Operations Research

Applications of Integer Programming

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Road map

- **Facility location problems**.
- Machine scheduling problems.
- Vehicle routing problems.

Facility location problems

- One typical managerial decision is "where to build my facility?"
  - Where to open convenience stores?
  - Where to build warehouses or distribution centers?
  - Where to build factories?
  - Where to build power stations, fire stations, or police stations?
- A similar question is "where to locate a scare resource?"
  - Where to put a limited number of fire engines or ambulances?
  - Where to put a limited number of police officers?
  - Where to put a limited number of ice cream machines?
- These problems are **facility location problems**.
  - In this lecture, we focus on **discrete** facility location problems: We choose a subset of locations from a set of finite locations.

Facility location problems

- In general, there are some **demand nodes** and some **potential locations**.
  - We build facilities at locations to serve demands.
  - E.g., build distribution centers to ship to retail stores.
  - E.g., build fire stations to cover cities, towns, and villages.
- Facility location problems are typically categorized based on their objective functions.
- In this lecture, we introduce three types of facility location problems:
  - **Set covering problems**: Build a minimum number of facilities to cover all demands.
  - **Maximum covering problems**: Build a given number of facilities to cover as many demands as possible.
  - **Fixed charge location problems**: Finding a balance between benefit of covering demands and cost of building facilities.

Set covering problems

- Consider a set of demands I and a set of locations J.
- The distance (or traveling time) between demand i and location j is $d_{ij} > 0$, $i \in I$, $j \in J$.
- A service level $s > 0$ is given: Demand i is said to be "covered" by location j if $d_{ij} < s$.
- Question: How to allocate as few facilities as possible to cover all demands?

Set covering problems

- Let's define the following parameter: $a_{ij} = 1$ if $d_{ij} < s$ or 0 otherwise, $i \in I$, $j \in J$.
- Let's define the following variables: $x_j = 1$ if a facility is built at location $j \in J$ or 0 otherwise.
- The complete formulation:

$$\begin{aligned} \min \quad & \sum_{j \in J} x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall i \in I \\ & x_j \in \{0, 1\} \quad \forall j \in J. \end{aligned}$$

- The weighted version: $\min \sum_{j \in J} w_j x_j$.

Maximum covering problems

- Consider a set of demands I and a set of locations J.
- The distances d_{ij}, service level s, and the covering coefficient a_{ij} are also given.
- We are restricted to build at most $p \in \mathbb{N}$ facilities.
- Question: How to allocate at most p facilities to cover as many demands as possible?

Maximum covering problems

- Still let $x_j = 1$ if a facility is built at location $j \in J$ or 0 otherwise.
- Also let $y_i = 1$ if demand $i \in I$ is covered by any facility or 0 otherwise.
- The complete formulation:

$$
\begin{aligned}
\max \quad & \sum_{i \in I} y_i \\
\text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \geq y_i \quad \forall i \in I \\
& \sum_{j \in J} x_j \leq p \quad \forall j \in J \\
& x_j \in \{0, 1\} \quad \forall j \in J \\
& y_i \in \{0, 1\} \quad \forall i \in I.
\end{aligned}
$$

- The weighted version: $\max \sum_{i \in I} w_i y_i$.

Fixed charge location problems

- Consider a set of demands I and a set of locations J.
- At demand i, the demand size is $h_i > 0$.
- The unit shipping cost from location j to demand i is $d_{ij} > 0$.
- The fixed construction cost at location j is $f_j > 0$.
- Question: How to allocate some facilities to minimize the total shipping and construction costs?

Fixed charge location problems

- We still need x_js: $x_j = 1$ if a facility is built at location $j \in J$ or 0 otherwise.
- We now need y_{ij}s: $y_{ij} = 1$ if demand $i \in I$ is served by facility at location $j \in J$ or 0 otherwise.
- The complete formulation:

$$
\begin{aligned}
\min \quad & \sum_{i \in I} \sum_{j \in J} h_i d_{ij} y_{ij} + \sum_{j \in J} f_j x_j \\
\text{s.t.} \quad & y_{ij} \leq x_j \quad \forall i \in I, j \in J \\
& \sum_{j \in J} y_{ij} = 1 \quad \forall i \in I \\
& x_j \in \{0, 1\} \quad \forall j \in J \\
& y_i \in \{0, 1\} \quad \forall i \in I.
\end{aligned}
$$

Fixed charge location problems

- The previous model is the **uncapacitated** version.
  - A facility can serve any amount of demand.
- If facility at location j has a limited capacity $K_j > 0$, we may add the capacity constraint

$$\sum_{i \in I} h_i y_{ij} \leq K_j \quad \forall j \in J.$$

- The **capacitated** version is usually called the capacitated facility location problem (abbreviated as CFL). The uncapacitated one is abbreviated as UFL.

Remarks

- When to use set covering?
  - When we are required to take care of (almost) everyone.
  - E.g., fire stations and police stations.
- When to use maximum covering?
  - When budgets are limited.
  - E.g., cellular data networks.
- When to use fixed charge location?
  - When service costs depends on distances.
  - E.g., distribution centers.
- All the three models are **NP-hard**.
  - For large instances, it really takes time to obtain an optimal solution.
  - Many researchers look for effective heuristics for these problems.

Road map

- Facility location problems.
- **Machine scheduling problems**.
- Vehicle routing problems.

Machine scheduling problems

- In many cases, **jobs/tasks** must be assigned to **machines**.
- As an example, consider a factory producing one product for n customers.
  - Serial production: Only one job can be processed at one time.
  - Each job has its due date.
  - How to schedule the n jobs to minimize the total number of delayed jobs?
- In this example, scheduling is nothing but **sequencing**.
  - Splitting jobs is not helpful.
  - There are $n!$ ways to sequence the n jobs.
  - Is there a polynomial-time algorithm?
- The problems of scheduling jobs to machines are **machine scheduling problems**.

Machine scheduling problems

- Machine scheduling problems can be categorized in multiple ways:
- Production mode:
  - Single machine serial production.
  - Multiple parallel machines.
  - Flow shop problems.
  - Job shop problems.
- Job splitting:
  - Non-preemptive problems.
  - Preemptive problems.
- Performance measurement:
  - Makespan (the time that all jobs are completed).
  - (Weighted) total completion time.
  - (Weighted) number of delayed jobs.
  - (Weighted) total lateness.
  - (Weighted) total tardiness.
- And more.

Minimizing total tardiness on a single machine

- Consider scheduling n jobs on a single machine.
- Job $j \in J = \{1, 2, ..., n\}$ has **processing time** p_j and due time d_j.
- Different schedules give these jobs different **completion times**. The completion time of job j is denoted as C_j.
- For job j, its **tardiness** is[1]

$$T_j = \max\{C_j - d_j, 0\}.$$

- There is only one machine, which can process only one job at a time.
- How to schedule all the jobs to minimize the total tardiness $\sum_{j \in J} T_j$?
- While many researchers study specific properties and algorithms for specific problems, we will only try to formulate the problem as an integer program.

[1]Its **lateness** is $L_j = C_j - d_j$, which may be negative.

Minimizing total tardiness on a single machine

- Let's use C_j to be our decision variables.
- Suppose we schedule jobs 1, 2, ..., and n in this order, we will have $C_1 = p_1$, $C_2 = p_1 + p_2$, ..., and $C_n = \sum_{i=1}^{n} p_i$.
- A **Gantt chart** is helpful to illustrate a schedule.
- Obviously, splitting jobs does not help for this problem. (Why?)
- Because the machine can start job 2 only after job 1 is completed, we have $C_2 \geq C_1 + p_2$ as a constraint. But what if job 2 should be scheduled before job 1?

Minimizing total tardiness on a single machine

- In a feasible schedule, job i is either before or after job j, for all $j \neq i$.
- Therefore, we need to satisfy at least one of the following two constraints:

$$C_j \geq C_i + p_j \quad \text{and} \quad C_i \geq C_j + p_i.$$

- Let $z_{ij} = 1$ if job j is before job i or 0 otherwise, $i \in J$, $j \in J$, $i < j$.
- The constraints we need:

$$\begin{aligned} C_i + p_j - C_j &\leq M z_{ij} \\ C_j + p_i - C_i &\leq M(1 - z_{ij}) \end{aligned}$$

- What value of M works?
  - How about $M = \sum_{j \in J} p_j$?

Minimizing total tardiness on a single machine

- It remains to linearize the objective function

$$\min \sum_{j \in J} \max\{C_j - d_j, 0\}.$$

- The complete formulation:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j \in J} T_j & \\
\text{s.t.} & T_j \geq C_j - d_j & \forall j \in J \\
& C_i + p_j - C_j \leq M z_{ij} & \forall i \in J, j \in J, i < j \\
& C_j + p_i - C_i \leq M(1 - z_{ij}) & \forall i \in J, j \in J, i < j \\
& T_j \geq 0, C_j \geq 0 & \forall j \in J \\
& z_{ij} \in \{0, 1\} & \forall i \in J, j \in J, i < j.
\end{array}
$$

Minimizing makespan on parallel machines

- Consider scheduling n jobs on m **parallel** machines.
- Job $j \in J = \{1, 2, ..., n\}$ has processing time p_j.
- The capacity of machine $i \in I = \{1, 2, ..., m\}$ is unlimited.
- A job can be processed at any machine. However, it can be processed only on one machine.

- Different schedules give these jobs different completion times C_js.
- The **makespan** of a schedule is $\max_{j \in J} C_j$.
- How may we minimize the makespan?

Minimizing makespan on parallel machines

- As long as some jobs are assigned to a machine, the sequence on that machine does not matter.
- The problem of minimizing makespan is just to **assign** jobs to machines.
- Let $x_{ij} = 1$ if job $j \in J$ is assigned to machine $i \in I$ or 0 otherwise.
- On machine $i \in I$, the last job is completed at

$$\sum_{j \in J} p_j x_{ij}.$$

- The objective is to

$$\min \max_{i \in I} \left\{ \sum_{j \in J} p_j x_{ij} \right\}.$$

How to linearize it?

Minimizing makespan on parallel machines

- The complete formulation is

$$
\begin{aligned}
\min \quad & M \\
\text{s.t.} \quad & M \geq \sum_{j \in J} p_j x_{ij} && \forall i \in I \\
& \sum_{i \in I} x_{ij} = 1 && \forall j \in J \\
& x_{ij} \in \{0, 1\} && \forall i \in I, j \in J.
\end{aligned}
$$

- Sometimes people want to maximize the completion time of the least-loaded machine (for, e.g., fairness):

$$
\begin{aligned}
\max \quad & M \\
\text{s.t.} \quad & M \leq \sum_{j \in J} p_j x_{ij} && \forall i \in I \\
& x_{ij} \in \{0, 1\} && \forall i \in I, j \in J.
\end{aligned}
$$

Road map

- Facility location problems.
- Machine scheduling problems.
- **Vehicle routing problems**.

Vehicle routing problems

- In many cases, we need to deliver/collect items to/from customers in the most efficient way.
- E.g., consider a post officer who needs to deliver to four addresses.
- The shortest path between any pair of two addresses can be obtained.
- This is a **routing** problem: To choose a route starting from the office, passing each address exactly once, and then returning to the office.
- This is a sequencing problem; in total there are $4! = 24$ feasible routes.
- Which route minimizes the total distance (or travel time)?

Vehicle routing problems

- The problem described above is the famous **traveling salesperson problem**.
  - It assumes that the truck has ample capacity.
- Consider the truck towing bicycles in NTU. It must start at the car pound, pass several locations in NTU, and then return to the origin.
  - However, the truck capacity is quite limited (because too many people violate the parking regulation).
  - The driver needs to find **multiple** routes to cover all the locations.
- The traveling salesperson problem (TSP) is a special case of **vehicle routing problems**.

Traveling salesperson problem

- How to formulate the TSP into an integer program?
- Let's consider a directed complete network $G = (V, E)$.
  - There are n nodes and $n(n-1)$ arcs.
  - The arc weight for arc (i, j) is $d_{ij} > 0$.
- We select a few arcs in E to form a **tour**.
  - To form a tour, we need to select n arcs.
  - These n arcs should form a cycle passing all nodes.
- Let $x_{ij} = 1$ if arc $(i, j) \in E$ is selected or 0 otherwise.
  - The objective:

$$\min \sum_{(i,j)\in E} d_{ij} x_{ij}.$$

  - How to ensure the routing requirement?
  - Is $\sum_{(i,j)\in E} d_{ij} x_{ij} = n$ enough?

Traveling salesperson problem

- For node $k \in V$:
  - We must select exactly one incoming arc:

$$\sum_{i \in V, i \neq k} x_{ik} = 1.$$

  - We must select exactly one outgoing arc:

$$\sum_{j \in V, j \neq k} x_{kj} = 1.$$

- Now each node is on a cycle.
- However, these are not enough to prevent **subtours**.

Eliminating subtours: alternative 1

- There are at least two ways to eliminate subtours.
- For each **subset of nodes** with at least two nodes, we limit the maximum number of arcs selected:

$$\sum_{i \in S, j \in S, i \neq j} x_{ij} \leq |S| - 1 \quad \forall S \subsetneq V, |S| \geq 2.$$

- When we have n nodes, we have $2^n - n - 1$ constraints.

Eliminating subtours: alternative 2

- Let u_is represent the order of passing nodes. More precisely, $u_i = k$ if node i is the kth node to be passed in a tour.
- We add the following constraints:

$$\begin{aligned} u_1 &= 1 \\ 2 \leq u_i &\leq n \quad \forall i \in V \setminus \{1\} \\ u_i - u_j + 1 &\leq (n-1)(1 - x_{ij}) \quad \forall (i,j) \in E, i \neq 1, j \neq 1. \end{aligned}$$

  - If $x_{ij} = 0$, there is no constraint for u_i and u_j; otherwise, u_j must be larger than u_i by at least 1.
  - If a tour does not contain node 1, the last constraint pushes those u_is to infinity and violates constraint 2.
  - Note that only node 1 is not restricted by these constraints!
- When we have n nodes, we have n additional variables and $n + (n-1)(n-2)$ constraints.

The complete formulation

- The complete formulation is

$$
\begin{array}{lll}
\min & \sum_{(i,j)\in E} d_{ij}x_{ij} & \\
\text{s.t.} & \sum_{i\in V, i\neq k} x_{ik} = 1 & \forall k \in V \\
 & \sum_{j\in V, j\neq k} x_{kj} = 1 & \forall k \in V \\
 & x_{ij} \in \{0,1\} & \forall (i,j) \in E.
\end{array}
$$

  with either alternative 1 or alternative 2.
- Which alternative is better?