ISyE/Math/CS 728
Integer Optimization

Integer Programming Models

Prof. Jim Luedtke
University of Wisconsin-Madison

Significant use from the book *Integer Programming* by M. Conforti, G. Cornuéjols, and G. Zambelli

Outline

The Knapsack Problem (2.1)
- Comparing Formulations (2.2)

Modeling Fixed Charges (2.10)

Modeling with a huge number of constraints
- The Traveling Salesman Problem (2.7)
- Solving models with a huge number of constraints

Modeling Disjunctions (2.11)

Binary Quadratic Optimizatiron and Fortet's Linearization

Packing, covering, partitioning

The Knapsack Problem (2.1)

The knapsack problem

- We are given a knapsack that can carry a maximum weight $b > 0$.
- There are n types of items that we could take.
- An item of type i has weight $a_i > 0$.
- We want to load the knapsack with items (possibly several items of the same type).

How do we model it?

- Let a variable x_i represent the number of items of type i to be loaded.
- The knapsack set S contains all the feasible loads:

$$S := \left\{ x \in \mathbb{Z}^n : \sum_{i=1}^{n} a_i x_i \leq b,\ x \geq 0 \right\}.$$

- If an item of type i has value c_i, the problem of maximizing the value of the load is the knapsack problem:

$$\max \left\{ \sum_{i=1}^{n} c_i x_i : x \in S \right\}.$$

How do we model it?

- If only one unit of each item type can be selected, we use binary variables instead of general integers.
- The $\underline{0,1 \text{ knapsack set:}}$

$$K := \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} a_i x_i \leq b \right\}.$$

- The $\underline{0,1 \text{ knapsack problem:}}$

$$\max\{cx : x \in K\}.$$

Minimal cover formulation

- Consider the $0,1$ knapsack set

$$K := \left\{x \in \{0,1\}^n : \sum_{i=1}^{n} a_i x_i \le b\right\}.$$

- A subset C of indices $\{1,\dots,n\}$ is a <u>cover</u> for K if

$$\sum_{i \in C} a_i > b.$$

- It is a <u>minimal cover</u> if

$$\sum_{i \in C \setminus \{j\}} a_i \le b \qquad \text{for every } j \in C.$$

Minimal cover formulation

Consider the set

$$K^C := \Big\{ x \in \{0,1\}^n : \sum_{i \in C} x_i \leq |C| - 1, \ \forall \text{ minimal cover } C \Big\}.$$

> **Proposition 2.1.**
> The sets K and K^C coincide.

Let's prove it!

Question: What about the following set?

$$\Big\{ x \in \{0,1\}^n : \sum_{i \in C} x_i \leq |C| - 1, \ \forall \text{ cover } C \Big\}.$$

Different Formulations

The 0, 1 knapsack problem is

$$\max\{cx : x \in K\} = \max\{cx : x \in K^C\}.$$

$$K = \Big\{x \in \{0,1\}^n : \sum_{i=1}^{n} a_i x_i \le b\Big\},$$
$$K^C = \Big\{x \in \{0,1\}^n : \sum_{i \in C} x_i \le |C| - 1, \ \forall \text{ minimal cover } C\Big\}.$$

- The constraints that define K and K^C look quite different:
  - K is defined by a single inequality with nonnegative integer coefficients.
  - K^C is defined by many inequalities (their number may be exponential in n) whose coefficients are $0, 1$.
- Which of the two formulations is “better”?

Which formulation is better?

Assume we have two different representations of the same mixed integer set S:

$$\begin{aligned} S &= \{(x, y) : A_1 x + G_1 y \le b_1,\ x \text{ integral}\} \\ &= \{(x, y) : A_2 x + G_2 y \le b_2,\ x \text{ integral}\} \end{aligned}$$

Consider their standard linear relaxations:

$$\begin{aligned} P_1 &= \{(x, y) : A_1 x + G_1 y \le b_1\} \\ P_2 &= \{(x, y) : A_2 x + G_2 y \le b_2\} \end{aligned}$$

- If $P_1 \subset P_2$ the first representation is <u>better</u>.
- If $P_1 = P_2$ the two representations are <u>equivalent</u>.
- If $P_1 \setminus P_2$ and $P_2 \setminus P_1$ are both nonempty, the two representations are <u>incomparable</u>.
- A perfect formulation is better or equivalent to any other formulation.

Example 1

$$K := \{x \in \{0,1\}^3 : 3x_1 + 3x_2 + 3x_3 \leq 5\}$$

$$K^C := \left\{ x \in \{0,1\}^3 : \begin{array}{r} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \end{array} \right\}$$

- How do they compare?
- Let's look at their standard linear relaxation!

Example 1

$$P := \{x \in [0,1]^3 : 3x_1 + 3x_2 + 3x_3 \leq 5\}$$

$$P^C := \left\{ x \in [0,1]^3 : \begin{array}{r} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \end{array} \right\}$$

- In this example $P^C \subset P$. Why?

$P^C \subseteq P$: Summing the three inequalities in P^C we get

$$2x_1 + 2x_2 + 2x_3 \leq 3 \qquad \Rightarrow \qquad 3x_1 + 3x_2 + 3x_3 \leq \frac{9}{2} \leq 5.$$

$P^C \subset P$: $(1, \frac{2}{3}, 0) \in P \setminus P^C$.

- So the minimal cover formulation is better than the knapsack formulation.

Example 2

$$K := \{x \in \{0,1\}^3 : x_1 + x_2 + x_3 \le 1\}$$

$$K^C := \left\{ \begin{array}{ll} & x_1 + x_2 \le 1 \\ x \in \{0,1\}^3 : & x_1 + x_3 \le 1 \\ & x_2 + x_3 \le 1 \end{array} \right\}$$

Example 2

$$P := \{x \in [0,1]^3 : x_1 + x_2 + x_3 \leq 1\}$$

$$P^C := \left\{ x \in [0,1]^3 : \begin{array}{r} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \end{array} \right\}$$

- In this example $P \subset P^C$. Why?

$P \subseteq P^C$:
  - Summing $x_1 + x_2 + x_3 \leq 1$ with $x_3 \geq 0$ we get

$$x_1 + x_2 \leq 1.$$

  - Symmetrically, we obtain the other two inequalities in P^C.

$P \subset P^C$: $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) \in P^C \setminus P$.

- Thus the knapsack formulation is better than the minimal cover formulation.
- Other times the formulations are incomparable. Example?

How to compare formulations in general?

$$\begin{aligned} S &= \{(x, y) : A_1 x + G_1 y \le b_1,\ x \text{ integral}\} \\ &= \{(x, y) : A_2 x + G_2 y \le b_2,\ x \text{ integral}\} \end{aligned}$$

$$\begin{aligned} P_1 &= \{(x, y) : A_1 x + G_1 y \le b_1\} \\ P_2 &= \{(x, y) : A_2 x + G_2 y \le b_2\} \end{aligned}$$

Assume $P_1 \neq \emptyset$. $P_1 \subseteq P_2$ if and only if every inequality $ax + gy \le \beta$ in $A_2 x + G_2 y \le b_2$ is valid for P_1.

How to compare formulations in general?

$$\begin{aligned} S &= \{(x,y) : A_1 x + G_1 y \le b_1,\ x \text{ integral}\} \\ &= \{(x,y) : A_2 x + G_2 y \le b_2,\ x \text{ integral}\} \end{aligned}$$

$$P_1 = \{(x,y) : A_1 x + G_1 y \le b_1\}$$
$$P_2 = \{(x,y) : A_2 x + G_2 y \le b_2\}$$

Assume $P_1 \neq \emptyset$. $P_1 \subseteq P_2$ if and only if every inequality $ax + gy \le \beta$ in $A_2 x + G_2 y \le b_2$ is valid for P_1.

- How can we check this condition?

How to compare formulations in general?

$$\begin{aligned} S &= \{(x,y) : A_1 x + G_1 y \leq b_1,\ x \text{ integral}\} \\ &= \{(x,y) : A_2 x + G_2 y \leq b_2,\ x \text{ integral}\} \end{aligned}$$

$$\begin{aligned} P_1 &= \{(x,y) : A_1 x + G_1 y \leq b_1\} \\ P_2 &= \{(x,y) : A_2 x + G_2 y \leq b_2\} \end{aligned}$$

Assume $P_1 \neq \emptyset$. $P_1 \subseteq P_2$ if and only if every inequality $ax + gy \leq \beta$ in $A_2 x + G_2 y \leq b_2$ is valid for P_1.

Farkas' lemma

The inequality $ax + gy \leq \beta$ is valid for P_1 if and only if the following system is feasible

$$uA_1 = a,\ uG_1 = g,\ ub_1 \leq \beta,\ u \geq 0.$$

- This condition can be checked by solving linear programs!

How to compare formulations in general?

$$\begin{aligned} S &= \{(x,y) : A_1 x + G_1 y \le b_1,\ x \text{ integral}\} \\ &= \{(x,y) : A_2 x + G_2 y \le b_2,\ x \text{ integral}\} \end{aligned}$$

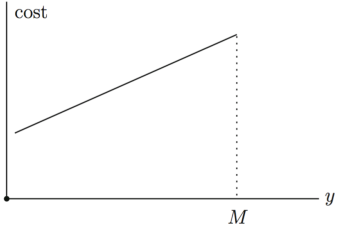
$$P_1 = \{(x,y) : A_1 x + G_1 y \le b_1\}$$
$$P_2 = \{(x,y) : A_2 x + G_2 y \le b_2\}$$

Assume $P_1 \neq \emptyset$. $P_1 \subseteq P_2$ if and only if every inequality $ax + gy \le \beta$ in $A_2 x + G_2 y \le b_2$ is valid for P_1.

Question: So how can you check in polynomial time if representations are better/equivalent/incomparable?

Modeling Fixed Charges (2.10)

Fixed Charges



- Economic activities frequently involve fixed and variable costs.
- Example: A production quantity.
  - Fixed cost if anything is produced (e.g., for setting up machines).
  - Variable cost linear in the amount produced (e.g., for operating machines).
- In this case, the cost associated with a certain variable $y \geq 0$ is
  - 0 when $y = 0$.
  - $c + hy$ when $y > 0$ (with $c, h > 0$).
- The cost is not linear in y.

Modeling Fixed Charges

- This situation can be modeled using a binary variable x s.t.

$$x = 1 \qquad \Longleftrightarrow \qquad y > 0.$$

Modeling Fixed Charges

- This situation can be modeled using a binary variable x s.t.

$$x = 1 \qquad \Longleftrightarrow \qquad y > 0.$$

- Then the cost of variable y can be written as

$$cx + hy.$$

Modeling Fixed Charges

- This situation can be modeled using a binary variable x s.t.

$$x = 1 \qquad \Longleftrightarrow \qquad y > 0.$$

- How do we model this relation?
- Let M be some upper bound on the value of variable y.

$$\begin{aligned} y &\leq Mx \\ x &\in \{0, 1\} \\ y &\geq 0 \end{aligned}$$

Problem: $x = 1$, $y = 0$ is feasible. But never optimal!

Modeling Fixed Charges

- This situation can be modeled using a binary variable x s.t.

$$x = 1 \qquad \Longleftrightarrow \qquad y > 0.$$

- How do we model this relation?
- Let M be some upper bound on the value of variable y.

$$\begin{aligned} &y \leq Mx \\ &x \in \{0, 1\} \\ &y \geq 0 \end{aligned}$$

Problem: $x = 1$, $y = 0$ is feasible. But never optimal!

Exercise: Prove that this formulation is valid!

You should prove:

1. For every "real-world solution", there exists a feasible solution to the formulation with cost equal or lower.
2. For every feasible solution to the formulation, there is a "real-world solution" with cost equal or lower.

Modeling Fixed Charges

- This situation can be modeled using a binary variable x s.t.

$$x = 1 \qquad \Longleftrightarrow \qquad y > 0.$$

- How do we model this relation?
- Let M be some upper bound on the value of variable y.

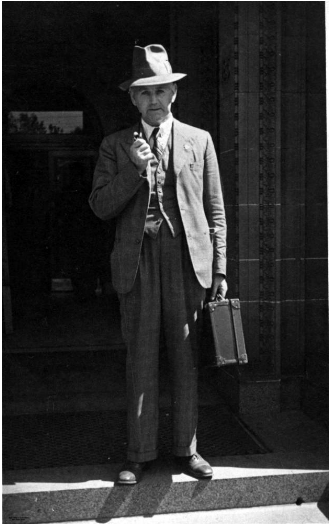
$$\begin{aligned} y &\leq Mx \\ x &\in \{0, 1\} \\ y &\geq 0 \end{aligned}$$

Problem: $x = 1$, $y = 0$ is feasible. But never optimal!

- Question: Does it matter how tight the upper bound M is?
- Linear programming relaxations of "big M" formulations tend to produce weak bounds in branch-and-bound.

Modeling with a huge number of constraints

The Traveling Salesman Problem (TSP)



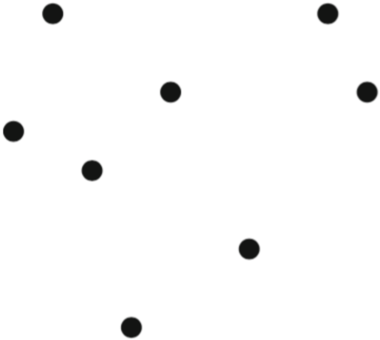
- A "traveling salesman" must visit n cities and return to the city he started from.
- Each city must be visited exactly once.
- The cost c_{ij} of traveling from city i to city j is given.
- In which order should the salesman visit the cities to minimize the cost of his tour?

The Traveling Salesman Problem

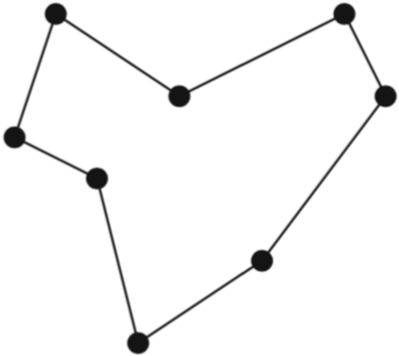
Many different versions:

- Asymmetric TSP:
  If we allow c_{ij} to be different from c_{ji}.
- Symmetric TSP:
  If $c_{ij} = c_{ji}$ for every pair of cities i and j.
- Metric TSP:
  If the costs satisfy the triangle inequality $c_{ij} \leq c_{ik} + c_{kj}$.
- Euclidean TSP:
  If the costs are distances in the plane.
- ...

Example of Euclidean TSP



Example of Euclidean TSP

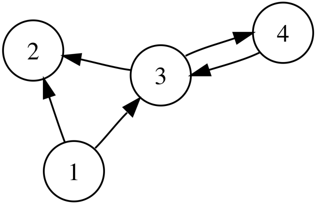


The Traveling Salesman Problem

To formulate the asymmetric TSP problem we will use digraphs.

A digraph is an ordered pair $D = (V, A)$, where

- V is a set of nodes.
- A is a set of arcs, which are ordered pairs of nodes.



$V = \{1, 2, 3, 4\}$,
$A = \{12, 13, 32, 34, 43\}$.

The Traveling Salesman Problem

Given:

- Digraph $D = (V, A)$.
- Costs c_a, for $a \in A$.

Find a minimum cost Hamiltonian cycle.

- Cycle: A sequence of arcs

$$v_1 v_2,\ v_2 v_3,\ v_3 v_4,\ \ldots,\ v_{k-1} v_k$$

  such that $v_k = v_1$ and each node is traversed at most once.
- Hamiltonian cycle: A cycle that traverses each node exactly once.

In general, D might not contain any Hamiltonian cycle!

TSP Formulation

Binary variables:

$x_{ij} = 1$ if and only if salesman goes directly from city i to city j for $a = ij$

Objective:

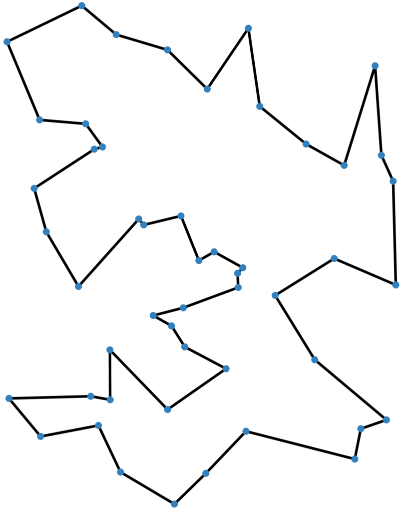
$$\min \sum_{a \in A} c_a x_a$$

Initial constraints (IC):

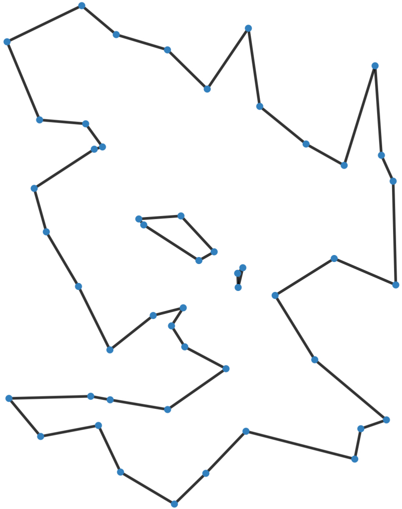
$$\begin{aligned} x_a &\in \{0,1\} && \forall a \in A && \text{binary constraints} \\ \sum_{ij \in A : j \neq i} x_{ij} &= 1 && \forall i \in V && \text{leave each city once} \\ \sum_{ji \in A : j \neq i} x_{ji} &= 1 && \forall i \in V && \text{enter each city once} \end{aligned}$$

Question: Is this formulation valid? I.e., is the optimal cost correct?

We want a Hamiltonian cycle.



Instead we might get two or more cycles.



How to eliminate cycles?

Subtour elimination constraints (SEC)
You must leave any set of nodes at least once:

$$\sum_{a \in \delta^{\text{out}}(S)} x_a \geq 1 \qquad \forall\, \emptyset \subset S \subset V$$

where $\delta^{\text{out}}(S) := \{ij \in A : i \in S,\ j \notin S\}$.

Let's prove that the formulation IC & SEC is valid!

We prove:

1. For every Hamiltonian cycle, its incidence vector x is feasible.
2. Every feasible solution x is the incidence vector of a Hamiltonian cycle.

How to eliminate cycles?

Subtour elimination constraints (SEC)
You must leave any set of nodes at least once:

$$\sum_{a \in \delta^{\text{out}}(S)} x_a \geq 1 \qquad \forall\, \emptyset \subset S \subset V$$

where $\delta^{\text{out}}(S) := \{ij \in A : i \in S,\ j \notin S\}$.

Let's prove that the formulation IC & SEC is valid!

We prove:

1. For every Hamiltonian cycle, its incidence vector x is feasible.
2. Every feasible solution x is the incidence vector of a Hamiltonian cycle.

Question: How many SEC are there?

How to eliminate cycles?

Subtour elimination constraints (SEC)

You must leave any set of nodes at least once:

$$\sum_{a \in \delta^{\text{out}}(S)} x_a \geq 1 \qquad \forall\, \emptyset \subset S \subset V$$

where $\delta^{\text{out}}(S) := \{ij \in A : i \in S,\ j \notin S\}$.

- This is the formulation that is most widely used in practice.
- Initially, one solves the standard linear programming relaxation of the initial constraints.
- The subtour elimination constraints are added later, on the fly, only when needed.
- This is possible because the separation problem can be solved efficiently for such constraints (see Chapter 4).

Separation problem

Separation problem

Let $P \subseteq \mathbb{R}^n$ be a polyhedron. The Separation Problem for P is: Given $\hat{x} \in \mathbb{R}^n$, is $\hat{x} \in P$? If not, find an inequality $\pi x \leq \pi_0$ satisfied by all points in P, but violated by $\hat{x}$.

Given a family (aka class) of inequalities (e.g., the subtour elimination constraints), we often speak about the separation problem over this family of inequalities:

- Given $\hat{x} \in \mathbb{R}^n$, does $\hat{x}$ satisfy all the inequalities in the family? If not, find an inequality in the family that is violated by $\hat{x}$.

How to solve a *linear program* with a huge number of constraints?

Theoreticaly efficient: Ellipsoid algorithm

"Theorem"
The Separation Problem for a convex set C can be solved in polynomial time if and only if the problem of optimizing a linear function over C can be solved in polynomial time.

- Key: does not matter if a polyhedron has a huge number of constraints, provided the *separation problem* can be solved efficiently
- The ellipsoid algorithm is used in both directions
- Unfortunately, the ellipsoid algorithm is not practically efficient

How to solve a *linear program* with a huge number of constraints?

Practically efficient (Usually): Cutting plane algorithm

0. Initialize LP with only a (small) subset of constraints (e.g., no subtour elim)
1. Solve current LP
2. Given LP optimal solution $\hat{x}$, solve a separation problem to see if $\hat{x}$ satisfies all the constraints of the model
   - If "Yes", the solution is optimal.
   - If "No", add a violated constraint to the LP, and return to step 1.

How to solve an *integer program* with huge number of constraints?

Option 1: Apply cutting-plane algorithm, except solve MIP instead of LP

0. Initialize MIP with only a (small) subset of constraints (e.g., no subtour elim)
1. Solve current MIP
2. Given integer feasible optimal solution $\hat{x}$, solve a separation problem to see if $\hat{x}$ satisfies all the constraints of the model
   - If "Yes", the solution is optimal.
   - If "No", add a violated constraint to the MIP, and return to step 1.

NB: Solving the separation problem is often much easier when the solutoin $\hat{x}$ is integer-valued

- TSP: detecting subtours when $\hat{x}$ integer can be done via a simple graph search over arcs selected by $\hat{x}$

How to solve an integer program with huge number of constraints?

Option 2: Solve a single MIP problem via branch-and-cut

- Start with a formulation that has a (small) subset of constraints
- At every integer feasible solution $\hat{x}$, solve a separation problem to see if $\hat{x}$ satisfies all the constraints of the model
  - If "Yes", the solution is *feasible*.
  - If "No", add a violated constraint as a lazy constraint, and continue.
  - At solutions that are not integer feasible, can *optionally* solve separation problem to improve LP relaxation

Which option to use?

Option 2 is generally more efficient, but somewhat more work to implement

- Requires implementing a callback routine within a commercial solver

How to eliminate cycles?

The subtour elimination constraints can be avoided!

Position constraints (PC)

Assume $V = \{1, \ldots, n\}$, and let u_i represent position of city $i \geq 2$ in the Hamiltonian cycle.

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) \qquad \forall ij \in A,\ i, j \neq 1$$

Let's prove that the formulation IC & PC is valid!

We prove:

- For every Hamiltonian cycle C, there exists a feasible solution (x, u) where x is the incidence vector of C.
- For every feasible solution (x, u), x is the incidence vector of a Hamiltonian cycle.

How to eliminate cycles?

How do these two formulations compare?

- We show the formulation with SEC's is better!
- Orirginal SEC's:

$$\sum_{a \in \delta^{\text{out}}(S)} x_a \geq 1 \qquad \forall\, \emptyset \subset S \subset V$$

- Equivalent formulation:

$$\sum_{ij \in A : i,j \in S} x_{ij} \leq |S| - 1 \qquad \forall\, \emptyset \subset S \subset V$$

To see equivalence, add constraints:

$$-\sum_{ij \in A : j \neq i} x_{ij} = -1$$

for $i \in S$ to the original SEC's

Which TSP formulation to use?

Formulation based on position variables:

- Easier ot implement

Formulation based on subtour constraints:

- Has fewer variables.
- Has more inequalities.
- Gives better bounds for brunch-and-cut.

The symmetric TSP

To formulate the symmetric TSP problem we will use graphs.

Given:

- Graph $G = (V, E)$.
- Costs c_e, for $e \in E$.

Find a minimum cost Hamiltonian cycle.

- Cycle: A sequence of edges

$$v_1 v_2,\ v_2 v_3,\ v_3 v_4,\ \ldots,\ v_{k-1} v_k$$

  such that $v_k = v_1$ and each node is traversed at most once.
- Hamiltonian cycle: A cycle that goes through each node exactly once.

The symmetric TSP

The Dantzig–Fulkerson–Johnson formulation for the symmetric TSP is:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s. t.} \quad & \sum_{e \in \delta(i)} x_e = 2 && \forall i \in V && \text{degree constraints} \\
& \sum_{e \in \delta(S)} x_e \geq 2 && \forall \emptyset \subset S \subset V && \text{SEC} \\
& x_e \in \{0, 1\} && \forall e \in E,
\end{aligned}
$$

where $\delta(S) := \{ij \in E : i \in S,\ j \notin S\}$.

The symmetric TSP

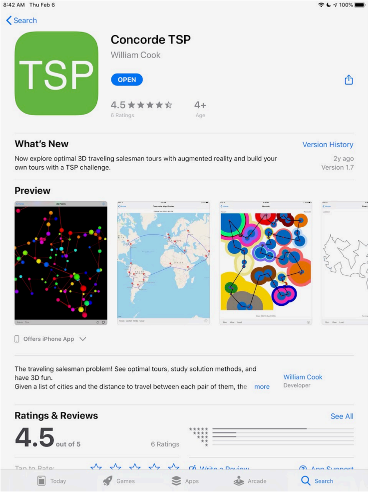
The Dantzig–Fulkerson–Johnson formulation for the symmetric TSP is:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(i)} x_e = 2 && \forall i \in V && \text{degree constraints} \\
& \sum_{e \in \delta(S)} x_e \geq 2 && \forall \emptyset \subset S \subset V && \text{SEC} \\
& x_e \in \{0, 1\} && \forall e \in E,
\end{aligned}
$$

- Exercise: Show that this formulation is valid.
- Despite its exponential number of constraints, this formulation is very effective in practice.

More TSP information

- For more information: http://www.math.uwaterloo.ca/tsp/index.html
- Successfully solved a TSP instance with 85,000 cities.
- The corresponding integer program has 3.5 billion binary variables!
- Play with Concorde TSP in the App Store.



Modeling Disjunctions (2.11)

Modeling Disjunctions

- Many applications have disjunctive constraints.
- For example, when scheduling jobs on a machine, we might need to model that either job i is scheduled before job j or vice versa.
- If p_i and p_j denote the processing times of these two jobs on the machine, we need a constraint stating that the starting times t_i and t_j of jobs i and j satisfy

$$t_j \geq t_i + p_i \quad \text{or} \quad t_i \geq t_j + p_j.$$

- In such applications, the feasible solutions lie in the union of two or more polyhedra.

Modeling Disjunctions

- How do we model that a point belongs to the union of k polytopes?
- Each polytope is a set of the form

$$\begin{aligned} A_i y &\leq b_i \\ 0 \leq y &\leq u_i \end{aligned}$$

  for $i = 1, \ldots, k$.
- To model the union of k polytopes in $\mathbb{R}^n$ we introduce:
  - k variables $x_i \in \{0, 1\}$, indicating whether y is in the ith polytope, and
  - k vectors of variables $y_i \in \mathbb{R}^n$, where y_i are the variables used for the ith polytope.

Modeling Disjunctions

The vector $y \in \mathbb{R}^n$ belongs to the union of the k polytopes $A_i y \leq b_i,\ 0 \leq y \leq u_i$ if and only if $\exists (y_1, \ldots, y_k, x_1, \ldots, x_k)$ s.t.

$$\begin{aligned} \sum_{i=1}^{k} y_i &= y \\ A_i y_i &\leq b_i x_i \qquad && i = 1, \ldots, k \\ 0 \leq y_i &\leq u_i x_i \qquad && i = 1, \ldots, k \\ \sum_{i=1}^{k} x_i &= 1 \\ x &\in \{0,1\}^k. \end{aligned} \tag{*}$$

We now show that this formulation is perfect!

Modeling Disjunctions

Proposition 2.6

The convex hull of solutions to (*) is

$$
\begin{aligned}
\sum_{i=1}^{k} y_i &= y \\
A_i y_i &\le b_i x_i \qquad && i = 1, \dots, k \\
0 \le y_i &\le u_i x_i \qquad && i = 1, \dots, k \\
\sum_{i=1}^{k} x_i &= 1 \\
x &\in [0,1]^k.
\end{aligned}
\tag{**}
$$

Modeling Disjunctions

Intuition behind Proposition 2.6.

- The convex hull of the k polytopes $A_i w \leq b_i,\ 0 \leq w \leq u_i$ can be written as:

$$
\begin{aligned}
\sum_{i=1}^{k} x_i w_i = y \quad &\iff \quad \sum_{i=1}^{k} y_i = y \\
A_i w_i \leq b_i \quad &\iff \quad A_i y_i \leq b_i x_i \\
0 \leq w_i \leq u_i \quad &\iff \quad 0 \leq y_i \leq u_i x_i \\
\sum_{i=1}^{k} x_i = 1 & \\
x \in [0,1]^k &
\end{aligned}
$$

- Replace w_i with $y_i := x_i w_i \quad \Longrightarrow \quad w_i = \frac{y_i}{x_i}$.

Let's prove Proposition 2.6 !

Binary Quadratic Optimizatiron and Fortet's Linearization

$0, 1$ polynomial program

- Consider the following $\underline{0, 1 \text{ polynomial program}}$

$$\begin{aligned} \min \quad & f(x) & \\ \text{s.t.} \quad & g_i(x) = 0 & \forall i = 1, \ldots, n \\ & x_j \in \{0, 1\} & \forall j = 1, \ldots, n \end{aligned}$$

  where the functions f and g_i, $i = 1, \ldots, m$, are polynomials.

- Such nonlinear functions can be linearized.

Proposition 2.7.
Any $0, 1$ polynomial program can be formulated as a pure $0, 1$ linear program by introducing additional variables.

Let's see how it's done!

$0, 1$ polynomial program: Example

$$f(x) = x_1^5 x_2 + 4 x_1 x_2 x_3^2.$$

- Function f is replaced by

$$f(x) = x_1 x_2 + 4 x_1 x_2 x_3.$$

- We introduce y_{12} in place of $x_1 x_2$:

$$f(x) = y_{12} + 4 y_{12} x_3.$$

- We introduce y_{123} in place of $y_{12} x_3$:

$$f(x) = y_{12} + 4 y_{123}.$$

- We impose linear constraints

$$\begin{aligned} y_{12} &\leq x_1 & \qquad y_{123} &\leq y_{12} \\ y_{12} &\leq x_2 & \qquad y_{123} &\leq x_3 \\ y_{12} &\geq x_1 + x_2 - 1 & \qquad y_{123} &\geq y_{12} + x_3 - 1. \end{aligned}$$

Packing, covering, partitioning

Packing, covering, partitioning

Data:

- Finite set $E := \{1, \ldots, n\}$
- $\mathcal{F} := \{F_1, \ldots, F_m\}$ family of subsets of E
- Weights $w_j, j = 1, \ldots, n$

Definitions:

- $S \subseteq E$ is a packing of $\mathcal{F}$ if S intersects each F_i at most once
- $S \subseteq E$ is a partitioning of $\mathcal{F}$ if S intersects each F_i exactly once
- $S \subseteq E$ is a covering of $\mathcal{F}$ if S intersects each F_i at least once

Packing, covering, partitioning (2)

Formulation for determining a set S: $x_j = 1$ if and only if $j \in S$

$$S^P := \left\{x \in \{0,1\}^n : \sum_{j \in F_i} x_j \leq 1, \forall F_i \in \mathcal{F}\right\},$$

$$S^T := \left\{x \in \{0,1\}^n : \sum_{j \in F_i} x_j = 1, \forall F_i \in \mathcal{F}\right\},$$

$$S^C := \left\{x \in \{0,1\}^n : \sum_{j \in F_i} x_j \geq 1, \forall F_i \in \mathcal{F}\right\}.$$

Set packing problem

$$\max\left\{\sum_{j=1}^{n} w_j x_j : x \in S^P\right\}$$

Set covering and partitioning problems similar, except max $\rightarrow$ min

Example: Stable sets in graphs

Definition

Given an undirected graph $G = (V, E)$, a stable set in G is a set of nodes no two of which are adjacent. (I.e., $S \subseteq V$ is a stable set if for any $ij \in E$, at most one of $i, j \in U$.)

Formulation as a set packing problem:

$$\mathrm{stab}(G) := \left\{x \in \{0,1\}^n : x_i + x_j \leq 1, \forall ij \in E\right\}$$

Example: Stable sets in graphs

Formulation as a set packing problem:

$$\mathrm{stab}(G) := \left\{x \in \{0,1\}^n : x_i + x_j \leq 1, \forall ij \in E\right\}$$

Strengthening the formulation:

Definition
Given an undirected graph $G = (V, E)$, a clique in G is a set of pairwise adjacent nodes. (I.e., $K \subseteq V$ is a clique if $ij \in E$ for all$i, j \in K$.)

If $K \subseteq V$ is a clique, then the following is a valid inequality for $\mathrm{stab}(G)$:

$$\sum_{i \in K} x_i \leq 1$$

Formulation with all maximal cliques is better

Another Example: Vehicle routing problem

Vehicle routing with a fixed number of trucks

- Customer set: V, with demand d_i for $i \in V$
  - Depot is node $0 \in V$, which has no demand
- Cost to travel between customers $i, j \in V$: c_{ij}
- Trucks: K identical trucks, each with capacity C
- Assign a set of customers to each truck, and find subtours visiting those customers, so that all customer demand is met at minimum cost
- Time window constraints
  - Travel time between customers i and j is t_{ij}, and each customer has a service time s_i
  - Each customer also has an earliest delivery time r_i and latest delivery time l_i

What would a "compact" formulation look like?

Decision variables:

- $z_{ik} = 1$ if customer i assigned to truck $k = 1, \ldots, K$
- $y_{ijk} = 1$ if truck k travels directly from customer i to j
- $t_{ik} =$ arrival time of truck k to customer i, if assigned to that customer

Constraints (in words)

- Each customer assigned to a truck
- Capacity of each truck
- Modified enter/leave constraints and subtour elimination constraints
- Constraints to determine arrival time of a truck to a customer
- Earliest and latest delivery time

Drawbacks of the “compact” formulation

- Not really so compact (although polynomial)
- LP relaxation is typically weak
- Formulation exhibits symmetry (trucks are identical)
- Adding still more constraints on a route makes model even more complex

Alternative modeling approach

Suppose we enumerate all feasible truck tours, indexed by $t = 1, \ldots, T$

- T may be very large, but might be manageable if tours are heavily constrained (e.g., at most 2-3 customers fit on a truck, or time windows are tough to combine)
- MIP solvers can sometimes handle T in hundreds of thousands!
- Later, we'll see branch-and-price to deal with T too large

Data associated with each tour t:

- c_t = total cost of tour t
- $a_{it} = 1$ if tour t serves customer i

That's it! All the rest of the data is "hidden" in the definition of the tours

Formulation: Set covering

Formulation is now simple: $x_t = 1$ if tour t is selected

$$\begin{aligned} \min \ & \sum_{t=1}^{T} c_t x_t \\ & \sum_{t=1}^{T} x_t \leq K \\ & \sum_{t=1}^{T} a_{it} x_t \geq 1, \quad \text{for all } i \in V \\ & x_t \in \{0,1\}, \quad t = 1, \dots, T \end{aligned}$$

Advantages of set covering formulation

- Complexity of tour constraints is hidden in definition of variables
- LP relaxation is typically very close to IP solution
- Symmetry is removed

This modeling approach is very commonly used transportation and scheduling problems