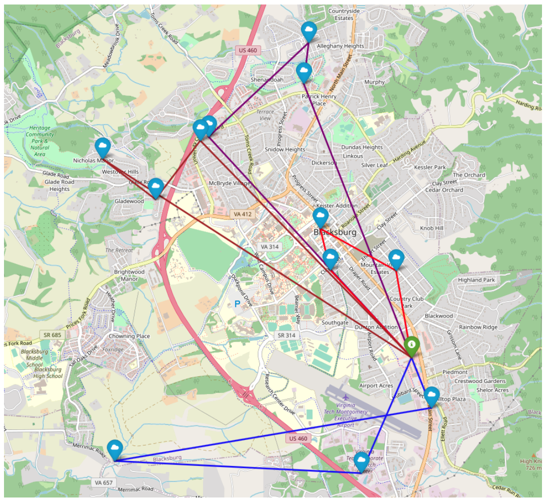
$$
\begin{aligned}
\min \sum_{i,j \in N} c_{ij} x_{ij} & & (15.25)\\
\sum_{j \in N} x_{ij} = 1 & \quad \text{for all } i \in N \quad \text{[outgoing arc]} & (15.26)\\
\sum_{i \in N} x_{ij} = 1 & \quad \text{for all } j \in N \quad \text{[incoming arc]} & (15.27)\\
\sum_{j \in N} x_{Dj} = m & \quad \text{for all } i \in N \quad \text{[outgoing arc]} & (15.28)\\
\sum_{i \in N} x_{iD} = m & \quad \text{for all } j \in N \quad \text{[incoming arc]} & (15.29)\\
x_{ii} = 0 & \quad \text{for all } i \in N \cup \{D\} \quad \text{[no self loops]} & (15.30)\\
\sum_{(i,j) \in S} x_{ij} \le |S| - 1 & \quad \text{for all subtours } S \subseteq N \quad \text{[prevents subtours]} & (15.31)\\
x_{ij} \in \{0,1\} & \quad \text{for all } i,j \in N \cup \{D\} & (15.32)
\end{aligned}
$$

When using the MTZ model, you can also easily add in a constraint that restricts any subtour through the deopt to have at most T stops on the tour. This is done by restricting $u_i \le T$. This could also be done in the DFJ model above, but the algorithm for subtour elimination cuts would need to be modified.

m-Traveling Salesman Problem - MTZ Model - :

Python Code

$$
\begin{aligned}
\min \sum_{i,j \in N} c_{ij} x_{ij} & & (15.33)\\
\sum_{j \in N} x_{ij} = 1 & \quad \text{for all } i \in N \quad \text{[outgoing arc]} & (15.34)\\
\sum_{i \in N} x_{ij} = 1 & \quad \text{for all } j \in N \quad \text{[incoming arc]} & (15.35)\\
\sum_{j \in N} x_{Dj} = m & \quad \text{for all } i \in N \quad \text{[outgoing arc]} & (15.36)\\
\sum_{i \in N} x_{iD} = m & \quad \text{for all } j \in N \quad \text{[incoming arc]} & (15.37)\\
x_{ii} = 0 & \quad \text{for all } i \in N \cup \{D\} \quad \text{[no self loops]} & (15.38)\\
u_i + 1 \le u_j + n(1 - x_{ij}) & \quad \text{for all } i,j \in N, i,j \ne 1 \quad \text{[prevents subtours]} & (15.39)\\
u_1 = 1 & & (15.40)\\
2 \le u_i \le T & \quad \text{for all } i \in N, i \ne 1 & (15.41)\\
u_i \in \mathbb{Z} & \quad \text{for all } i \in N & (15.42)\\
x_{ij} \in \{0,1\} & \quad \text{for all } i,j \in N \cup \{D\} & (15.43)
\end{aligned}
$$



© **m-tsp_solution**[5]

Image html

15.3.4.2. TSP with order variants

Using the MTZ model, it is easy to provide order variants. Such as, city 2 must come before city 3

$$u_2 \leq u_3$$

or city 2 must come directly before city 3

$$u_2 + 1 = u_3.$$

15.4 Vehicle Routing Problem (VRP)

The VRP is a generlization of the TSP and comes in many many forms. The major difference is now we may consider multiple vehicles visiting the around cities. Obvious examples are creating bus schedules and mail delivery routes.

Variations of this problem include

- Time windows (for when a city needs to be visited)
- Prize collecting (possibly not all cities need to be visited, but you gain a prize for visiting each city)
- Multi-depot vehicle routing problem (fueling or drop off stations)
- Vehicle rescheduling problem (When delays have been encountered, how do you adjust the routes)

[5]**m-tsp_solution**, from **m-tsp_solution**. **m-tsp_solution**, **m-tsp_solution**.

- Inhomogeneous vehicles (vehicles have different abilities (speed, distance, capacity, etc.).

To read about the many variants, see: Vehicle Routing: Problems, Methods, and Applications, Second Edition. Editor(s): Paolo Toth and Daniele Vigo. MOS-SIAM Series on Optimization.

For one example of a VRP model, see GUROBI Modeling Examples - technician routing scheduling.

15.4.1. Case Study: Bus Routing in Boston

Review this case study after studying algorihtms and heuristics for integer programming.

15.4.2. An Integer Programming Model

ILP

$$\begin{aligned} c_{ij} = &\text{ cost of travel } i \text{ to } j \\ x_{ijk} = &\, 1 \text{ iff vehicle } k \text{ travels} \\ &\text{directly from } i \text{ to } j \end{aligned}$$

subject to

$$\begin{array}{rll} \min \sum_{i,j} c_{ij} \sum_k x_{ijk} & & \\ \sum_i \sum_k x_{ijk} = 1 & \forall j \neq \text{ depot} & \text{Exactly one vehicle in} \\ \sum_j \sum_k^k x_{ijk} = 1 & \forall i \neq \text{ depot} & \text{Exactly one vehicle out} \\ \sum_i \sum_k x_{ihk} - \sum_j \sum_k x_{hjk} = 0 & \forall k, h & \text{It's the same vehicle} \\ \sum_i q_i \sum_j x_{ijk} \leq Q_k \quad \forall k & \text{Capacity constraint} & \\ \sum_{ijk} x_{ijk} = |S| - 1 \forall S \subseteq P(N), 0 \notin S & \text{Subtour elimination} & \\ x_{ijk} \in \{0,1\} & & \end{array}$$

15.4.3. Clark Wright Algorithm

Borrowed from `https://www.researchgate.net/publication/285833854_Chapter_4_Heuristics_for_the_Vehicle_Routing_Problem`

The Clarke and Wright Savings Heuristic The Clarke and Wright heuristic [12] initially constructs back and forth routes $(0,i,0)$ for $(i = 1,\dots,n)$ and gradually merges them by applying a saving criterion. More specifically, merging the two routes $(0,\dots,i,0)$ and $(0,j,\dots,0)$ into a single route $(0,\dots,i,j,\dots,0)$ generates a saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. Since the savings remain the same throughout the algorithm, they can be computed a priori. In the so-called parallel version of the algorithm which appears to be the best (see Laporte and Semet [46]), the feasible route merger yielding the largest saving is implemented at each iteration, until no more merger is feasible. This simple algorithm possesses the advantages of being intuitive, easy to implement, and fast. It is often used to generate an initial solution in more sophisticated algorithms. Several enhancements and acceleration procedures have been proposed for this algorithm (see, e.g., Nelson et al. [59] and Paessens [62]), but given the speed of today's computers and the robustness of the latest metaheuristics, these no longer seem justified.

[12] G. CLARKE AND J. W. WRIGHT, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, 12 (1964), pp. 568-581.

Resources

https://www.informs.org/Impact/O.R.-Analytics-Success-Stories/Optimized-school-bus-routing-helps-school-districts-design-better-policies
https://pubsonline.informs.org/doi/abs/10.1287/inte.2019.1015
https://www.informs.org/Resource-Center/Video-Library/Edelman-Competition-Videos/2019-Edelman-Competition-Videos/2019-Edelman-Finalist-Boston-Public-Schools
https://www.youtube.com/watch?v=LFeeaNP_rbY
Fantastic talk - Very thorough
https://www.opendoorlogistics.com/tutorials/tutorial-v-vehicle-routing-scheduling/

15.5 Steiner Tree Problem

Model 1

$$\min \sum_{(u,v)\in E} w_{uv} x_{uv}$$

such that

$$x_t = 1 \qquad \forall t \in T$$

$$2x_{uv} - x_u - x_v \leq 0 \qquad \forall (u,v) \in E$$

$$x_v - \sum_{(u,v)\in E} x_{uv} \leq 0 \forall v \in V$$

$$\sum_{(u,v)\in\delta(S)} x_{uv} \geq x_w \forall S \subseteq V, \forall w \in S$$

Model 2

$$
\begin{aligned}
\min \sum_{(u,v)\in E} w_{uv}x_{uv} & \\
x_t = 1 & \qquad \forall t \in T \\
2x_{uv} - x_u - x_v \le 0 & \qquad \forall (u,v) \in E \\
x_v - \sum_{(u,v)\in E} x_{uv} \le 0 & \qquad \forall v \in V \\
x_{uv} + x_{vu} \le 1 & \\
\sum_{v\in V} x_v - \sum_{(u,v)\in E} x_{uv} = 1 & \\
nx_{uv} + l_v - l_u \ge 1 - n(1 - x_{vu}) & \qquad \forall (u,v) \in E \\
nx_{vu} + l_u - l_v \ge 1 - n(1 - x_{uv}) & \qquad \forall (u,v) \in E
\end{aligned}
$$

```
Optimizing the shop footprint:
Step 1: Machine learning model predicts effect of opening or closing shops
Step 2: ILP Breaks down shop into manageable slusters
Step 3:  ILP for optimal footprint planning
```

15.6 Literature and other notes

- Gilmore-Gomory Cutting Stock [**Gilmore-Gomory**]
- A Column Generation Algorithm for Vehicle Scheduling and Routing Problems
- The Integrated Last-Mile Transportation Problem
- `http://www.optimization-online.org/DB_FILE/2017/11/6331.pdf`A BRANCH-AND-PRICE ALGORITHM FOR CAPACITATED HYPERGRAPH VERTEX SEPARATION

15.6.1. Google maps data

Blog - Python | Calculate distance and duration between two places using google distance matrix API

15.6.2. TSP In Excel

TSP with excel solver