ILP modeling examples

Ben Rosenberg

January 6, 2026

Overview

- Knapsack
- Fixed-charge min cost flow
- Scheduling
- Assignment problem
- Set cover problem
- Vertex cover problem

Knapsack problem

In the knapsack problem, we have a knapsack with a certain capacity (maximum weight), and a set of items that we can choose to include. Each item has an associated weight and an associated utility (or benefit).

The goal is to maximize the total benefit, while staying at or under the capacity limit of the knapsack.

Three different types of problem:

- 0-1 knapsack: Each item can only be used at most once
- Bounded knapsack: We a bound B on the number of copies of each item
- Unbounded knapsack: We can have as many copies of each item type as we want (subject to the constraints)

Knapsack problem formulation

We will model all three types of the problem.

Given:

- Set of items I
- Associated weights $w(i)$ and utility values $u(i)$
- Total knapsack capacity C

Knapsack problem formulation (solution)

We'll create decision variables x_i for each $i \in I$, to indicate the amount of that item that we include in the knapsack.

The objective function will be the same for all of the problem variations:

$$\max \quad \sum_{i=1}^{n} u(i) \cdot x_i$$

So will the knapsack capacity constraint:

$$s.t. \quad \sum_{i=1}^{n} w(i) \cdot x_i \leq C$$

The only change across variations is in the definition of x_i:

- 0-1 knapsack problem: $x_i \in \{0, 1\}$
- Bounded knapsack: $x_i \in \{0, 1, \ldots, B\}$
- Unbounded knapsack: $x_i \in \mathbb{N}$

Fixed-charge min cost flow

Given a graph $G = (V, E)$, a capacity function $U : E \to \mathbb{N}$, a cost function $C : E \to \mathbb{N}$, a supply/demand function $S : V \to \mathbb{Z}$, and a **fixed-cost function** $F : E \to \mathbb{N}$ the **fixed-charge min cost flow** problem is the problem of determining an assignment of flow values $x_{i,j}$ for each edge $(i, j) \in E$ such that total cost is minimized, while obeying the supply/demand and capacity constraints given by S and U, and incorporating the fixed costs given by $F(i, j)$ if the flow $x_{i,j}$ from i to j is nonzero.

This problem has applications in shipping and elsewhere. For example, the LP project for airline MCF should realistically have had fixed costs; the cost of a 1-mile flight is not half the cost of a 2-mile flight, because you need to pay the ground crew the same amount.

Fixed-charge min cost flow (cont.)

We saw min cost flow before, which we modeled as an LP. This problem had the integrality property, meaning that all the extreme points corresponded to integer values for the decision variables, so we didn't need an ILP to solve it.

We will be introducing an additional complication to the problem that causes it to require the use of an ILP solver: *fixed costs*.

Each edge will have an additional fixed cost $F(i, j)$ that is only applied if a nonzero amount of flow is sent through the edge (i, j). If the edge is not used, the fixed cost should be 0.

Min cost flow review

This is the formulation for the unmodified MCF problem:

$$\min \sum_{(i,j)\in E} C(i,j) \cdot x_{i,j}, \quad \text{subject to:}$$

- Flow in equals flow out, adjusting for supply or demand:

$$\sum_{(i,j)\in E} x_{k,j} - \sum_{(i,k)\in E} x_{i,j} = S(k) \quad \forall k \in V$$

- Edge capacities are respected:

$$x_{i,j} \leq U(i,j) \quad \forall (i,j) \in E$$

- Edge flow cannot be negative:

$$x_{i,j} \geq 0 \quad \forall (i,j) \in E$$

Fixed-charge MCF formulation

We can incorporate the fixed cost by adding a decision variable $e_{i,j} \in \{0, 1\}$ to indicate whether an edge is used or not, and a big-M constraint to define the new decision variable in terms of $x_{i,j}$ being strictly greater than 0.

New objective:

$$\min \sum_{(i,j) \in E} C(i,j) \cdot x_{i,j} + F(i,j) \cdot e_{i,j}$$

New big-M constraint, to define $e_{i,j}$:

$$e_{i,j} \geq \frac{1}{M} \cdot x_{i,j}$$

Fixed-charge MCF formulation (cont.)

We also need to say that $x_{i,j}$ is an integer, because we no longer have the integrality property (the big-M constraint kills the total unimodularity of the constraint matrix).

If we do want to let $x_{i,j}$ take non-integer values we may need to increase M significantly to ensure that $\frac{1}{M}$ is a sufficiently small ϵ that the constraint is activated whenever $x_{i,j} > 0$, but as discussed this may cause numerical instability.

Scheduling

Typical scheduling problem:

- N tasks $t \in T$, each with a (constant) duration $D : T \to \mathbb{R}$
- M cores $c \in C$ (or tracks, or production lines, depending on application)
- Prerequisite relationships $t_i \prec t_j$, indicating that task t_i comes before task t_j

Objective: minimize total duration.

Scheduling formulation

Decision variables:

- $x_{t,c} \in \{0, 1\}$ to indicate that task t runs on core c
- s_t and e_t to hold start and end times of task t
- p_{t_1,t_2} to indicate whether (if t_1 and t_2 are on the same track) t_1 precedes t_2 or t_2 precedes t_1. This is used to prevent overlap
- L, to capture the total length (duration) of the process

Constraints:

- $\forall t \quad s_t + D(t) = e_t$: duration is respected
- $\forall t \quad \sum_{c \in C} x_{t,c} = 1$: a task is assigned to exactly one track
- Non-overlap constraint and total duration constraint

Scheduling formulation (cont.)

Non-overlap constraint:

- If t_1 and t_2 are running on the same core, then:
  - If t_1 comes before t_2, then $e_{t_1} \leq s_{t_2}$
  - Otherwise, $e_{t_2} \leq s_{t_1}$
- Otherwise, no constraint

First we will model the inner if-condition using big-M constraints:

$$e_{t_1} \leq s_{t_2} + M \cdot (1 - p_{t_1,t_2}) \qquad \forall t_1 \in T, t_2 \in T, t_1 \neq t_2$$

$$e_{t_2} \leq s_{t_1} + M \cdot p_{t_1,t_2} \qquad \forall t_1 \in T, t_2 \in T, t_1 \neq t_2$$

Scheduling formulation (cont.)

Now we need to ensure this is only applied if t_1 and t_2 are on the same core.

To do this we can add the following term to all of our big-M constraints, and extend their scope to be over all cores $c \in C$:

$$M \cdot (2 - x_{t_1,c} - x_{t_2,c})$$

For example, the first big-M constraint would look like this:

$$e_{t_1} \leq s_{t_2} + M \cdot (1 - p_{t_1,t_2}) + M \cdot (2 - x_{t_1,c} - x_{t_2,c})$$
$$\forall t_1 \in T, t_2 \in T, t_1 \neq t_2, \forall c \in C$$

If either of $x_{t_1,c}$ or $x_{t_2,c}$ is 0, the big-M is activated, which disables the constraint.

Scheduling formulation (cont.)

The decision variable p_{t_1,t_2} is used only for the big-M constraints among tasks run on the same core, not necessarily to capture prerequisite relationships.

Any specific prerequisite constraints among items not necessarily on the same core can be defined using ad-hoc constraints like so:

$$t_i \prec t_j \implies e_{t_i} \leq s_{t_j}$$

Scheduling formulation (cont.)

We need to ensure the total duration is correctly defined:

$$L \geq e_t \quad \forall t \in T$$

Then, our objective function becomes:

$$\min L$$

Assignment problem

Typical assignment problem:

- N tasks $t \in T$
- M people $p \in P$
- Costs $C(t, p)$ of assigning a task t to a person p

Each person is restricted to doing exactly one task, and all tasks need to be completed.

Assignment problem formulation

(Class exercise.)

Assignment problem formulation (solution)

$$
\begin{aligned}
\min \quad & \sum_{t \in T} \sum_{p \in P} x_{t,p} \cdot C(t,p) \\
s.t. \quad & \sum_{t \in T} x_{t,p} = 1 \quad \forall p \in P \\
& \sum_{p \in P} x_{t,p} = 1 \quad \forall t \in T \\
& x_{t,p} \in \{0,1\} \quad \forall t \in T, \forall p \in P
\end{aligned}
$$

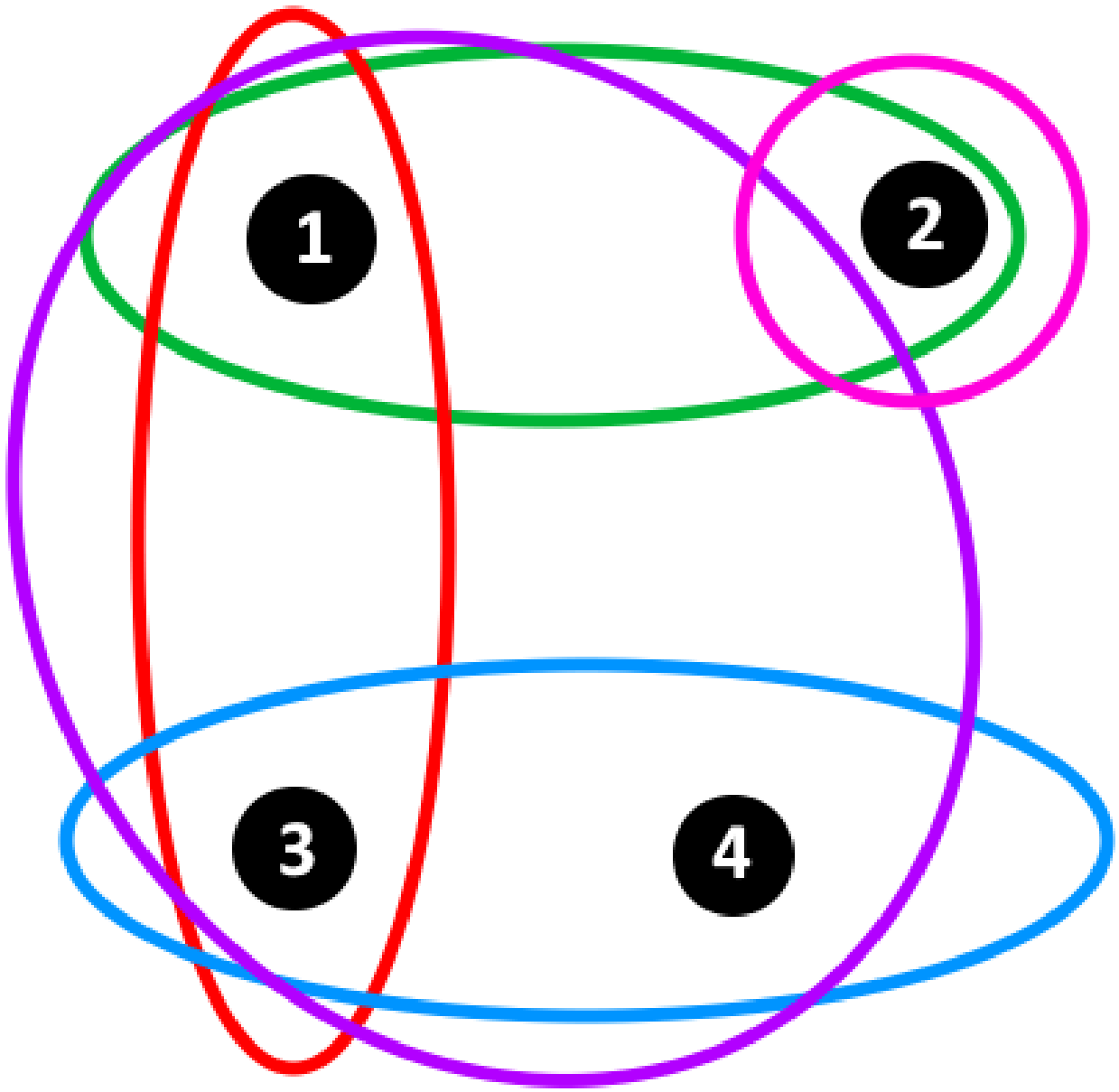
Set cover problem

In the set cover problem, we are given:

- U, a set of N elements $\{u_1, u_2, \ldots, u_N\}$ ("universe")
- S, a set of M subsets of U, $\{S_1, S_2, \ldots, S_M\}$ ("family of subsets")

The objective is to select the fewest possible subsets $S^* \subseteq S$ so that all elements in U are covered (that is, $U \subseteq \bigcup_{s \in S^*} s$).

Set cover example



Set cover problem formulation

(Class exercise.)

Set cover problem formulation (solution)

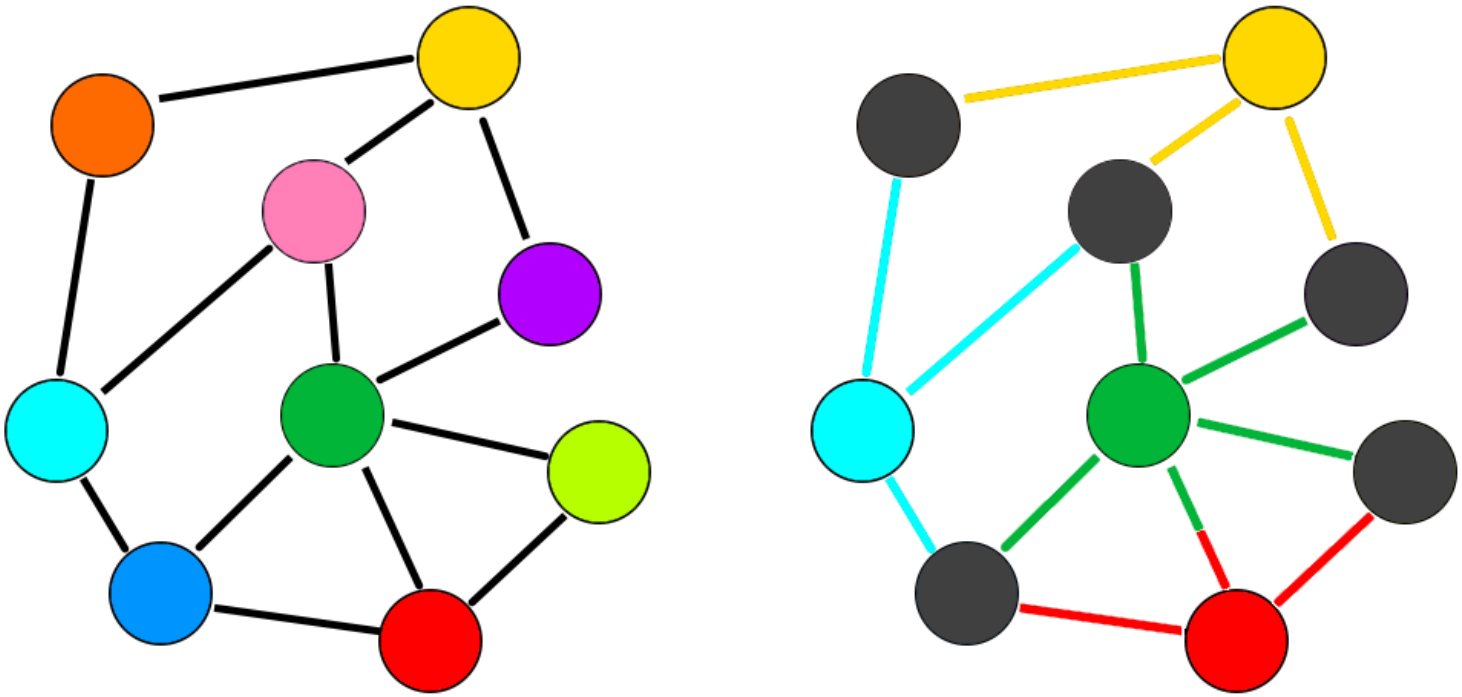
$$
\begin{aligned}
\min \quad & \sum_{s \in S} x_s \\
s.t. \quad & \sum_{s : u \in s} x_s \geq 1 \quad \forall u \in U \\
& x_s \in \{0, 1\} \quad \forall s \in S
\end{aligned}
$$

Vertex cover problem

In the vertex cover problem, we are given an undirected graph $G = (V, E)$.

The objective is to select a minimum subset V^* of V such that all edges $e_{i,j} = \{v_i, v_j\}$ in E have at least one vertex (i.e., either v_i, v_j, or both v_i and v_j) in V^*.

Vertex cover example



Vertex cover problem formulation

(Class exercise.)

Vertex cover problem formulation (solution)

$$
\begin{aligned}
\min \quad & \sum_{v \in V} x_v \\
s.t. \quad & x_{v_i} + x_{v_j} \geq 1 \quad \forall v_i, v_j : \{v_i, v_j\} \in E \\
& x_v \in \{0, 1\} \quad \forall v \in V
\end{aligned}
$$